REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 03/31/06	**2. REPORT TYPE** Final Project Report	**3. DATES COVERED** *(From - To)* 01/01/03-03/31/06
4. TITLE AND SUBTITLE Automatic Extraction and Coordination of Audit Data and Features for Intrusion and Damage Assessment		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER F49620-03-1-0109
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Dr. Nong Ye		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Arizona State University, Department of Industrial Eng., PO BOX 875906, Tempe, AZ 85287-5906		**8. PERFORMING ORGANIZATION REPORT NUMBER** XSA3300/TE
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR, ATTN: Dr. Robert Herklotz, 4015 Wilson Boulevard,Rm. 713, Arlington, VA 22203-1954		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE - Distribution A

AFRL-SR-AR-TR-06-0118

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Intrusion and damage assessment is an important step after intrusion detection. The goal of this research is to shorten time of constructing a coherent intrusion trace and assessing the damage through automatic extraction and coordination of audit data and features for intrusion and damage assessment. In this project, we develop the System-Fault-Risk framework to define cause-effect chains of intrusions as intrusion profiles and also classify intrusions. We create a new attack-norm separation approach to developing detection models for building cyber sensors monitoring and identifying intrusion data characteristics at various points along the path of an intrusion cause-effect chain. Mean, autocorrelation and wavelet data characteristics of cyber attack and norm data are discovered to enable the definition of attack data models and norm data models which are in turn used to build detection models for cyber sensors. The testing results the superior performance of detection models based on the attack-norm separation approach to that of detection models based on two conventional approaches of signature recognition and anomaly detection.

15. SUBJECT TERMS
Intrusions, intrusion detection, damage assessment

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Nong Ye
a. REPORT UNCLASS	**b. ABSTRACT** UNCLASS	**c. THIS PAGE** UNCLASS	UNLIMITED	93+134	**19b. TELEPHONE NUMBER** *(include area code)* 480-965-7812

FINAL PROJECT REPORT

Automatic Extraction and Coordination of Audit Data and Features for Intrusion and Damage Assessment

GRANT NO.:	**F49620-03-1-0109**
PI:	**Nong Ye**
SPONSOR:	**AFOSR**
PROJECT PERIOD:	**1 January 2003 – 31 March 2006**

Table of Contents

List of Figure Captions

Figure 5.10. Cuscore charts for the variable, Network Interface\ Bytes Received/sec.

List of Table Captions

Abstract

Intrusion and damage assessment is an important step after intrusion detection. The goal of this research is to shorten time of constructing a coherent intrusion trace and assessing the damage through automatic extraction and coordination of audit data and features for intrusion and damage assessment. In this project, we develop the System-Fault-Risk framework to define cause-effect chains of intrusions as intrusion profiles and also classify intrusions. We create a new attack-norm separation approach to developing detection models for building cyber sensors monitoring and identifying intrusion data characteristics at various points along the path of an intrusion cause-effect chain. Mean and autocorrelation data characteristics of cyber attack and norm data are discovered to enable the definition of attack data models and norm data models which are in turn used to build detection models for cyber sensors. The testing results the superior performance of detection models based on the attack-norm separation approach to that of detection models based on two conventional approaches of signature recognition and anomaly detection.

Objectives

As soon as an intrusion (cyber attacks) into a computer and network system is detected, it is necessary to construct a coherent trace of the intrusion and assess the extent (e.g., location and severity) of the intrusion and associated damages. Intrusion detection triggers intrusion and damage assessment that in turn provides the basis for intrusion reaction to recover the system and prevent future reoccurrences of the same intrusion. Without intrusion and damage assessment, it is difficult to determine corrective actions that will not only recover the system but also prevent future reoccurrences of the same intrusion. Although the system can be recovered to a pre-intrusion state by simply loading backup system data and reinstalling system programs, this recovery method leaves the system vulnerable to reoccurrences of the same intrusion that can immediately bring the system back to a compromised state. Hence, intrusion and damage assessment is an important step after intrusion detection.

Intrusion and damage assessment requires the identification of system data and intrusion data characteristics from many different locations, times and sources. Hence, large amounts of system data from many different locations, times and sources must be processed to extract a small amount of useful information for constructing a coherent intrusion trace and assessing the damage. Currently we are counting on system administrators to manually perform intrusion and damage assessment. However, the manual approach to intrusion and damage assessment has serious drawbacks. First of all, system administrators may not have time to perform the intrusion and damage assessment thoroughly or at all. Secondly, many intrusions are programmed and thus proceed at a fast machine speed. However, humans perform the intrusion and damage assessment at a much slower speed. To stop the further development of an intrusion while system administrators are performing the intrusion and damage assessment, system administrators usually have to temporarily control the intrusion by disabling a network service, disconnecting a computer, and so on. Those actions of temporary intrusion control interrupt system services to legitimate users, which may not be desirable. To sustain system services under the impact of an intrusion, we must shorten time to construct a coherent intrusion trace and assess the damage so that we can quickly react to the intrusion for system recovery and intrusion prevention. Automatic intrusion and damage assessment is required to significantly shorten time to perform the intrusion and damage assessment.

Thus, we focus on intrusion and damage assessment; especially automatic extraction and identification of audit data, features and characteristics. The goal of this research is to shorten time of constructing a coherent intrusion trace and assessing the damage through automatic extraction and identification of audit data, features and attack data characteristics.

We take a directly diagnostic assessment approach; specifically, we collect useful audit data from computer and network systems from different locations, times and sources and extract mathematical/statistical features from the data. Then we monitor and identify mathematical/statistical features of audit data, which quantitatively characterize various intrusions along with their phases, locations and damages, resulting in the quick assessment of intrusions and damages. Hence, distinctive attack data characteristics play a central role in quickly directing the assessment of intrusions and their damages. Three important elements of this directly diagnostic assessment approach are:

1. Kinds of audit data that contain useful information for intrusion and damage assessment,
2. Mathematical/statistical features that quantitatively and uniquely characterize intrusions and their damages, and

3. Quick detection and identification of mathematical/statistical attack data characteristics for the directly diagnostic assessment of intrusions and their damages.

Therefore, the objectives of this research are to investigate, develop and test the following:

- Audit data that contain useful information for intrusion and damage assessment to reveal kinds of audit data (i.e., activity, state and performance data) in computer and network systems and their theoretical properties (i.e., stationarity, statistical distribution, independence, covariance and autocorrelation) in normal and intrusive conditions of system operation;
- Mathematical/statistical data features that can be automatically extracted from audit data to show data characteristics that quantitatively and uniquely characterize various intrusions and their damages, and thus direct intrusion and damage assessment;
- Analytical techniques that will enable the efficient incorporation of mathematical/statistical attack data characteristics into directed statistics that will quickly detect distinctive attack data characteristics for intrusion and damage assessment.

This research on audit data, features and attack data characteristics lays the scientific foundation for developing analytical techniques of quick intrusion and damage assessment. Hence, through the work on audit data, features, characteristics, and analytical techniques, we establish a coherent set of scientific theories and analytical techniques that enable the automatic extraction and identification of audit data and attack data characteristics for intrusion and damage assessment.

In Chapter 1, we present our work on cyber attack classification and attack profiling to capture cause-effect chains of cyber attacks, which in turn can be used to correlate results of cyber sensors monitoring and identifying attack data characteristics at various points of attack cause-effect chains. In Chapter 2, we describe our methods of attack experimentation and data collection which provide audit data for our investigation. In Chapter 3, we present audit data, their mathematical/statistical features, and attack and normal use (norm) data characteristics which are discovered from the collected data of cyber attack and norm activities. In Chapter 4, we present a new attack-norm separation approach to detecting and identifying cyber attacks based on attack and norm data characteristics. In Chapter 5, we provide the method of building cyber sensors for cyber attack detection and identification according to the attack-norm separation approach along with the testing results.

Chapter 1. Cyber Attack Classification and Profiling

In this chapter, we describe our research methodology for defining a cause-effect chain of a cyber attack.

1. Introduction

We established a System-Fault-Risk (SFR) framework to capture resource-process interactions, activity-state-performance interactions, and asset-vulnerability-threat interactions in the cause-effect chain of an attack (intrusion) propagation. Initially, we classified existing known cyber attacks and developed twelve attack profiles using the SFR framework. We identified observable points for cyber attack detection in each attack profile and defined data, features and characteristics at each observable point. We generalized a list of data, features and characteristics for cyber attack detection from those identified in the twelve attack profiles.

2. Related Work

Existing audit data include mainly network traffic data (data packets) from networks and computer audit/log data from host machines. Network traffic data and computer audit/log data capture activities in computer and network systems. Network traffic data and computer audit/log data are mainly system activity data. Our past work [1-13] has shown the effectiveness of system activity data for intrusion detection. For intrusion and damage assessment, system activity data may not be sufficient to reveal impacts and damages caused by intrusions [14-16]. A sophisticated intrusion can go through several phases, such as reconnaissance, scanning for vulnerabilities, gaining access, maintaining access, further attacks, and covering tracks [12]. An intrusion often produces a cause-effect propagation chain in a computer and network system.

Activities of an intrusion—the cause and the start of the cause-effect propagation chain—directly act on certain assets in a computer and network system and may bring those assets into a compromised state with degraded or discontinued performance. The compromised state and performance of those assets may in turn interrupt the state and performance of other associated assets. As a result, the effects of the intrusion propagate throughout a part or the entire body of the computer and network system. For example, an intrusion may directly interact with a network application program and bring it into an error state that the kernel of the operating system cannot handle and thus becomes stalled, which in turn causes the system crash, the discontinued services of other programs in the system, and other consequences.

To fully assess an intrusion and its damages, intrusion and damage assessment must uncover the cause-effect propagation chain produced by an intrusion. System activity data contain information about the cause in the cause-effect propagation chain. System state and performance data reveal the state and performance of assets (e.g., processes, CPU, memory, data files, and communication links) in a computer and network system, and thus contain information about the rest of the cause-effect propagation chain following the cause. That is, system state and performance data should play an important role in intrusion and damage assessment. Hence, if intrusion detection based on system activity data is to detect the cause in the cause-effect propagation chain, intrusion and damage assessment is to reveal the entire cause-effect propagation chain. In other words, intrusion and damage assessment extends the detection of changes in system activity (as we see in intrusion detection) to the detection of changes in not only system activity but also system state and performance. In addition, changes in system activity, state and performance must be correlated to reveal the entire cause-effect propagation chain. Hence, the proposed research will be carried out according to the concept that intrusion and damage assessment is to detect and correlate changes in system activity, state and performance, especially changes from normal conditions to intrusive conditions of system operation.

Therefore, we have expanded audit data to include system state and performance data along with system activity data for intrusion and damage assessment. In this report, we use the term "audit data" to refer to system activity, state and performance data.

Audit data come with large amounts. Not all information contained in audit data is useful for intrusion and damage assessment. Since some changes in system activity, state and performance are common in both normal and intrusive conditions of system operation, information that is useful in distinguishing between normal and intrusive conditions of system operation is important for intrusion and damage assessment and needs to be extracted from audit data.

Existing work on intrusion detection [12] has shown many pieces of information in system activity data that are useful in detecting intrusions through changes in system activity. For example, information from network traffic data, such as the destination port, source port, source IP address, TCP control bits, TCP sequence number, timestamp, IP fragment length, Type of Service (TOS), IP header length, commands, machine codes and file names in data payload, has been used in intrusion signatures for intrusion detection. For another example, types of commands, system call and audit events from computer audit/log data have been used in building normal activity profiles for detecting anomalies and thus intrusions.

However, little is known about what information about system state and performance is useful to reveal intrusion effects or what changes of system state and performance are induced by various intrusions. Hence, we have conducted research to identify information about system activity, state and performance that is useful to reveal cause-effect propagation chains of intrusions.

3. System-Fault-Risk Framework

In this section we describe the System-Fault-Risk (SFR) framework for classifying cyber attacks. The SFR framework incorporates a cause-effect chain into its design. In keeping with fault modeling theory, our classification is ordered in terms of cause and effect. The overall incident begins with a threat, followed by an attack. A threat is composed of the three columns: objective, propagation and attack origin. The attack includes action, vulnerability, asset, and state and performance effects. Additionally, the first six columns indicate the cause, while the last two the effect.

We identified a list of computer and network assets that are involved and affected in intrusions by analyzing existing known intrusion scenarios [17-22] and reviewing existing frameworks and analysis of computer and network vulnerabilities [23-27]. Computer and network assets include hardware and software assets, such as CPU, memory, data files, programs, routers, communication links, and so on. We defined operation, state and performance attributes of each identified asset according to the design specification of the asset and its operation. The operation, state and performance of an asset form a cause-effect chain for that asset. An operation on the asset changes the asset state that in turn affects the asset performance to serve its user.

As shown in Figure 1, the elements involved in the cause-effect chain of a cyber attacks are sorted into the categories: objective, propagation, attack origin, action, vulnerability, asset, state effects and performance effects. These categories make up an incident. The incident encompasses the two parts: threat and attack, and a cause-effect chain.

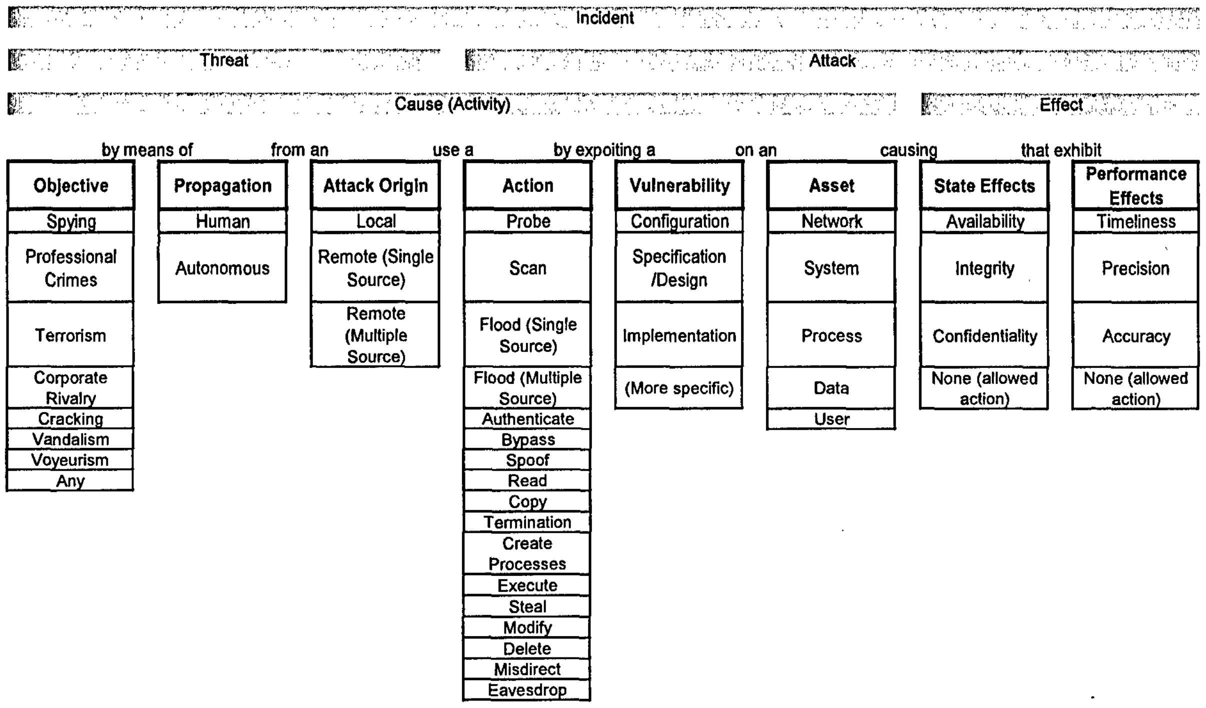


Figure 1.1. System-Fault-Risk framework.

The columns in Figure 1 show some examples for each category of each element. A more comprehensive or even specialized list of elements in each category can be created. We merely show some examples. Additionally, entries under each column can be modified or extended as cyber attacks evolve. We include further details of the SFR framework in our paper [28].

4. Attack Classification

The proposed framework can be used in many situations. For example, anyone designing an ID solution to protect against a subset of attacks can classify them using our framework and quickly reveal commonalities and differences within the set. Or, consider designing an ID solution to protect against a specific type of attack, such as DoS attacks. With this framework we can list all known DoS attacks, and exploit their similarities in an effort to generalize these types of attacks; Thereby providing a basis for detecting a novel attack that fits into the classification of a DoS attack. The framework provides a tool for simplifying this type work.

Table 1.1. Attack classification table.

Attack Name	**Objective**	**Propagation**	**Attack Origin**	**Action**	**Vulnerability**	**Asset**	**State Effect**	**Perf ormance Effect**
UDP Storm	Any	Human	Remote (Single Source)	Flood	Specification / Design	Network	Availability	Timeliness
Slammer Worm	Cracking	Autonomous	Remote (Single Source)	Copy	Implementation	Process	Integrity	Accuracy
Slammer Worm	Cracking	Autonomous	Local	Execute	Specification / Design	System	Integrity	Accuracy
ARP Poison	Corporate Rivalry, Cracking, Vandalism	Human	Remote (Single Source)	Misdirect	Specification / Design	Network	Availability, Integrity	Timeliness, Accuracy
Half Life Buffer Overflow	Cracking Vandalism	Human	Remote (Single Source)	Terminate	Implementation	Process	Availability, Confidentiality	Timeliness
NMAP Scanner	Any	Human	Remote (Single Source)	Probe, Scan	Specification / Design	Network	Confidentiality	None
EZPublish	Any	Human	Remote (Single Source)	Bypass, Steal	Implementation	Data	Confidentiality	None
Dictionary	Any	Human	Local	Steal	Specification / Design	Process	Availability, Confidentiality	Precision, Accuracy
Meteor FTP	Cracking Vandalism	Human	Remote (Single Source)	Terminate	Implementation	Process	Availability	Timeliness
Netbus Trojan	Any	Human	Remote (Single Source)	Bypass	Implementation	System	Integrity, Confidentiality	None
TCP Reset	Any	Human	Single Source	Terminate	Specification / Design	Network	Availability, Integrity	Timeliness
Photchat Cross Site Scripting	Any	Human	Remote (Single Source)	Bypass, Steal	Implementation	Data, Process	Integrity, Confidentiality	None
Process Table	Any	Human	Remote (Single/Multiple Source)	Flood	Implementation	System	Availability	Timeliness
Apache Web Server	Any	Human	Remote (Single Source)	Flood	Implementation	System, Data	Availability	Timeliness
Chat Server Abuse	Non-malicious	Human	Multiple Source	Misuse	Implementation	Network	Availability	Timeliness

We classified a set of example attacks used in our research to demonstrate one use of the SFR framework. From Table 1, we extract common details among our subset of attacks to aid in our understanding. Then, as we discover characteristics of attacks in our ongoing research, we compared those characteristics with those of similar attacks within the framework [29].

5. Attack Profiling

To derive the data observations of activity, state and performance changes that can help in detecting an attack while it is progressing along its cause-effect chain, the elements of a cyber attack in the SFR framework are enlarged in an attack profile, which includes an explicit description of the cause-effect chain along with observations of activity, state and performance changes. The method of attack profiling is explained in our paper and a thesis on the topic [30, 31]. Here we give an example using the attack profile for the Apache Denial of Service attack. Based on activity-state-performance changes, observation points can be selected such that monitoring the observation points will be useful in detecting the attack. An observation point could be an activity, a state change or a performance impact anywhere along the cause-effect chain for the attack shown in Figure 2.

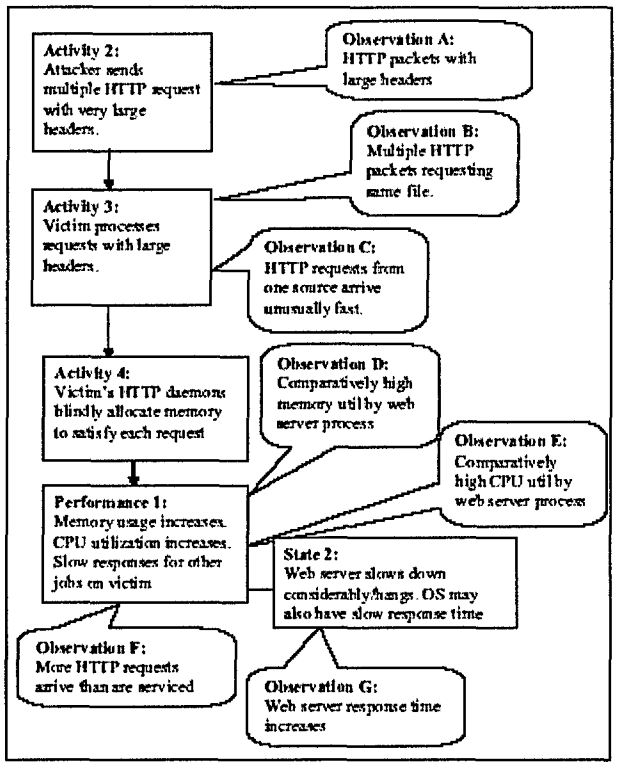

Activity 2: Attacker sends multiple HTTP request with very large headers.
Observation A: HTTP packets with large headers
Observation B: Multiple HTTP packets requesting same file.
Activity 3: Victim processes requests with large headers.
Observation C: HTTP requests from one source arrive unusually fast.
Activity 4: Victim's HTTP daemons blindly allocate memory to satisfy each request
Observation D: Comparatively high memory util by web server process
Observation E: Comparatively high CPU util by web server process
Performance 1: Memory usage increases. CPU utilization increases. Slow responses for other jobs on victim
State 2: Web server slows down considerably/hangs. OS may also have slow response time
Observation F: More HTTP requests arrive than are serviced
Observation G: Web server response time increases

Figure 1.2. Apache DOS attack profile.

Observation points for the attack can be refined further, in terms of how they can be identified by analyzing activity/state/performance data on the systems. Thus, each observation point can be defined in terms of the data which needs to be analyzed, its feature and the characteristic on the feature to detect the attack. In general, data is the raw data captured by computers/networks, feature is a measure from the data, such as individual observation, mean, variance, probability distribution, covariance, auto-correlation, dependency, etc. The characteristic of a given feature enables the distinction of an attack from normal system behavior, such as shift, trend (i.e., cyclic and seasonal), drift (i.e., upward and downward), intermittent spike or bump, change (i.e., step change, slope change, sine wave, and square wave), etc. Thus, a one-to-one mapping can be created between an observation point and its corresponding Data-Feature-Characteristic. For example, for the observation points of the Apache attack, Table 2 indicates their corresponding data, feature and characteristic.

Table 1.2. Apache DOS observable points as well as attack data, features and characteristics at these points.

OBS	*Location*	*Data*	*Feature*	*Characteristic*
A	l_1	HTTP GET message header size	EWMA	Step increase
	l_2	HTTP GET message header size with same DEST IP	EWMA	Step increase
B	l_1	Similarity score of filenames in consecutive HTTP GET messages	EWMA	Step increase
	l_2	Similarity score of filenames in consecutive HTTP GET messages with same DEST IP	EWMA	Step increase
C	l_1	Inter-Arrival Time of HTTP GET messages from same SRC IP	EWMA	Step decrease
	l_2	Inter-Arrival Time of HTTP GET messages from same SRC IP with same DEST IP	EWMA	Step decrease
D	l_1	Ratio of (Web server memory usage/Sum of all other processes memory usage)	EWMA	Step increase
E	l_1	Ratio of (Web server CPU usage/Sum of all other processes CPU usage)	EWMA	Step increase
	l_1	Ratio of count of HTTP GET/POST messages	EWMA	Step increase
	l_2	Ratio of count of HTTP GET/POST messages to/from same IP	EWMA	Step increase
G	l_1	Difference in arrival times of GET and corresponding POST HTTP messages	EWMA	Step increase
	l_2	Difference in arrival times of GET and corresponding POST HTTP messages to/from same IP	EWMA	Step increase

6. Summary

In this chapter we outline the SFR framework, attack classification, and attack profiling methods to define data, features and characteristics of a subset of known attacks. Cause-effect chains of attacks derived from such work will enable the correlation of results from cyber sensors automatically monitoring and identifying attack data characteristics at various locations of attack cause-effect chains. Our investigation of cyber sensors is presented in the following chapters. See Attachments A and B for two published/accepted papers with more details describing the work in this chapter.

References

1. N. Ye, and Q. Chen, "EWMA techniques for detecting computer intrusions through anomalous changes in event intensity." IEEE Transactions on Reliability, Vol. 52, No. 1, March 2003.
2. N. Ye, Y. Zhang, and C. M. Borror, "Robustness of the Markov chain model for cyber attack detection," IEEE Transactions on Reliability, Vol. 52, No. 1, March 2003.
3. N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, "Multivariate statistical analysis of audit trails for host-based intrusion detection." IEEE Transactions on Computers, Vol. 51. No. 7, 2002, pp. 810-820.
4. N. Ye, T. Ehiabor, and Y. Zhang, "First-order versus high-order stochastic models for computer intrusion detection." Quality and Reliability Engineering International, Vol. 18, No. 3, 2002, pp. 243-250.

5. X. Li, and N. Ye, "Grid- and dummy-cluster-based learning of normal and intrusive clusters for computer intrusion detection." Quality and Reliability Engineering International, Vol. 18, No. 3, 2002, pp. 231-242.

6. S. M. Emran, and N. Ye, "Robustness of chi-square and Canberra techniques in detecting intrusions into information systems." Quality and Reliability Engineering International, Vol. 18, 2002, pp. 19-28.

7. N. Ye, X. Li, Q. Chen, S. M. Emran, and M. Xu, "Probabilistic techniques for intrusion detection based on computer audit data." IEEE Transactions on Systems, Man, and Cybernetics, Vol. 31, No. 4, 2001, pp. 266-274.

8. N. Ye, and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems." Quality and Reliability Engineering International, Vol. 17, No. 2, 2001, pp. 105 - 112.

9. X. Li, and N. Ye, "Decision tree classifiers for computer intrusion detection." Journal of Parallel and Distributed Computing Practices, accepted.

10. N. Ye, and X. Li, "A supervised, incremental learning algorithm for classification problems." Computers & Industrial Engineering Journal, accepted.

11. N. Ye, C. Borror, and Y. Zhang, "EWMA techniques for computer intrusion detection through anomalous changes in event intensity." Quality and Reliability Engineering International, accepted.

12. N. Ye, Computer Intrusion Detection: Data, Algorithms and Applications. London: Springer Verlag, 2003.

13. S. M. Emran, and N. Ye, "A system architecture for computer intrusion detection." Information, Knowledge, Systems Management, Vol. 2, No. 3, 2001, pp. 271-290.

14. N. Ye, "QoS-centric stateful resource management in information systems." Information Systems Frontiers, Vol. 4, No. 2, 2002, pp. 145-156.

15. N. Ye, J. Giordano, and J. Feldman, "A process control approach to cyber attack detection." Communications of the ACM, Vol. 44, No. 8, 2001, pp. 76-82.

16. N. Ye, "Robust intrusion tolerance for information systems." Information Management and Computer Security, Vol. 9, No. 1, 2001, pp. 38-43.

17. Dr-K, A Complete Hacker's Handbook. Carlton Books, Great Britain, 2000.

18. D. H. Freedman, and C. C. Mann, @Large: The Strange Case of the World's Biggest Internet Invasion. Simon & Schuster, 1997.

19. S. Garfinkel, and G. Spafford, Practical UNIX & Internet security. O'Reilly & Associates, Cambridge, 1996.

20. S. Northcutt, M. Cooper, M. Fearnow, K. Frederick, Intrusion Signatures and Analysis. New Riders, Indianapolis, Indiana, 2001.

21. E. Skoudis, Counter Hack. Prentice Hall PTR, Upper Saddle River, New Jersey, 2002.

22. J. Viega, and G. McGaw, Building Secure Software. Addison-Wesley, Boston, 2002.

23. R. P. Abbott, J. S. Chin, J. E. Donnelley, W. L. Konigsford, S. Tokubo, D. A. Webb, Security Analysis and Enhancements of Computer Operating Systems, NBSIR 76-1041, Institute for Computer Sciences and Technology, National Bureau of Standards, 1976.

24. T. Aslam, A Taxonomy of Security Faults in the UNIX Operating System. Master Thesis, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, 1995.

25. I. Krsul, Software Vulnerability Analysis. Department of Computer Sciences, Purdue University, West Lafayette, Indiana, 1998.

26. W. Du, A. P. Mathur, "Testing for software vulnerability using environment perturbation," Quality and Reliability Engineering International 18, 2000.

27. C. E. Landwehr, "A taxonomy of computer program security flaws," Computing Surveys 26:211-255, 1994.

28. N. Ye, C. Newman and T. Farley, "A System-Fault-Risk Framework for Cyber Attack Classification and Profiling," Information, Knowledge, Systems Management, accepted.

29. N. Ye and T. Farley, Cyber Signal Detection: A New Approach to Intrusion Detection, Computer, 38:11, November 2005, pp.55-61.

30. N. Ye, H. Bashettihalli and T. Farley, "Attack Profiles to Derive Data Observations, Features, and Characteristics of Cyber Attacks," Information, Knowledge, Systems Management, 5:1, 2005-2006, pp. 23-47.

31. Harish, B (2004). Cyber attack profiling using cause-effect networks (Master's Thesis). Tempe, AZ: Arizona State University.

Chapter 2. Attack-Norm Separation Approach

Cyber attacks on computers and networks exploit vulnerabilities in the information infrastructure and pose threats to online transactions, critical operations such as control of power supply, and many other activities that rely on this infrastructure. There are a number of defense mechanisms to protect computers and networks from such cyber attacks. These mechanisms generally serve one of three purposes: prevention, detection, or reaction [1-4]. Prevention mechanisms, such as firewalls, control or limit access to a computer and network system. Prevention raises the difficulty in launching attacks, but cannot completely block attacks from especially determined, organized, skilled attackers. Detection mechanisms monitor activities on computers or networks to identify the activities of an ongoing attack. Reaction mechanisms control the further spreading of an attack and its impact, then trace and diagnose the attack to determine its path, cause, and consequences, and finally take actions to recover systems and correct problems along the cause-effect path. This effort focuses on detection mechanisms, and presents a new approach to cyber attack detection

1. Conventional Approaches

Most of cyber attack detection systems fall into two conventional cyber detection approaches–Signature recognition and Anomaly detection [1-2]. The description of each of these approaches is provided below.

1.1 Signature Recognition

A signature of an attack represents the characteristics of the attack and is extracted either by human analysts or automatically discovered through data mining of computer activity data collected under attack and normal operating conditions [1, 4]. Attack signatures are stored and used in a cyber attack detection system to find matches of the signatures. If a signature is present, the system detects an attack. For example, three incorrect passwords can be used as a signature of a password-guessing attack. Thus, a detection system monitors the number of consecutive login failures, and compares it with the signature to detect this attack. Signature recognition [1] fails if an attack is novel and thus its signatures are unknown.

1.2 Anomaly Detection

In this approach, any large deviation from normal behavior is considered as a cyber attack [2]. For example, to detect a masquerader a norm profile is constructed for the normal user and any large deviation by the masquerader from the norm profile will be detected as an attack. An anomaly detection [3] technique can detect a novel attack if it shows a large deviation from the norm profile. However, an irregular normal activity may produce a large deviation and thus a false alarm. Hence, anomaly detection has not gained much practical use in the cyber attack detection arena due to the high workload associated with many false alarms.

1.3 Shortcomings of Conventional Approaches

Ye and Farley [4] discuss the following shortcomings of the signature recognition and anomaly detection approaches. Existing attack detection systems mostly use network traffic data to monitor activities on networks and audit/log data to monitor activities on computers. Since attacks may occur in an intermittent manner, skipping any data packet on a network or any event on a computer presents the risk of missing a critical step of an attack. On the other hand, the continuous monitoring of all network data packets and computer events requires processing large volume of data and thus presents a challenge in the computational efficiency of cyber detection systems.

Neither of the two approaches (signature recognition and anomaly detection) requires both attack and norm models of cyber data. Neither the signature recognition nor the anomaly detection performs the separation of attack data and norm data by filtering out norm data from the observed data mixture of attack and normal user activities to address the problem of the weakened attack or norm characteristic in the mixed data. The mixture of attack data and norm data or presence of norm data in the attack data may weaken the distinctive characteristics of attack data and/or norm data, resulting in poor detection performance of signature recognition and anomaly detection that rely on only one model or the characteristic of one data element, attack data or norm data.

Furthermore, existing signature recognition and anomaly detection techniques are mostly developed empirically, rather than based on the scientific understanding of the attack and normal data. In fact, there exists little scientific knowledge of attack and normal data in the field of cyber attack detection.

Another shortcoming of current solutions to cyber attack detection is in their reliance on only activity data on computers and networks without state and performance data. The execution of a user's process (an activity, which may be an attacker's activities) on a resource changes the state of that resource, which in turn affects the performance of the process. This state and performance change may propagate to other resources and processes. Thus, state and performance data are parts of the cause-effect chain or network induced by an attack, and can be helpful in attack detection.

2. Attack-Norm Separation Approach

2.1 Overview

Let us consider attack data as a signal to detect and normal user data as noise mixed with the signal in cyber space. Then, there exists a mapping between cyber attack detection and signal detection in the physical space. To overcome the shortcomings of the two conventional approaches, a new attack-norm separation approach has been developed [4] to base the attack detection on the principles of signal detection in the physical space, which employ noise cancellation to improve the signal-to-noise ratio of the mixed signal-noise data before signal detection. This approach performs three important steps for attack detection. First, the models of attack and normal user (norm) data are defined. Secondly, the norm data is filtered out from the observed data (mixed attack and norm data when an attack occurs), using the norm model. Finally, the attack is identified in the residual data, using the attack model.

For further analysis, let us study the cuscore model [5] developed for detecting a sine wave signal buried in random noise that fluctuates around the level of T in the physical space. The cuscore model considers the following noise and signal models [4-5]:

Norm model: $y_t = T + a_{t0}$ (2.1)

Attack model: $y_t = T + \delta \sin x_t + a_t$ (2.2)

where a_{t0} and a_t are white noise components. The cuscore is [4]:

$$Q = \sum_t a_{t0} r_t = \sum_t (y_t - T)\frac{(a_{t0} - a_t)}{\delta} = \sum_t (y_t - T)\frac{\delta \sin x_t}{\delta} = \sum_t (y_t - T)\sin x_t . \quad (2.3)$$

The y_t in each case represents the time series input data, x_t is the series of values for the sine component of the attack signal, r_t is the rate of signal change and δ represents the time step involved in the sine wave. Here noise cancellation is performed through (y_t-T) based the norm model, and signal identification is performed by correlating the residual data from (y_t-T) to the rate of signal change, r_t, specifically, sin x_t based on the signal model.

A collection of attack detection models are then constructed, which cover different attacks occurring in various norm environments. Each cyber attack detection model performs the monitoring and processing of only a small amount of specific data which manifest certain characteristics of attack and norm. Hence, each model is efficient, accurate, and adequate in detecting a special data characteristic of a given attack in a certain norm environment. A comprehensive knowledge of cyber attack and norm data characteristics will establish a solid, scientific foundation of cyber attack detection to overcome the shortcomings of existing empirical techniques.

2.2 Data, Features, Characteristics and Detection Models of Cyber Attack and Norm

An attack detection model based on the attack-norm separation approach requires a thorough, scientific understanding of cyber attack data and norm data. As stated earlier, since most existing work on cyber attack detection is empirical in nature, we currently have little scientific knowledge of cyber attack and norm data characteristics. Hence, it becomes imperative to obtain the scientific understanding of cyber attack and norm data characteristics to enable the new approach.

Three elements need to be defined for a cyber attack in a given norm environment to build an attack detection model: data, features, and characteristics [4]. A characteristic of cyber attack or norm is defined on a feature of a data variable. Data must be relevant to cyber attack detection, and may include data variables representing activity, state performance changes of computers and networks. A feature is a measure from an individual data observation or multiple data observations. Features may address mathematical, statistical, spatial, temporal, or causal properties of data observation(s) (e.g., such statistics as mean, variance, correlation, autocorrelation, transition probability, and others). A characteristic is a change on a given feature that enables the distinction of cyber attack from cyber norm. Characteristics may be shift (e.g., step change), intermittent spike or bump, drift (i.e., upward and downward), trend (e.g., slope, sine wave, square-wave, cyclic, and seasonal change), etc. Table 2.1 illustrates an example of these three elements and associated signal detection models in the physical space for the radar detection of a special object in the air, and in cyber space for the detection of the Denial of Service (DoS) attack [4].

Figure 2.1 illustrates these three elements along with a signal detection model. In Figure 2.1, raw data (e.g., network traffic data) collected from computers and networks go through data processing or screening to obtain a given data variable (e.g., the intensity ratio of packets for the web server to all packets) from which the feature (e.g., an arithmetic calculation of the sample average) is extracted using a feature extraction method. The signal detection model incorporates and monitors the characteristic change of attack data from norm data on the feature to detect the characteristic change of cyber attack from norm and decide if a cyber attack is present in the mixed data of attack and normal activities.

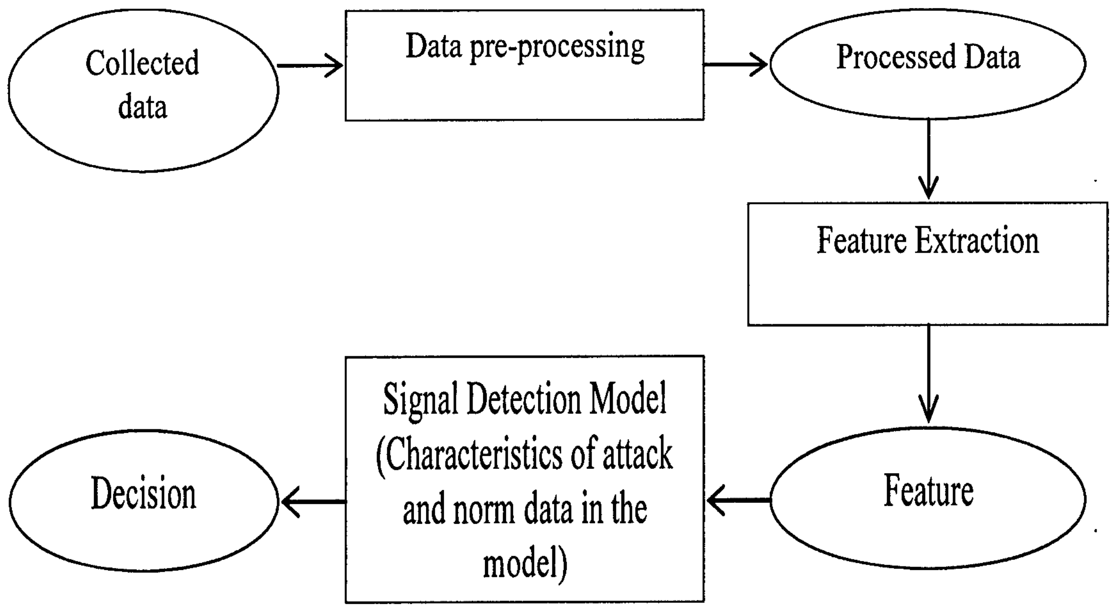


Figure 2.1 Data, features, characteristics, and signal detection models in the attack-norm separation approach.

Table 2.1. Examples of data, features, characteristics, and signal detection models in the physical and cyber spaces.

Element	**Physical Space**	**Cyber Space**
Data	Image data	UDP Datagrams sent/sec
Feature	Color and shape	Paul wavelet
Characteristic	Color is blue and shape is round	Spike
Signal Detection Model	A rule-based model: if color is blue & shape is round, then signal	Cuscore model for spike

3. Summary

This chapter introduces a new attack-norm separation approach to cyber attack detection and identification in comparison with two conventional approaches of signature recognition and anomaly detection. More details can be found in [4]. See Attachment C for a published paper with more details describing the work in this chapter.

References

1. N.Ye and X.Li, "A Scalable Clustering Technique for Intrusion Signature Recognition," *In Proceedings of the 2001 IEEE Information Assurance Workshop*, pages 1-4, United States Military Academy West Point, New York, IEEE press, NJ, June 5-6, 2001.
2. N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detection intrusions into information systems," *Quality and Reliability Engineering International*, Vol. 17, No. 2, 2001, pp. 105 - 112.
3. N. Ye, Q. Chen, S.M. Emran and S. Vilbert, "Hotelling's 2T multivariate profiling for anomaly detection," *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security United States Military Academy*, West Point, NY, IEEE press, NJ, 6-7 June, 2000. pp. 175-181.
4. N. Ye and T. Farley, "A scientific approach to cyberattack detection." *IEEE Computer, Vol. 38, No. 11, 2005, pp. 71-77.*
5. George Box and Albert Luceno, "Statistical Control by monitoring and Feedback Adjustment," Wiley publishers, NY, 1997.

Chapter 3. Collection of Cyber Attack and Norm Data

We execute a number of attacks on Windows-based computer networks and collect activity, state and performance data on attacker and victim computers as well as the network. We also collect activity, state and performance data on the same computer network in normal active use or inactive conditions. This chapter describes these attack and norm activities along with data collection.

1. Attack and Normal Use Activities

The attacks considered in this study include NMAP, Ettercap, Hardware Key logger, Nessus, Software Key logger, Remote dictionary, Root Kit, Apache, War (FTP), Process Table and DoS massive input traffic. The norm activities are Web browsing and Text editing. A total of eleven attacks on the background of two norm activities are tested to verify the attack detection performance. Hence, thirteen different activities are run on a testbed of a computer network to collect the data. This set of thirteen activities contains eleven attacks and two normal activities. A list of all these activities is given in Table 3.1. Each of these activities is described in detail and also the methodology for running each activity is explained.

Table 3.1. A list of attack and normal use activities.

Number	**Activity name**	**Type**
1	NMAP	Attack
2	Ettercap	Attack
3	Hardware Key logger	Attack
4	Nessus	Attack
5	Software key logger	Attack
6	Remote Dictionary	Attack
7	Root Kit	Attack
8	Apache	Attack
9	War(FTP)	Attack
10	Process table	Attack
11	DoS Massive input Traffic	Attack
12	Web Browsing	Norm
13	Text editing	Norm

1.1 NMAP Port Scan

NMAP is primarily used to find open ports on a computer as well as software and the operating system running on those ports [1]. It first probes each port on the computer to find if the port is open. Once it finds open ports, it will interrogate the port with a number of protocols in an attempt to determine the service provided by the port. Often, NMAP can determine the software and its version that is running.

1.2 Ettercap

Ettercap is one of the ARP poison attacks [2]. It can be used to redirect network traffic through an attacker's computer. This will allow the attacker to see even encrypted traffic. When started, Ettercap will send out an "ARP storm". This ARP storm consists of a series of ARP requests; one goes out to every IP address on the current subnet. Although noisy, this is an efficient method to determine which computers are currently on the network. When instructed to start poisoning, Ettercap will send out spoofed ARP-replies about every 10 seconds. Victim constantly receives ARP responses containing IP address of other computers on the network, and with attacker's MAC address instead, and updates its ARP table accordingly. After that, network traffic sent by all computers within the victim network goes through the attacker's machine. Ettercap automatically pulls out usernames and passwords. It also has the ability to filter and inject network traffic.

1.3 Hardware Key Logger

The keykatcher 64K mini plugs in between the back of the computer and the keyboard [3]. It intercepts all the keystrokes. Users will apply their own unique password to their keykatch. When this password is typed in, a menu will appear on the screen inside whichever text-editor they are currently using (such as notepad). This menu has options to display all the recorded keystrokes, to search through its memory looking for web-site URLs that were visited, as well as the ability to delete all keystrokes from memory.

1.4 Nessus

Nessus is an automated security auditor [4]. Given a range of computers to scan, it will create a report detailing the security vulnerabilities of each computer. Nessus can test for numerous vulnerabilities. Plug-in are released all the time for Nessus, enabling it to find more vulnerabilities.

1.5 Software Key Logger

Windows Key logger 5.0 will "trap" system calls [5]. System calls are methods that user-space programs can call in order to ask the Operating System to perform some action on the user-space program's behalf. Software key loggers trap system calls related to keyboard events.

Every time a key is pressed, the trapped system call will record the keystroke to a file for later viewing.

1.6 Remote Dictionary

Tscrack 2.1 is the software to perform this attack. It attempts to discover the administrator's password on a windows computer that has terminal services (or remote desktop) enabled [6]. The attacker supplies it with a dictionary of passwords to try. Most accounts will lock-out after about three incorrect login attempts. However, the administrator's account should never lock-out.

1.7 Root Kit

A Root Kit is like a Trojan and allows an attacker to take control over the computer [7]. It is comparable to a trojan on steroids. Unlike a trojan, a Root kit will alter the operating system in order to hide itself. Root Kit development for windows has been increasing in pace and in complexity.

1.8 Apache

An attacker could open a few connections to an Apache server and force the server to dedicate more and more memory to these connections [8]. Eventually Apache will either crash, or its performance will noticeably degrade.

1.9 War (FTP) Buffer Overflow

Warftpd contains a buffer overflow that leaves the room for 512 bytes of data payload for the ftp command, USER, which is not validated correctly [9]. Metasploit 2.4 is used to remotely overrun the buffer and open a bind shell to remotely control the victim computer.

1.10 Process Table

Fork-bombs have been around for a long time [10]. A process may accidentally (or intentionally) fall into a loop where upon each iteration, a new process is spawned. These new processes take up an enormous amount of resources, and in particular, clog the process-table with hundreds of new entries. Winfb.pl is a file that opens up copies of the windows calculator repeatedly. It ends up opening 101 calculators, which causes a significant load on the computer.

1.11 DoS Massive Amount of Traffic

Trinoo is capable of producing a distributed denial of service attack [11]. The Trinoo master controls an army of Trinoo zombies. An attacker logs into the Trinoo master and issues the command to attack a victim computer. The attack consists of sending massive amounts of network traffic to consume and clog the network bandwidth. Communication between the attacker and the master is done over TCP and is not encrypted. Communication between the Trinoo master and the Trinoo zombies is done through UDP, and is also not encrypted. However, a password is used for authentication.

1.12 Web Browsing

In this norm activity a user browses the World Wide Web (www) using the Internet Explorer and perform a set of pre-defined web search using the Google website. The same sets of websites are searched in each run of the activity to make the norm activity as consistent as possible. However, the user's natural running pace may not be exactly the same from one run to another run.

1.13 Text Editing

In this norm activity a user types a paragraph of text data in the M.S word application continuously for a specific period of the data collection time. The user types in the same text data for each run to make the activity consistent.

2. Procedure for Running Attack and Norm Activities

Three runs are carried out for data collection of each attack. Table 3.2 shows the three runs of data collection. The first run comprises of first keeping the machine idle for 10 minutes and then running the attack for the duration of completing the attack. The second run comprises of keeping the machine idle for 10 minutes, followed by running the Web browsing (norm activity) for 10 minutes and then running the attack along with the Web browsing activity for the duration of completing the attack. The third run comprises of keeping the machine idle for 10 minutes, then running the Text Editing activity (norm activity) for 10 minutes and then running the attack along with the Text editing activity for the duration of completing the attack. Hence, the second and third runs are based on the scenario in which an attack occurs while a user is doing a normal activity on a computer. The first run gives the data for comparing the attack condition with the idle condition and extracting the attack features and characteristics. The second or third run provides the data to compare each of the two norm conditions, Web Browsing and Text Editing, with the idle condition for extracting the norm features and characteristics. The second or third run also provides the data to test if the sensor models produce any false alarms in the condition of a norm activity and how early the sensor models detect an attack from the data mixture of the attack and norm activities.

Table 3.2. Three runs of data collection.

Run	Data collection time in minutes				
	Idle data	Norm activity data		Pure attack data	(Attack & Norm) data
		Web Browsing	Text Editing		
Run1	10 min	-	-	W	
Run2	10 min	10 min	-	-	W
Run3	10 min	-	10 min	-	W

"W" in Table 3.2 is the duration for which each attack takes to complete and it is dependent upon the running time of each attack.

From Table 3.2 we see that a total of 11 (attacks) * 3 (runs) combinations of data collection are carried out with each attack having three sets of data collected, one for each run. Each norm activity is run for every attack to ensure that there is a continuous data stream of norm data followed by the attack & norm data for testing the sensor models. Also the data sets of a norm activity from different runs can be used to ensure that the extracted norm features and characteristics are robust regardless of possibly different paces in different runs of the norm activity.

3. Data Collected

The data in this study is from the data log of activity, state and performance changes collected in attack and norm conditions from a victim computer using the Windows Performance Objects utility. The data log contains on average 1000 to 1200 variables which are related to many computer objects such as Cache, Memory, Network Interface, System etc. An example of performance variables is Process (_Total)\ Page Faults /sec. A page fault occurs when a thread refers to a virtual memory page that is not in its working set in main memory. An example of activity variables is Network Interface packets/sec which indicates the number of packets sent and received through the network interface card. An example of state variables is Memory \Available bytes which measures the amount of memory space available.

4. Summary

Executing attacks and normal use activity allows us to collect the data necessary to investigate intrusion and damage assessment techniques. We have included in this chapter a description of our attack and normal use activities. In the next chapter we present the steps we take to analyze this data and extract what is necessary and sufficient for our investigation.

References

1. NMAP port scan, URL: http://www.insecure.org/nmap/
2. Ettercap, http://ettercap.sourceforge.net
3. Hardware Key Logger, http://www.keykatcher.com/
4. Nessus, http://www.nessus.org/
5. Software Key Logger, http://www.littlesister.de
6. Remote Dictionary, http://web.archive.org/web/20021014015012/
7. Root Kit, http://www.iamaphex.cjb.net/
8. Apache, http://www.apache.org/
9. War(FTP), http://metasploit.com/projects/Framework/exploits.html#warftpd_1 65_user
10. Process Table, http://www.iamaphex.cjb.net/
11. DoS Massive amount of traffic, http://packetstormsecurity.org/distributed/trinoo.tgz

Chapter 4. Mathematical/Statistical Features and Characteristics of Cyber Attack and Norm Data

From the work outlined in previous chapters, we have a set of activity, state and performance variables and their time-series data in different normal and intrusive conditions of system operation. Because these audit variables capture useful information about system activity, state and performance, they capture not only intrusions, but their effects and damages in entire cause-effect chains. Therefore, the use of information from these audit variables allows us to perform intrusion and damage assessment instead of simply intrusion detection. In this chapter, we report our analytical studies of extracting mathematical/statistical features as well as attack and norm data characteristics from the collected data.

1. Introduction

Many features may be useful to discover the distinctive characteristics of attack and norm data, e.g., probability distribution, mean, autocorrelation and wavelet signal strength. This project focuses on the mean and autocorrelation features of cyber attack and norm data for building sensor models of cyber attack detection and identification. Our analysis of a mathematical feature, wavelet signal strength, is supported in the related ARDA-funded effort. The description of the wavelet feature can be found in our final report for the ARDA-funded effort.

To discover the mean and autocorrelation features of the data, statistical analysis, including Mann Whitney U test and autocorrelation, is applied to both attack and norm data. The data are first preprocessed with all variables that contain only zeroes or non-varied entries screened out. The aforementioned Mann Whitney U test and Autocorrelation analysis are implemented to the pre-processed data which are time series data observations. Rank Sum for Mann Whitney U test and Degree of Autocorrelation for Autocorrelation test are obtained. With the selected features for differentiating the attack and norm, the attack and norm data models are constructed. More specifically, for the variables that have a distinction in Mann Whitney result, the statistical mean of the attack or norm data is used as its attack or norm data model. For variables with difference in degree of Autocorrelation between attack data and norm data, the ARIMA time series analysis is used to build an attack or norm data model. Both attack and norm data models are then used in cuscore statistic detection models for cyber attack detection and identification. The following sections provide more details in the discovery of statistics-based data, features and characteristics of attack and norm data. Figure 4.1 shows the process of discovering data, features and characteristics of cyber attack and norm data and building cuscore detection models. This chapter describes mainly the discovery of data, features and characteristics of cyber attack and norm data. Chapter 5 provides the description of building attack and norm data models as well as Cuscore detection models.

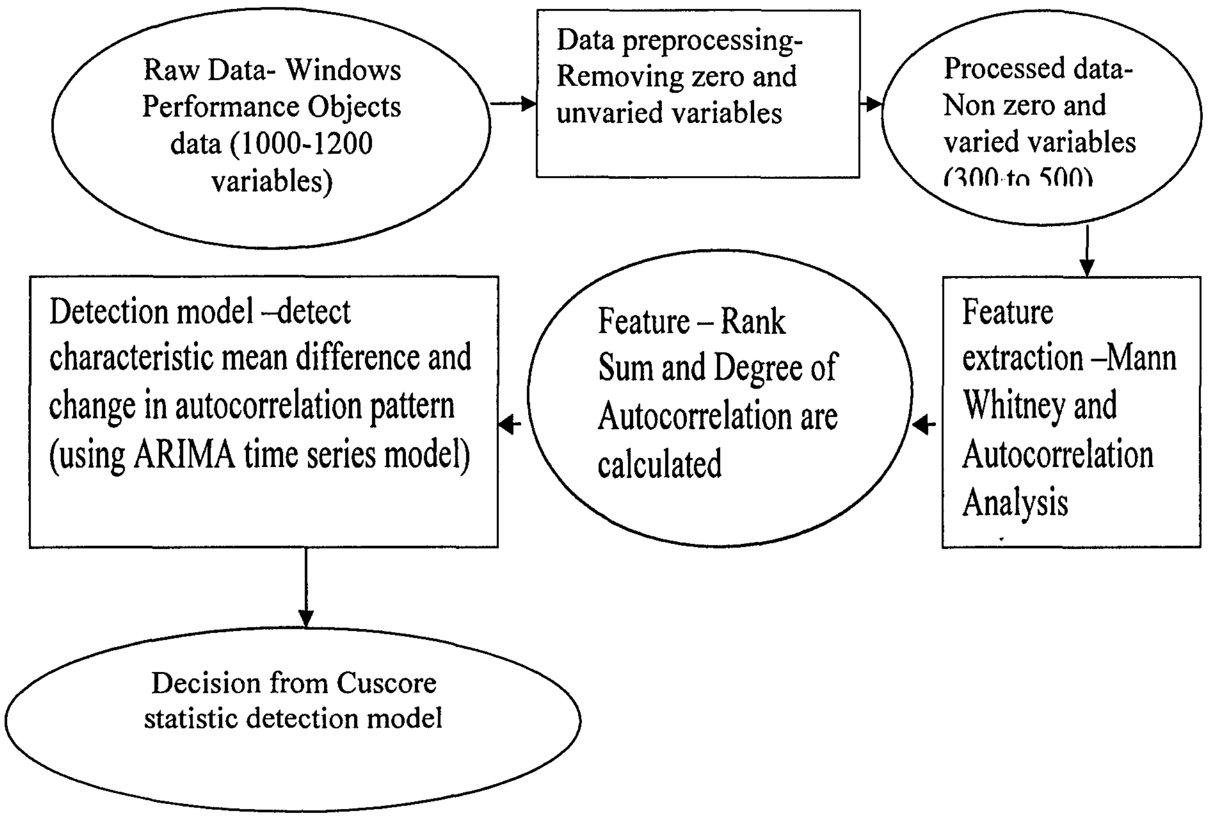

Raw Data- Windows Performance Objects data (1000-1200 variables)
Data preprocessing- Removing zero and unvaried variables
Processed data- Non zero and varied variables (300 to 500)
Feature extraction –Mann Whitney and Autocorrelation Analysis
Feature – Rank Sum and Degree of Autocorrelation are calculated
Detection model –detect characteristic mean difference and change in autocorrelation pattern (using ARIMA time series model)
Decision from Cuscore statistic detection model

Figure 4.1. Process of discovering data, features, characteristics and building detection models for the attack-norm separation approach.

2. Pre-Processing

Two steps are carried out in the pre-processing stage of data analysis. They are 1) screening zeros and 2) screening unvaried variables because these variables cannot be used for finding any useful information about the pattern of data. Both the steps are discussed in detail below.

2.1 Screening Zero Values

Raw data collected from the performance log go through screening so that all the variables with observation values containing only zeros are eliminated from further analysis. In addition, this step removes the first ten observations of each variable to eliminate the unstable warm-up effect at the beginning of data collection.

2.2 Screening Unvaried Variables

The data set after screening out the variables with all zero values is examined to remove any variables that do not show any change of values in the observations. Such variables are not useful for cyber attack detection and are thus removed. The above preprocessing steps are done using statistical software, Statistica. Typically out of the 1000-1200 variables collected only 300-500 variables remain after preprocessing.

3. Statistical Analysis

This section provides the rationale for selecting Mann Whitney U test and Autocorrelation [1] as the statistical analysis to be performed on attack and norm data, and also provides the brief description of each statistical analysis. Descriptive statistics of data are often examined as a part of discovering characteristics of any particular data. Mean and autocorrelation are among the most popular ones.

We use a test of mean difference on data of a variable to detect a change in average with a statistical significance between idle data and attack data to discover attack data characteristic. Mann Whitney U analysis is chosen as a test for difference in mean, since it is distribution-free, and is generally considered at least as powerful as T-Test which is a parametric difference-in-mean test that is based on the assumption of data with the Normal distribution. Steps for Mann Whitney analysis include first sorting idle and attack data into one stream. Then the rank of the data from the same group (idle or attack) are summed together, producing the Rank Sum result. Then the Rank Sum results from two data groups are compared.

Autocorrelation analysis is used to detect a change in data's self-correlation pattern over time. An autocorrelation is a correlation of the data series with itself, lagged by a certain time or number of observations. This statistical analysis allows the data to be viewed in another dimension of its self-correlation structure (in a form of autocorrelation plot) rather than simple time series plots, thus helps in detection of subtle change even when it is not visible by mere graphical analysis. The autocorrelation plots are also an essential tool in determining the appropriate ARIMA time series model, which is used in this study to define attack data model or norm data model.

STATISTICA Version 7.0 is used as the analysis tool to perform both Man Whitney U test and Autocorrelation analysis. For Man Whitney U test, the rank sum, which we use as a statistical feature, from each data group of idle and attack are compared to determine statistical significance of difference.

When data samples are larger than 20, such as almost all of the idle and attack datasets in this studies, the U statistics sampling distribution approaches the Normal distribution, and will be presented along with an Z value from the Normal distribution and the p-value [1].

For a smaller sample size, the exact probability of the particular U statistics is calculated. For any variable with statistical significance in Mann Whitney U test, we use the difference in mean as a data characteristic. If the same change in mean does not appear in the comparison of idle and norm data, we use this as a unique attack characteristic.

For autocorrelation analysis, 1-lag up to 10-lag autocorrelations are analyzed. The autocorrelation coefficients are accompanied by the respective p-values. The total numbers of lags with statistically significant autocorrelation coefficient are used as a degree of autocorrelation of a variable (which we use as attack-norm feature). For example, if all the 10 lags are autocorrelated with statistical significance, then we mark the variable as highly autocorrelated (AH in Tables 4.1-4.3, where A stands for autocorrelation and H stands for high). On the other hand, if none of the 10 lags are significant, the variable is considered not-autocorrelated (AL, where A stands for autocorrelation and L stands for low). For the variable with 1 to 9 significant autocorrelations out of 10 autocorrelations, we use (AI). The changes in the level of autocorrelation between idle data and attack data are then considered a data characteristic. If the same change in autocorrelation does not appear between idle data and norm data, we consider this as a unique attack characteristic.

Tables 4.1-4.3 show examples of characteristics of attack, text editing norm, and web browsing norm respectively. The examples are from 4 variables among over 400 variables that have a change in autocorrelation during the attack and the two norm conditions from the idle condition. In these tables, "A" denotes an autocorrelation characteristic, while "diff()" denotes a difference-in-mean characteristic. For example, In Table 4.1, the variable, "Processor(0)\% C1 Time", has an increase in degree of autocorrelation, (AL, AH), from not-autocrrelated (AL) under the idle condition to highly autocorrelated (AH) under the attack condition. The same variable also has an increase in mean, noted as diff(+), from the idle condition to the attack condition.

Table 4.1. Attack data characteristics for the Ettercap attack

Processor(0)\% C1 Time	AL, AH diff(+)
Processor(0)\% DPC Time	AL, AI
Processor(0)\% Idle Time	AL, AI
Processor(0)\% Processor Time	AL, AI

Table 4.2. Norm data characteristics for the text editing norm.

Processor(0)\% C1 Time	AL, AH diff(-)
Processor(0)\% DPC Time	AL, AI diff(-)
Processor(0)\% Idle Time	AL, AH diff(-)
Processor(0)\% Processor Time	AL, AH diff(+)

Table 4.3. Norm data characteristics for the web browsing norm.

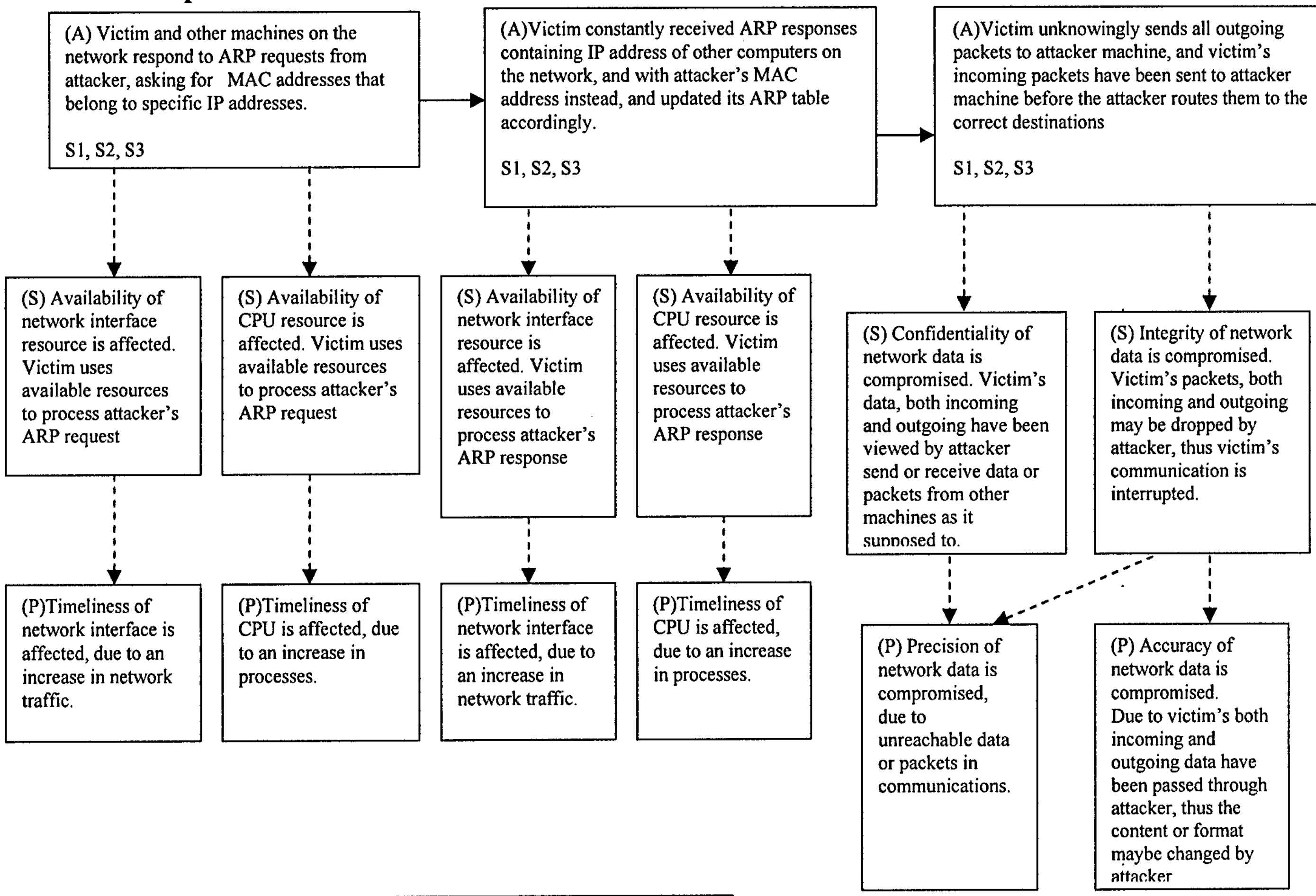
\\ALPHA02-VICTIM\Processor(0)\% C1 Time	AL, AH diff(-)
\\ALPHA02-VICTIM\Processor(0)\% DPC Time	AL, AI
\\ALPHA02-VICTIM\Processor(0)\% Idle Time	AL, AH diff(-)
\\ALPHA02-VICTIM\Processor(0)\% Processor Time	AL, AH diff(+)

From Table 4.1-4.3, we can see that the "diff(+)" characteristic in "Processor(0)\% C1 Time" of the Ettercap attack does not appear in any of norm data characteristics for the same variable, and therefore we discover a unique attack characteristic of the Ethercap attack. Such unique attack characteristics leads to the definition of an attack data model, which is discussed in Chapter 5.

4. An Example: Ettercap Attack Profile and Selected Data Characteristics

An attack profile reveals the cause-effect chain of activities, state changes and performance changes during an attack, as discussed in Chapter 1. Figure 4.2 shows the Etthercap attack profile and the potential detection sensors that can be built based on attack data characteristics which are discovered to be related to various points of the cause-effect chain. These attack data characteristics are listed below.

Ettercap Attack Profile


(A) Victim and other machines on the network respond to ARP requests from attacker, asking for MAC addresses that belong to specific IP addresses. S1, S2, S3
(A)Victim constantly received ARP responses containing IP address of other computers on the network, and with attacker's MAC address instead, and updated its ARP table accordingly. S1, S2, S3
(A)Victim unknowingly sends all outgoing packets to attacker machine, and victim's incoming packets have been sent to attacker machine before the attacker routes them to the correct destinations S1, S2, S3
(S) Availability of network interface resource is affected. Victim uses available resources to process attacker's ARP request
(S) Availability of CPU resource is affected. Victim uses available resources to process attacker's ARP request
(S) Availability of network interface resource is affected. Victim uses available resources to process attacker's ARP response
(S) Availability of CPU resource is affected. Victim uses available resources to process attacker's ARP response
(S) Confidentiality of network data is compromised. Victim's data, both incoming and outgoing have been viewed by attacker send or receive data or packets from other machines as it supposed to.
(S) Integrity of network data is compromised. Victim's packets, both incoming and outgoing may be dropped by attacker, thus victim's communication is interrupted.
(P)Timeliness of network interface is affected, due to an increase in network traffic.
(P)Timeliness of CPU is affected, due to an increase in processes.
(P)Timeliness of network interface is affected, due to an increase in network traffic.
(P)Timeliness of CPU is affected, due to an increase in processes.
(P) Precision of network data is compromised, due to unreachable data or packets in communications.
(P) Accuracy of network data is compromised. Due to victim's both incoming and outgoing data have been passed through attacker, thus the content or format maybe changed by attacker

Sensor number	Variable	Feature and Characteristic
S1	Network Interface\Packets/sec	AL, AI
S2	Network Interface\Packets Received/sec	AL, AI
S3	Network Interface\Bytes Received/sec	AL, AI

A = Activity	⟶ Temporal Link
S = State	- - → Causal Link
P = Performance	

Figure 4.2. Ettercap attack profile.

1. *AL, AI of Network Interface\Packets/sec.* This data variable measures the rate that the packets are sent and received on the network interface. AL, AI indicates that the variable has an increase in autocorrelation from not-autocorrelated to somewhat autocorrelated (2-9 lags of significant autocorrelation).
2. *AL, AI of Network Interface\Packets Received/sec.* This data variable measures the rate that the packets are received on the network interface. AL, AI indicates that the variable has an increase in autocorrelation from not-autocorrelated to somewhat autocorrelated (2-9 lags of significant autocorrelation).
3. *AL, AI of Network Interface\ Bytes Received/sec.* This data variable measures the rate of bytes of the incoming packets on the network interface. AL, AI indicates that the variable has an increase in autocorrelation from not-autocorrelated to somewhat autocorrelated (2-9 lags of significant autocorrelation).

5. Selected Attack Data Characteristics of Eleven Attacks

Table 4.4 gives a list of 11 attacks whose data are collected and analyzed for discovering attack data characteristics. Table 4.5 lists a set of selected variables which reveal attack data characteristics. Table 4.6 provides the summary of selected attack data characteristics which are used to build sensor models for cyber attack detection and identification, which are described in Chapter 5. These variables and corresponding attack data characteristics are selected because they can be well explained in associated attack contexts.

Table 4.4. List of 11 attacks in this study.

Attack Number	Attack Name
1	NMAP
2	Hardware Keylooger
3	EtterCap
4	Nessus
5	Software Keylogger
6	Remote Dictionary
7	Rootkit
8	Apache
9	War(FTP)
10	ProcessTable
11	DOS Massive Amount of Traffic

Table 4.5. List of selected data variables.

Var #	Variable Name
1	TCP\Segments/sec
2	IP\Datagrams/sec
3	IP\Datagrams Received Delivered/sec
4	Network Interface\Packets/sec
5	IP\Datagrams Sent/sec
6	TCP\Connections Passive
7	Processor(0)\DPCs Queued/sec
8	Network Interface\Packets Received/sec
9	Network Interface\Bytes Received/sec
10	Process(_Total)\Page Faults/sec
11	Process(services)\IO Write operations/sec
12	System\File Control Operations/sec
13	System\Context Switches/sec
14	Memory\Page Faults/sec
15	Memory\Committed Bytes
16	Process(war-ftpd)\Page File Bytes
17	Process(war-ftpd)\Working Set
18	LogicalDisk(C:)\Avg. Disk Queue Length
19	Memory\Write Copies/sec
20	Process(_Total)\Private Bytes
21	Processor(_Total)\% DPC Time

Table 4.6. A summary table of selected attack characteristics to be used for building sensor models.

	Attack Number										
Var #	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	AI, AH					AH,AH					
2	AI, AH										
3	AI, AH										
4	AI, AH		AL, AI	AI,AH		AH,AH		AH,AL			AI,AH
5	Diff(+)							AH,AL			
6	Diff(+)					Diff(+)					
7		AH, AI									
8			AL, AI	AI,AH							
9			AL, AI								
10					AH, AI						
11					AL,AH						
12							AL,AH				
13							AL,AH				
14								AH,AH			
15								Diff(+)			
16									Diff(+)		
17									Diff(+)		
18									AH,AL		
19										AH,AI	
20										Diff(+)	
21											AI,AH

6. Summary

Our data analysis allows us to discover the data, features and corresponding characteristics to identify an attack or normal activity. These data variables, statistical features and attack or norm characteristics together define the observables necessary to distinguish between the attacks in our study. This analysis leads us to a set of data, features and characteristics with which we can build individual sensor models to detect each observable.

Reference

1. Siegel, S. (1956). *Nonparametric Statistics for the Behavioral Sciences.* McGraw-Hill Book Company, Inc., New York.

Chapter 5: Sensor Models for Cyber Attack Detection and Identification

This chapter describes mean- and autocorrelation-based cuscore sensor models we develop using the attack-norm separation approach using the discovered attack data characteristics.

1. Cyber Attack Detection Techniques and Models

To compare our attack-norm approach with the two conventional approaches of signature recognition, we select a specific technique from each approach for testing. We select the Artifical Neural Network (ANN) technique to represent the signature recognition approach, the EWMA control chart technique to represent the anomaly detection approach, and Cuscore statistic models to represent the attack-norm separation approach. Hence, for each variable showing an attack data characteristics, an ANN model, an EWMA control chart model and a Cuscore statistic model are built to examine their detection performance in comparison. This section describes these techniques and models.

1.1 Artificial Neural Networks (ANN)

ANN is used as a technique for the conventional approach of signature recognition. ANN is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connectionist approach to computation. In this study the back propagation learning algorithm is used for designing the artificial neural network having one input (user defined which ranges from 0 to maximum observation of the data variable under test), one output (describing the input as attack when close to 1 or norm when close to 0 based on the classification of the trained network) and a hidden layer with 20 units as shown in Figure 5.1.

An ANN in this study computes all the outputs using the sigmoid threshold of the inner product of the corresponding weight and input vectors.

The procedure is as follows. Attack data from Run1 in Table 3.2 and the first half of norm data from Runs 2 and 3 are used for training an ANN for a given variable. The norm data and attack&norm data from runs 2 and 3 in Table 3.2 are used for testing. Training is done by using a statistical tool called the STATISTICA Neural Networks which includes the back propagation algorithm with time-varying learning rate, case-presentation order shuffling and additive noise for robust generalization. The learning rate used for this study is 0.01. STATISTICA Neural Networks automatically retains a copy of the best network discovered, which can be retrieved at the press of a button. When training has finished based on the specified training error value of 0.01, performance against the test data can be checked. Testing is done by running each data observation in the testing data and then determining if the ANN output values is signaled as an attack or norm based on a signal threshold. The signal threshold is set up during the training to minimize the classification error of the ANN.

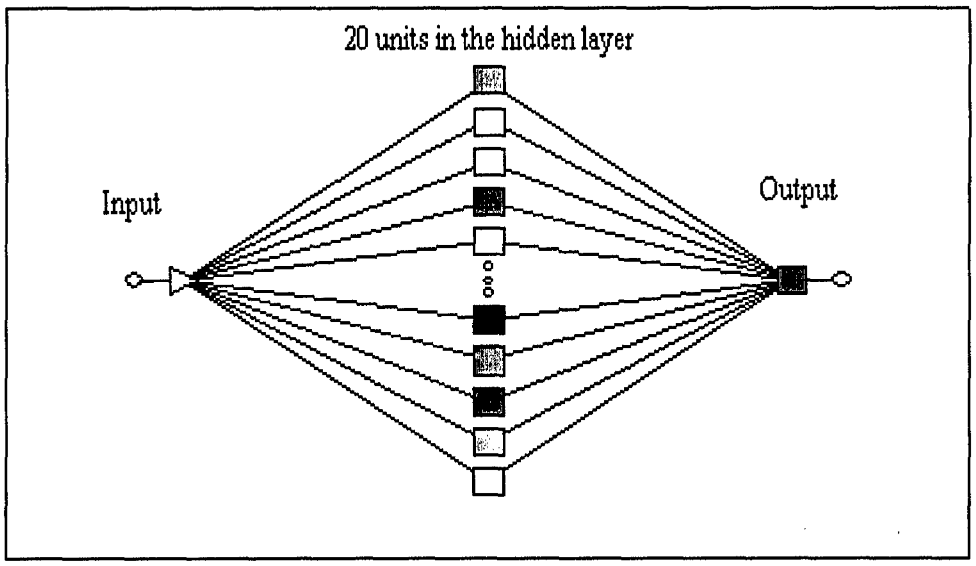


Figure 5.1. ANN hidden layer with 20 units.

1.2 EWMA Control Chart

An EWMA control chart [1-2] is an anomaly detection technique and is typically used when it is desirable to detect out-of-control situations. Many intrusions manifest in dramatic changes in the intensity of events occurring in information systems. Because of the ability of exponentially weighted moving average (EWMA) control charts to monitor the rate of occurrences of events based on the their intensity [1-2], it is considered to be a very good technique to represent the anomaly detection when comparing it with the Cuscore detection model.

The procedure is as follows. In the training phase, z(0) in Equation 1 is initialized to the average of x's (time series data) from the training data. For each *x(i)* in the training data, EWMA [1-2] statistic, *z(i)* , is calculated using the equation given below:

$$z(i) = \lambda x(i) + (1-\lambda)z(i-1) \qquad (5.1)$$

The λ is the parameter to determine the EWMA statistic, *z(i)* is the EWMA statistic for the observation *#i and x(i)* the input data for training.

For each *x(i)* in the training data, the one-step-ahead prediction error *e(i)* is calculated using the equation given below:

$$e(i) = x(i) - z(i-1) \qquad (5.2)$$

and $e^2(0)$ being the average of the sum of squared errors, $e^2(i)$. The data for the training phase is obtained from the first half of the norm data from run 2 and run 3 in the Table 3.2 from the column two indicating the web browsing norm and text editing norm respectively. Data for the testing phase is initialized as *z(0)* is average of *z(i)*'s from the training data. In the testing phase for each *x(i)* in the testing data, EWMA statistic is calculated as:

$$z(i) = \lambda x(i) + (1-\lambda)z(i-1) \qquad (5.3)$$

$\sigma_e^2(i)$ is estimated by calculating a smoothened variance using the equation below:

$$\sigma_e^2(i) = \theta e(i)^2 + (1-\theta)\sigma_e^2(i-1) \tag{5.4}$$

The $\sigma_e^2(i)$ is the square of the estimated standard deviation of *e(i)*. θ is the parameter to represents the smoothened variance of *e(i)*.

UCLx(i) and LCLx(i) are computed as:

$$UCLx(i) = z(i-1) + L\sigma_e(i-1) \text{ –upper control limit} \tag{5.5}$$

$$LCLx(i) = z(i-1) - L - \sigma_e(i-1) \text{ -lower control limit} \tag{5.5}$$

The UCLx(i) is the upper control limit and LCLx(i) is the lower control limit and L is the control limit. If x(i) does not fall in [LCL, UCL], it is detected as an attack. The data for the testing phase is obtained from the second half of the norm data and the attack&norm data from runs 2 and 3 in Table 3.2. The parameters for EWMA are defined in Table 5.1. The parameter L is chosen to be three because it will result in lower false alarm and a larger control limit. θ and λ are 0.3 because previous work by Ye and her colleagues [1-2] show that this combination of values results in the least false alarms and good hit rate. The software used for EWMA is Microsoft excel.

Table 5.1. Parameters of EWMA control charts.

L	Λ	θ
3	0.3	0.3

1.3 Sensor Model Using Cuscore Statistics

Cuscore statistic [3] is used to detect the attack in the presence of the norm activity at the same time based on the attack-norm separation approach. The detection process goes through three steps. First, attack and norm models are defined. Second, the norm cancellation is carried out in the Cuscore statistic model by subtracting the norm model from the testing data (attack and norm mixed) as shown in Equation (5.6):

(5.6)

where *f(t)* is the norm model, *g(t)* is the attack model, and y_t is an observation of the testing data at time *t*. Lastly the residual data after norm data filtering is used to detect the attack by multiplying it with the attack model, *g(t),* as shown in Equation(5.6). The attack data model and the norm data model are first built from the discovered characteristics of attack data and norm data before applying equation 5.6.

The presence of an attack is identified by the substantial slope change of slope in the line of Cuscore values. In this study, the Cuscore before a sharp slope change is used as the threshold value of detecting an attack. An observation is signaled for an attack whenever the Cuscore value for this observation is greater than the threshold value. The details of building Cuscore sensor models are provided in the following sections.

1.3.1 Definition of Attack and Norm Models

To build a norm and attack model for variables with mean difference, we simply use a statistical mean of the testing data as a model. However, for the variables with change in autocorrelation, the best fitted ARIMA time series model [4] needs to be determined.

Before fitting any ARIMA models, data transformation may be done in order to remove any seasonality, trends or outliers. After that, many ARIMA models are fitted to the data for identifying the appropriate ARIMA time series model. The ARIMA model, which produces the lowest Mean Square Error between the original data time series and the time series reconstructed by ARIMA is selected as the most appropriate ARIMA model for the data.

The equation for ARIMA model [4] is given by:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2}. \qquad (5.7)$$

where Y_t is the transformed rating at time t,

ϕ_1 is the portion of previous rating carried over to the rating at time t,

e_t is the shock at time t,

θ_1 is the moving average coefficient.

For example, the normal and attack model for Network Interface\Bytes Received/sec variable in the Ettercap attack are constructed using the ARIMA model, specifically ARMA(1, 2), in Equations (5.8), (5.9), (5.10):

$$Y_t = 0.11398Y_{t-1} + e_t - 0.7570e_{t-1} - 0.6599e_{t-2} \qquad (5.8)$$

The first term, $0.11398Y_{t-1}$, represents the Auto Regressive (AR) part of the model, while last three terms represents the Moving Average (MA) part of the model.

The actual ARIMA equation representing the text editing norm model, which is an ARMA (1, 1), is given below:

$$Y_t = -0.6867Y_{t-1} + e_t - 0.5773e_{t-1} \qquad (5.9)$$

The first term, $-0.6867Y_{t-1}$, represents the Auto Regressive (AR) part of the model, while second and third terms represents the Moving Average (MA) part of the model.

The actual ARIMA equation representing the web browsing norm model, which is an ARMA (1, 1), is given below:

$$Y_t = 0.47883Y_{t-1} + e_t + 0.99185\, e_{t-1} \qquad (5.10)$$

Again, the first term, $0.47883Y_{t-1}$, represents the Auto Regressive (AR) part of the model, while last two terms represents the Moving Average (MA) part of the model.

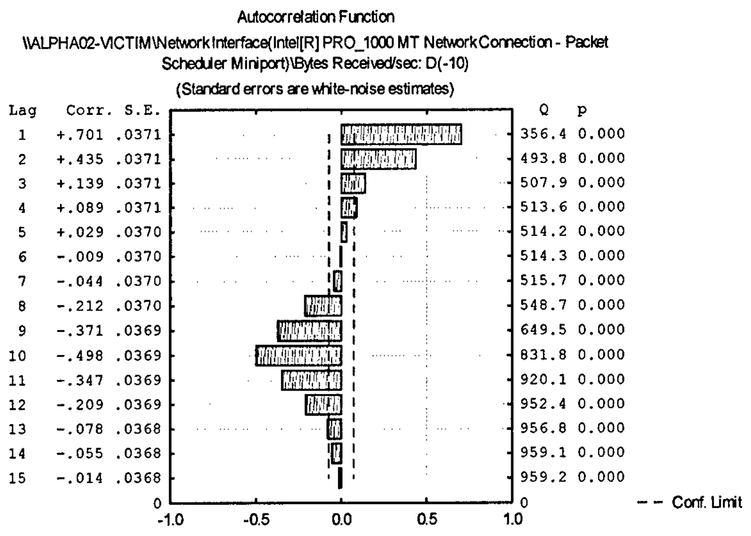
1.3.2 Verification of Attack and Norm Model

To verify the selected attack and norm models for variables with mean difference, the time series data are plotted, and the difference in statistical mean between the two data sets is checked.

However, verifying attack and norm model for variables with autocorrelation change is more complicated. This involves plotting the variable's Autocorrelation Function plot (ACF) and Partial Autocorrelation Function plot (PACF). The selected ARIMA model is then checked with the pattern shown in the two plots to see whether the ACF and PACF plots agree with the chosen model. For example, if the ACF shows a negative spike at lag 1 and PACF shows exponential decay of negative spikes, the appropriate ARIMA model is likely to be an ARIMA (0, 0, 1), with a positive Moving Average (MA) coefficient. If the ACF shows an exponential decay with positive spikes and PACF shows 1 positive spike at lag 1, the appropriate ARIMA model is likely to be ARIMA (1, 0, 0) with a positive Auto Regressive (AR) coefficient.

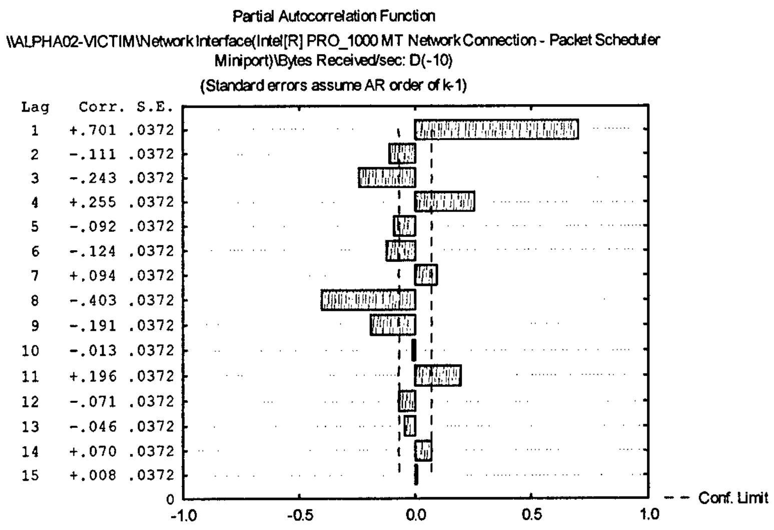
The ACF and PACF plots for Network Interface\Bytes Received/sec variable in the Ettercap attack for attack, text editing norm and web browsing norm are shown in Figures 5.2 to 5.7.

1. Attack model for Network Interface\Bytes Received/sec variable in Ettercap attack:



Lag	Corr.	S.E.	Q	p
1	+.701	.0371	356.4	0.000
2	+.435	.0371	493.8	0.000
3	+.139	.0371	507.9	0.000
4	+.089	.0371	513.6	0.000
5	+.029	.0370	514.2	0.000
6	-.009	.0370	514.3	0.000
7	-.044	.0370	515.7	0.000
8	-.212	.0370	548.7	0.000
9	-.371	.0369	649.5	0.000
10	-.498	.0369	831.8	0.000
11	-.347	.0369	920.1	0.000
12	-.209	.0369	952.4	0.000
13	-.078	.0368	956.8	0.000
14	-.055	.0368	959.1	0.000
15	-.014	.0368	959.2	0.000
0			0	

Figure 5.2. Autocorrelation Function plot (ACF) for the transformed Network Interface\Bytes Received/sec under the attack condition.



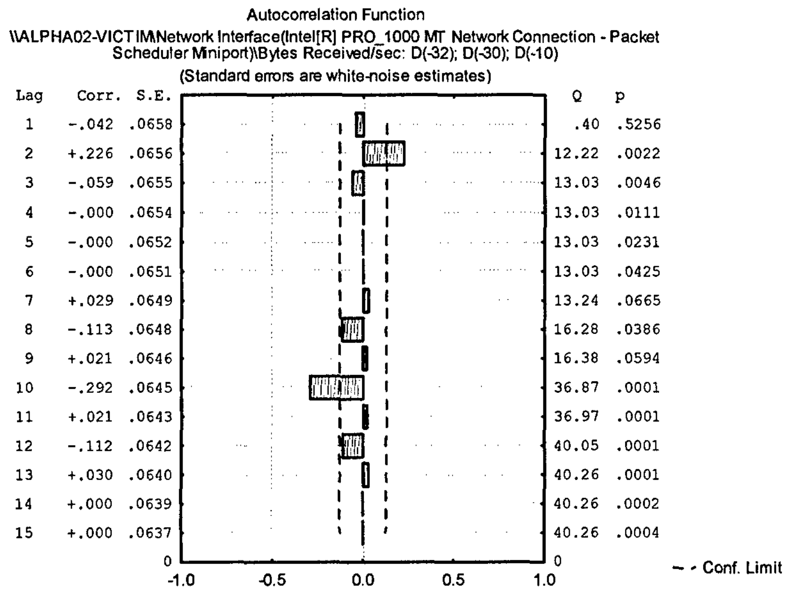
Lag	Corr.	S.E.
1	+.701	.0372
2	-.111	.0372
3	-.243	.0372
4	+.255	.0372
5	-.092	.0372
6	-.124	.0372
7	+.094	.0372
8	-.403	.0372
9	-.191	.0372
10	-.013	.0372
11	+.196	.0372
12	-.071	.0372
13	-.046	.0372
14	+.070	.0372
15	+.008	.0372
0		

Figure 5.3 Partial Autocorrelation Function plot (PACF) for the transformed Network Interface\Bytes Received/sec under the attack condition.

In Figure 5.2, ACF shows an exponential decay of positive spikes, while in Figure 5.3 PACF shows an oscillating decay of positive and negative spikes. This is agreeable to the

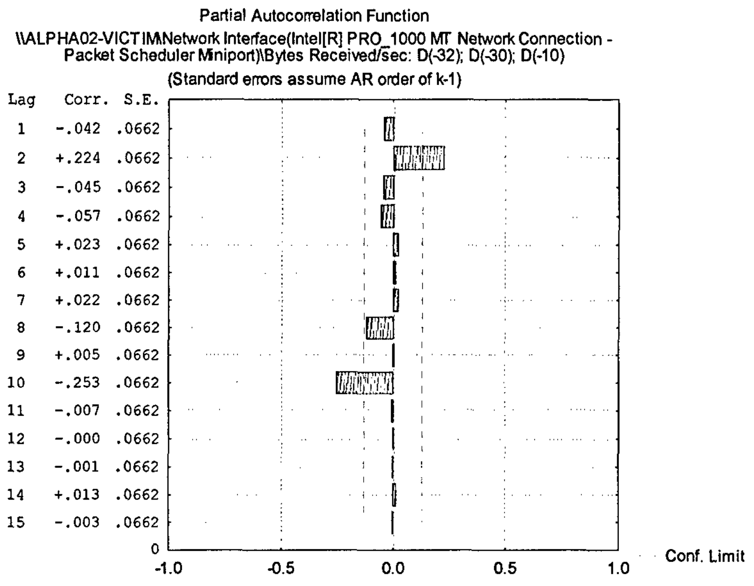
ARMA(1, 2) model with a positive Autoregressive (AR) coefficient and negative Moving Average (MA) coefficients.

2. Text editing norm model for Network Interface\Bytes Received/sec variable in the Ettercap attack:



Lag	Corr.	S.E.	Q	p
1	-.042	.0658	.40	.5256
2	+.226	.0656	12.22	.0022
3	-.059	.0655	13.03	.0046
4	-.000	.0654	13.03	.0111
5	-.000	.0652	13.03	.0231
6	-.000	.0651	13.03	.0425
7	+.029	.0649	13.24	.0665
8	-.113	.0648	16.28	.0386
9	+.021	.0646	16.38	.0594
10	-.292	.0645	36.87	.0001
11	+.021	.0643	36.97	.0001
12	-.112	.0642	40.05	.0001
13	+.030	.0640	40.26	.0001
14	+.000	.0639	40.26	.0002
15	+.000	.0637	40.26	.0004

Figure 5.4. Autocorrelation Function plot (ACF) for the transformed Network Interface\Bytes Received/sec under the text editing norm condition.

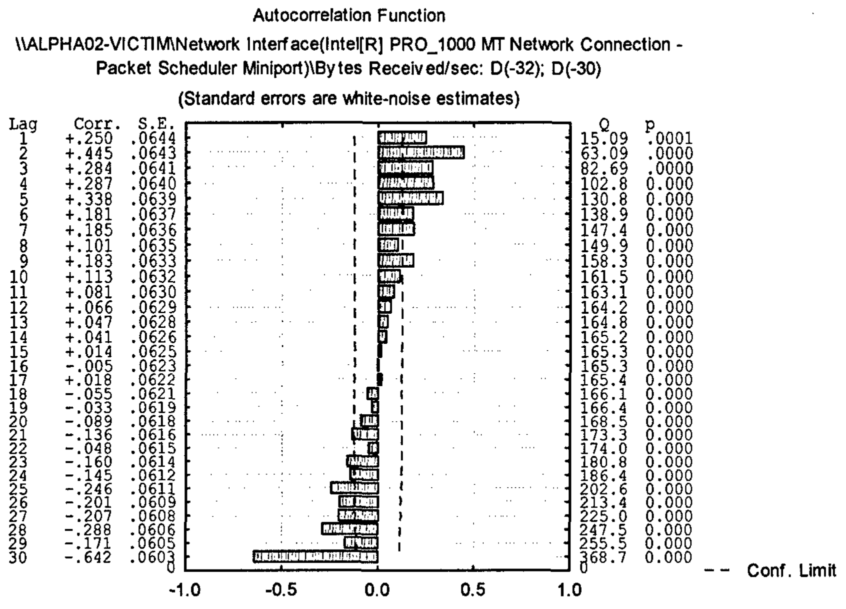


Lag	Corr.	S.E.
1	-.042	.0662
2	+.224	.0662
3	-.045	.0662
4	-.057	.0662
5	+.023	.0662
6	+.011	.0662
7	+.022	.0662
8	-.120	.0662
9	+.005	.0662
10	-.253	.0662
11	-.007	.0662
12	-.000	.0662
13	-.001	.0662
14	+.013	.0662
15	-.003	.0662

Figure 5.5 Partial Autocorrelation Function plot (PACF) for the transformed Network Interface\Bytes Received/sec under the text editing norm condition.

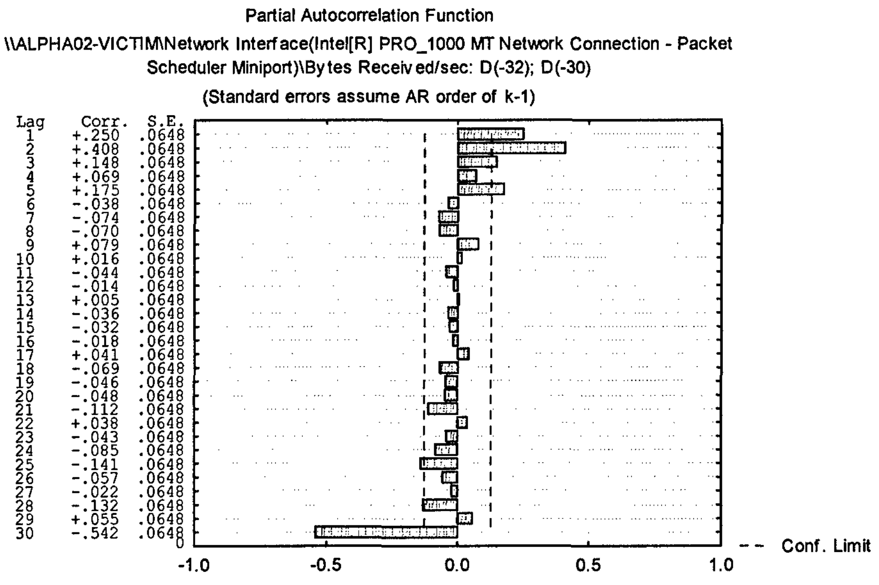
In Figure 5.4 ACF shows an oscillating decay of negative and positive spikes, and in Figure 5.5 PACF also shows the same oscillating pattern. This confirms the ARMA (1, 1) model with negative coefficients for both Auto Regressive (AR) and Moving Average (MA) parts is appropriate for the data.

3. Web browsing norm model for Network Interface\Bytes Received/sec variable in Ettercap attack:



Lag	Corr.	S.E.	Q	p
1	+.250	.0644	15.09	.0001
2	+.445	.0643	63.09	.0000
3	+.284	.0641	82.69	.0000
4	+.287	.0640	102.8	0.000
5	+.338	.0639	130.8	0.000
6	+.181	.0637	138.9	0.000
7	+.185	.0636	147.4	0.000
8	+.101	.0635	149.9	0.000
9	+.183	.0633	158.3	0.000
10	+.113	.0632	161.5	0.000
11	+.081	.0630	163.1	0.000
12	+.066	.0629	164.2	0.000
13	+.047	.0628	164.8	0.000
14	+.041	.0626	165.2	0.000
15	+.014	.0625	165.3	0.000
16	-.005	.0623	165.3	0.000
17	+.018	.0622	165.4	0.000
18	-.055	.0621	166.1	0.000
19	-.033	.0619	166.4	0.000
20	-.089	.0618	168.5	0.000
21	-.136	.0616	173.3	0.000
22	-.048	.0615	174.0	0.000
23	-.160	.0614	180.8	0.000
24	-.145	.0612	186.4	0.000
25	-.246	.0611	202.6	0.000
26	-.201	.0609	213.4	0.000
27	-.207	.0608	225.0	0.000
28	-.288	.0606	247.5	0.000
29	-.171	.0605	255.5	0.000
30	-.642	.0603	368.7	0.000

Figure 5.6 Autocorrelation Function plot (ACF) for the transformed Network Interface\Bytes Received/sec under the web browsing norm condition.



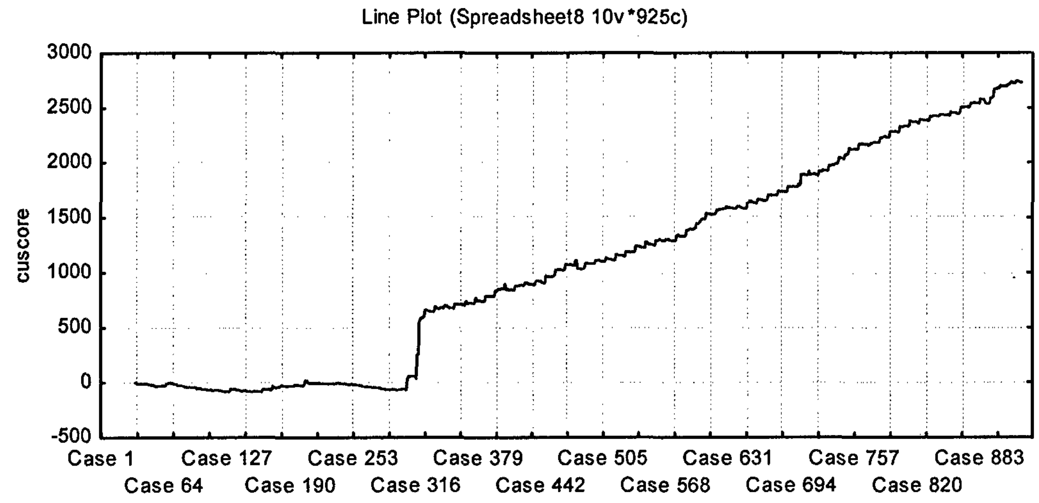
Lag	Corr.	S.E.
1	+.250	.0648
2	+.408	.0648
3	+.148	.0648
4	+.069	.0648
5	+.175	.0648
6	-.038	.0648
7	-.074	.0648
8	-.070	.0648
9	+.079	.0648
10	+.016	.0648
11	-.044	.0648
12	-.014	.0648
13	+.005	.0648
14	-.036	.0648
15	-.032	.0648
16	-.018	.0648
17	+.041	.0648
18	-.069	.0648
19	-.046	.0648
20	-.048	.0648
21	-.112	.0648
22	+.038	.0648
23	-.043	.0648
24	-.085	.0648
25	-.141	.0648
26	-.057	.0648
27	-.022	.0648
28	-.132	.0648
29	+.055	.0648
30	-.542	.0648

Figure 5.7 Partial Autocorrelation Function plot (PACF) for the transformed Network Interface\Bytes Received/sec under the web browsing norm condition.

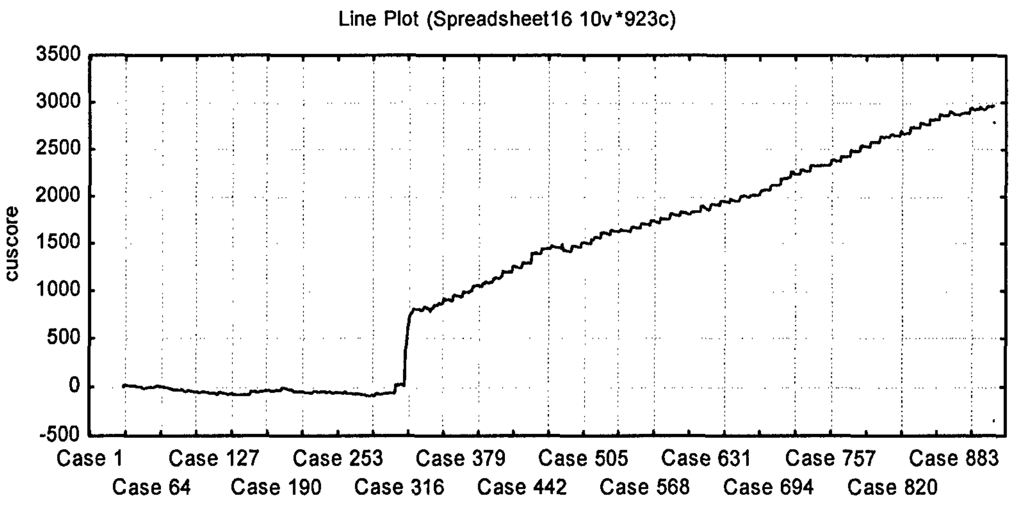
In Figure 5.6 ACF shows an exponential decay of positive spikes, while in Figure 5.7 PACF also shows an exponential decay of positive spikes. This is reasonable for the ARMA (1, 1) model with positive coefficients for both Auto Regressive (AR) and Moving Average (MA) parts.

1.3.3 Sensor Model Using Cuscore Statistics

After verification, the attack and norm models are implemented into an Cuscore statistic model to detect the attack occurring in the same time as normal user activity. The following figures show Cuscore statistic charts for the Ettercap attack with normal activity data as the background noise data. The first 300 data observations are the norm data. Starting from the 301st observation, the data are mixed with both attack and normal activities.

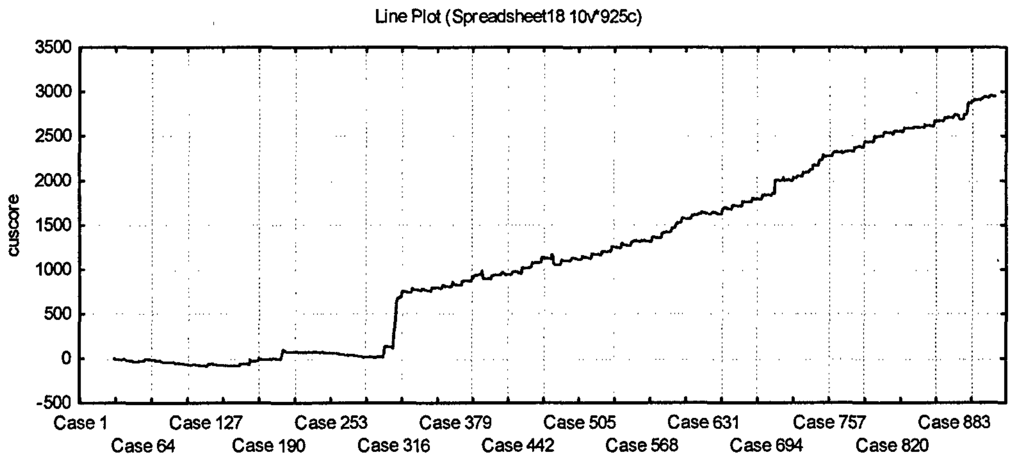


a) Web browsing norm and the Ettercap attack (1st detection at observation 307)

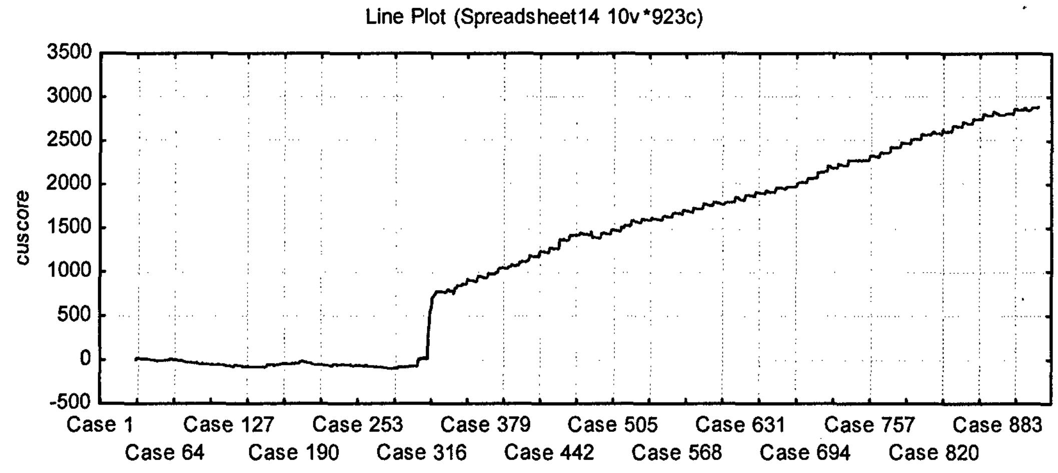


b) Text editing norm and the Ettercap attack (1st detection at observation 314)

Figure 5.8. Cuscore charts for the variable, Network Interface\Packets Received/sec.

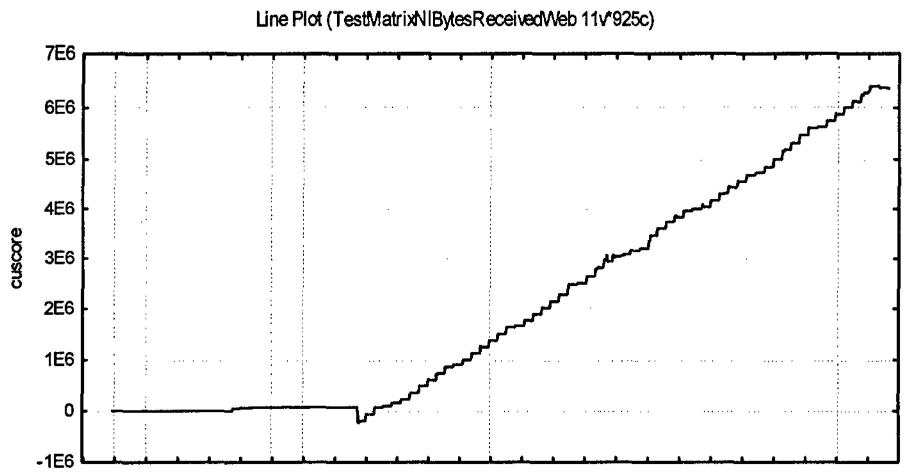


a) Web browsing norm and the Ettercap attack (1st detection at observation 306)

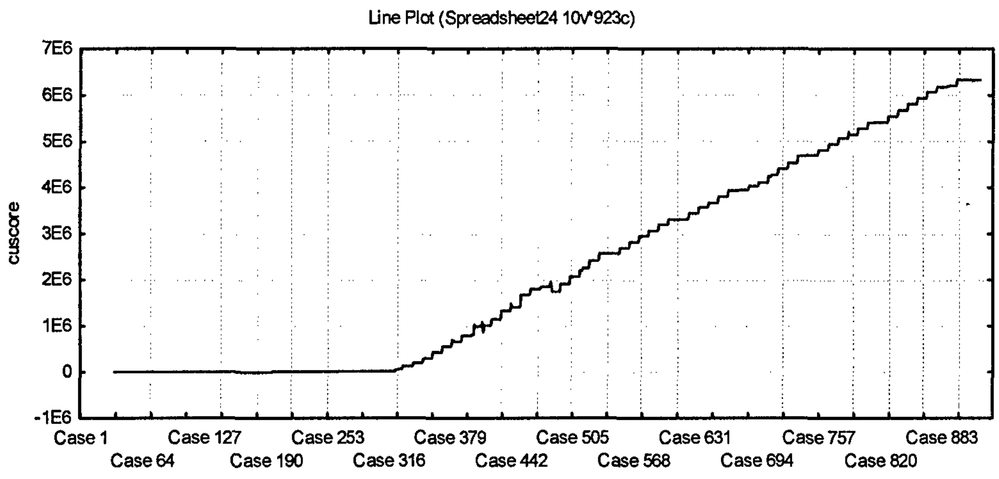


b) Text editing norm and the Ettercap attack (1st detection at observation 318)

Figure 5.9. Cuscore charts for the variable, Network Interface\ Packets/sec.



a) Web browsing norm and the Ettercap attack (1st detection at observation 306)



b) Text editing norm and Ettercap attack (1st detection at observation 307)

Figure 5.10. Cuscore charts for the variable, Network Interface\ Bytes Received/sec.

2. Performance of ANN, EWMA and Cuscore Models

In this section we give the performance comparison of our Cuscore sensor models with ANN models and EWMA control chart models for the selected data variables described in Chapter 4 for each of 11 attacks. A total of 11 attacks, each in combination with 2 norm activities, are tested and the results are summarized using two performance measures: false alarms and the first signal. False alarms compute the number of observations in the norm data (the first 300 observations of the testing data) that are signaled an attack by a model. The first signal is the first data observation in the attack&norm data when an attack is detected.

Tables 5.2-5.23 give the performance results of our autocorrelation- and mean-based Cuscore models, ANN models and EWMA models in comparison. The false alarm ratio is shown as the number of observations out of the first 300 observations in the testing data, which are the norm data. For example, in Table 5.2 for the variable IP\ Datagrams/sec, the false alarm ratio of the Cuscore model is 0/300. This means that the Cuscore model for this variable produces zero false alarms in the first 300 observations of the testing data.

The first signal column in the result tables shows the observation number of the first observation which is signaled as attack over the total number of observations in the mixed attack&norm data. The first signal is a measure of earliness in detection. For example, in Table 5.2 for the variable, IP\ Datagrams/sec, the first signal ratio of the Cuscore model is 2/218. This means that the Cuscore model detects an attack at the 2nd observation from the attack starting point, and the total number of the attack&norm data observations is 218.

Table 5.2. Performance results for the test condition: NMAP with text editing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	First Signal	False Alarms	First Signal	False Alarms	First Signal
1	IP/Datagrams/sec	0/300	2/218	4/300	3/218	1/300	4/218
2	IP/Datagrams Received Delivered/sec	0/300 2/300	4/218 1/218	7/300	1/218	3/300	1/218
3	Network Interface/Packets/sec	0/300	3/218	3/300	4/218	2/300	5/218
4	TCP/Segments/sec	0/300	2/218	0/300	1/218	1/300	1/218
5	IP/Datagrams Sent/sec	0/300	1/218	3/300	4/218	3/300	2/218
6	TCP\Connections Passive	0/300	42/218	0/300	42/218	0/300	42/218

In Table 5.2 for most variables, the Cuscore detection models have zero false alarm and the better first signal performance than ANN and EWMA. For the variables with the same first signal, the Cuscore models produce fewer or same false alarms than ANN and EWMA.

Table 5.3. Performance results for the test condition: NMAP with web browsing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	First Signal	False Alarms	First Signal	False Alarms	First Signal
1	IP/Datagrams/sec	0/300	3/215	6/300	5/215	3/300	4/215
2	IP/Datagrams Received Delivered/sec	0/300	2/215	7/300	2/215	2/300	4/215
3	Network Interface/Packets/sec	0/300 2/300	3/215 1/215	2/300	2/215	2/300	2/215
4	TCP/Segments/sec	0/300	2/215	4/300	3/215	3/300	4/215
5	IP/Datagrams Sent/sec	0/300 2/300	3/215 2/215	2/300	2/215	2/300	2/215
6	TCP\Connections Passive	0/300	8/215	0/300	8/215	0/300	8/215

In Table 5.3 for most variables, the Cuscore detection model has zero false alarm and athe better first signal performance than ANN and EWMA. For the variables with the same first signal, the Cuscore models produce fewer or same false alarms than ANN and EWMA. For the variables with the same false alarm rate, the Cuscore models have the better first signal performance than ANN and EWMA.

Table 5.4. Performance results for the test condition: Ettercap with text editing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Network Interface Packets/sec	0/300	18/623	37/300	16/623	13/300	12/623
2	Network Interface Packets Received/sec	0/300 3/300	14/623 12/623	37/300	16/623	13/300	12/623
3	Network Interface Bytes Received/sec	0/300	7/623	36/300	16/623	14/300	12/623

In Table 5.4 for one variable, the Cuscore model has zero false alarm and the better first signal detection than ANN and EWMA. For the variable with the same first signal, the Cuscore model presents much fewer false alarms than ANN and EWMA.

Table 5.5. Performance results for the test condition: Ettercap with web browsing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	First Signal	False Alarms	First Signal	False Alarms	First Signal
1	Network Interface Packets/sec	0/300	6/627	3/300	27/627	14/300	6/627
2	Network Interface Packets Received/sec	0/300	7/627	3/300	27/627	12/300	12/627
3	Network Interface Bytes Received/sec	0/300	6/627	3/300	27/627	15/300	6/627

In Table 5.5 for one variable, the Cuscore model has zero false alarm and the better first signal performance than ANN and EWMA. For the variables with the same first signal, the Cuscore models produce zero false alarms.

Table 5.6. Performance results for the test condition: Hardware key logger with text editing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Processor(0)\DPCs Queued/sec	0/300	74/614	218/300	>614	37/300	112/614

In Table 5.6 the false alarms of ANN and EWMA are much higher than those of the Cuscore models. Moreover, ANN cannot detect the attack for Processor(0)\DPCs Queued/sec variable. The first signal, ">614", means that there is no attack signal identified on all 614 observations under the attack&norm condition.

Table 5.7. Performance results for the test condition: Hardware key logger with web browsing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Processor(0)\DPCs Queued/sec	0/300	14/614	231/300	>667	21/300	127/667

In Table 5.7 the false alarms of ANN and EWMA are much higher than those of the Cuscore models. Moreover, ANN cannot detect the attack for Processor(0)\DPCs Queued/sec. The first signal, ">677", means that there is no attack signal identified on all 677 observations under the attack&norm condition.

Table 5.8. Performance results for the test condition: Nessus with text editing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	First Signal	False Alarms	First Signal	False Alarms	First Signal
1	Network Interface/Packets/sec	0/300	4/431	98/300	21/431	21/300	11/431
2	Network Interface/Packets Received/sec	0/300	4/431	98/300	21/431	21/300	11/431

In Table 5.8 ANN models produce high false alarms. The Cuscore models have zero false alarm and the better first signal performance than ANN and EWMA.

Table 5.9. Performance results for the test condition: Nessus with web browsing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Network Interface/Packets/sec	0/300	7/437	140/300	22/437	34/300	9/437
2	Network Interface/Packets Received/sec	0/300	7/437	140/300	22/437	34/300	9/437

In Table 5.9 ANN models produce very high false alarms. The Cuscore models have zero false alarm and the better first signal performance than ANN and EWMA.

Table 5.10. Performance results for the test condition: Software key logger with text editing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Process(services)\IO Write Operations/sec	0/300	13/634	4/300	17/364	9/300	13/634
2	Process(_Total)\Page Faults/sec	0/300	3/634	9/300	13/634	14/300	3/634

In Table 5.10 for both variables the Cuscore models have zero false alarm, which is less than those of ANN and EWMA. The first signal performance of the Cuscore models is comparable to that of EWMA, but is better than that of ANN.

Table 5.11. Performance results for the test condition: Software key logger with web browsing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Process(services)\IO Write Operations/sec	0/300	13/631	2/300	13/631	0/300	13/631
2	Process(_Total)\Page Faults/sec	0/300	3/631	18/300	4/631	20/300	3/631

In Table 5.11 for both variables the Cuscore models have zero false alarm, which is less than those of ANN and EWMA. The first signal performance is relatively good for all three models.

Table 5.12. Performance results for the test condition: Remote dictionary with text editing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Network Interface/Packets/sec	0/300	1/270	3/300	2/270	5/300	2/270
2	TCP\Segments/sec	0/300	3/270	3/300	2/270	0/300	3/270
3	TCP\Connections Passive	0/300	3/270	0/300	3/270	0/300	3/270

In Table 5.12 the false alarm rates of the Cuscore models are lower than those of ANN and EWMA in some cases. The first signal performance of the Cuscore models is relatively good for all three models.

Table 5.13. Performance results for the test condition: Remote dictionary with web browsing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Network Interface/Packets/sec	0/300	3/270	2/300	8/270	2/300	4/270
2	TCP\Segments/sec	0/300	2/270	2/300	8/270	11/300	2/270
3	TCP\Connections Passive	0/300	2/270	0/300	2/270	0/300	2/270

In Table 5.13 the false alarm rates of the Cuscore models are lower than those of ANN and EWMA in most cases. The first signal performance of the Cuscore models, which is relatively good for all three models, is comparable to that of EWMA, but is mostly better than that of ANN.

Table 5.14. Performance results for the test condition: Rootkit with text editing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	First Signal	False Alarms	First Signal	False Alarms	First Signal
1	System\File Control Operations/sec	0/300	22/599	23/300	22/599	17/300	22/599
2	System\Context Switches/sec	0/300	1/599	14/300	22/599	9/300	1/599

In Table 5.14 for both variables the Cuscore models have zero false alarm, which is less than those of ANN and EWMA. The first signal performance of the Cuscore models is comparable to that of EWMA and ANN.

Table 5.15. Performance results for the test condition: Rootkit with web browsing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	First Signal	False Alarms	First Signal	False Alarms	First Signal
1	System\File Control Operations/sec	0/300	3/623	22/300	3/623	17/300	3/623
2	System\Context Switches/sec	0/300	3/623	24/300	3/623	18/300	3/623

In Table 5.15 for both variables the Cuscore models have zero false alarm, which is less than those of ANN and EWMA. The first signal performance of the Cuscore models is comparable to that of EWMA and ANN.

Table 5.16. Performance results for the test condition: Apache with text editing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Memory\Page Faults/sec	0/300	1/120	1/300	1/120	3/300	1/120
2	Network Interface\Packets/sec	0/300	1/120	13/300	1/120	14/300	1/120
3	IP\Datagrams Sent/sec	0/300	1/120	13/300	1/120	14/300	1/120
4	Memory\Committed Bytes	0/300	1/120	12/300	1/120	12/300	1/120

In Table 5.16 for all variables the Cuscore models have zero false alarms, which is less than those of ANN and EWMA. The first signal performance is relatively good for all three models.

Table 5.17. Performance results for the test condition: Apache with web browsing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	First Signal	False Alarms	First Signal	False Alarms	First Signal
1	Memory\Page Faults/sec	0/300	1/122	0/300	1/122	1/300	1/122
2	Network Interface\Packets/sec	0/300	1/122	15/300	1/122	16/300	1/122
3	IP\Datagrams Sent/sec	0/300	1/122	15/300	1/122	16/300	1/122
4	Memory\Committed Bytes	0/300	1/122	20/300	1/122	18/300	1/122

In Table 5.17 for all variables the Cuscore models have zero false alarm, which is less than those of ANN and EWMA. The first signal performance is relatively good for all three models.

Table 5.18. Performance results for the test condition: Warftpd with text editing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	First Signal	False Alarms	First Signal	False Alarms	First Signal
1	Process(war-ftpd)\Page File Bytes	0/300	5/6	2/300	5/6	1/300	5/6
2	Process(war-ftpd)\Working Set	0/300	5/6	1/300	5/6	0/300	5/6
3	LogicalDisk(C:)\Avg. Disk Queue Length	0/300	1/6	17/300	>6	12/300	>6

In Table 5.18 for all variables the Cuscore models have zero false alarm, which is less than those of ANN and EWMA. The first signal performance for the variables in the Process group is comparable to that of EWMA, and ANN. However, both ANN and EWMA cannot detect the attack for LogicalDisk(C:)\Avg. Disk Queue Length. The first signal ">6" means that there is no attack signal identified on all 6 observations under the attack&norm condition.

Table 5.19. Performance results for the test condition: Warftpd with web browsing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Process(war-ftpd)\Page File Bytes	0/300	3/6	1/300	3/6	1/300	3/6
2	Process(war-ftpd)\Working Set	0/300	2/6	1/300	2/6	1/300	2/6
3	LogicalDisk(C:)\Avg. Disk Queue Length	0/300	1/6	3/300	4/6	1/300	>6

In Table 5.19 for all variables the Cuscore models have zero false alarm, which is less than those of ANN and EWMA. The first signal performance for the variables in the Process group is comparable to that of EWMA, and ANN. For LogicalDisk(C:)\Avg. Disk Queue Length, the Cuscore model has the better first signal performance than ANN and EWMA which cannot detect the attack in all attack&norm observations. The first signal ">6" means that there is no attack signal identified on all 6 observations under the attack&norm condition.

Table 5.20. Performance results for the test condition: Process Table with text editing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Process(_Total)\Private Bytes	0/300	1/13	23/300	2/16	20/300	2/13
2	Memory\Write Copies/sec	0/300	2/13	8/300	2/16	5/300	2/13

In Table 5.20 for Process(_Total)\Private Bytes, the Cuscore model has zero false alarm, which is considerably lower than those of ANN and EWMA, and the better first signal performance. For Memory\Write Copies/sec variable with the same first signal as ANN and EWMA, the Cuscore model has zero false alarm which is less than those of ANN and EWMA.

Table 5.21. Performance results for the test condition: Process Table with web browsing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Process(_Total)\Private Bytes	0/300	1/16	19/300	5/16	14/300	4/16
2	Memory\Write Copies/sec	0/300	4/16	7/300	4/16	3/300	4/16

In Table 5.21 for Process(_Total)\Private Bytes, the Cuscore model has zero false alarm which is considerably lower than those of ANN and EWMA, and the better first signal performance. For Memory\Write Copies/sec variable with the same first signal as ANN and EWMA, the Cuscore model has zero false alarm which is lower than those of ANN and EWMA.

Table 5.22. Performance results for the test condition: Trinoo with text editing.

Sensor #	**Variable**	**Cuscore Sensor Model**		**ANN**		**EWMA Control Chart**	
		False Alarms	**First Signal**	**False Alarms**	**First Signal**	**False Alarms**	**First Signal**
1	Network Interface/Packets/sec	0/300	1/600	2/300	2/600	0/300	2/600
2	Processor(_Total)\% DPC Time	0/300	1/600	23/300	1/600	18/300	1/600

In Table 5.22 for Network Interface/Packets/sec, the Cuscore model has zero false alarm which is considerably lower that those of ANN and EWMA, and the better first signal performance. For Processor(_Total)\% DPC Time variable with the same first signal as ANN and EWMA, the Cuscore model still presents zero false alarm which lower than those of ANN and EWMA.

Table 5.23. Performance results for the test condition: Trinoo with web browsing.

Sensor #	Variable	Cuscore Sensor Model		ANN		EWMA Control Chart	
		False Alarms	First Signal	False Alarms	First Signal	False Alarms	First Signal
1	Network Interface/Packets/sec	0/300	1/600	2/300	2/600	2/300	1/600
2	Processor(_Total)\% DPC Time	0/300	2/600	19/300	3/600	17/300	2/600

In Table 5.23 Cuscore models show the better or same first detection performance and lower false alarm rates than ANN and EWMA.

3. Summary

In this chapter we provide example Cuscore models developed using the attack-norm separation approach. This work on audit data, features, characteristics, and detection models establishes a coherent set of scientific theories and analytical techniques that enable the automatic extraction and identification of audit data, features and characteristics for intrusion and damage assessment.

References

1. N. Ye, Q. Chen, and C. Borror, "EWMA forecast of normal system activity for computer intrusion detection," IEEE Transactions on Reliability, Vol. 53, No. 4, 2004, pp. 557-566.

2. N. Ye, C. Borror, and Y. Zhang, "EWMA techniques for computer intrusion detection through anomalous changes in event intensity," Quality and Reliability Engineering International, Vol. 18, No. 6, 2002, pp. 443-451.

3. George Box and Albert Luceno, "Statistical Control by monitoring and Feedback Adjustment," Wiley publishers, NY, 1997.

4. Yaffee, R. (2000) *Introduction to Time Series Analysis and Forecasting* (San Diego, California: Academic Press).