AC 2007-173: USING DATA MINING TO DETECT INTRUSIONS IN COMPUTER NETWORKS

Mario Garcia, Texas A&M University-Corpus Christi

Using Data Mining to Detect Intrusions in Computer Networks

Abstract

In recent years Data mining techniques have been applied in many different fields including marketing, manufacturing, process control, fraud detection and network management. Over the past several years a growing number of research projects have applied data mining to various problems in intrusion detection. The goal of this research is to design and implement an anomaly detector using data mining. The project includes the use of open source tools and/or modifications to existing tools to incorporate the goals of collection, filtering, storage, archival, and attack detection in a cohesive software system. This paper also surveys a representative cross section of these research efforts. Conclusions are drawn and directions for future research are suggested.

Introduction

Intrusion detection is the process of monitoring and analyzing the events occurring in a computer system in order to detect signs of security problems[3]. The importance of Intrusion Detection Systems (IDS) has grown tremendously recently because of our dependence on electronic forms of data. Sensitive information, which has to be kept secure, is kept in an electronic form on computers. Military and the Government have been the most vocal of the IDS users, but more and more private organizations are realizing the importance of such a system.

Current IDS are tuned to detect known attacks. Enough data exists or could be collected to allow network administrators to detect policy violations. Unfortunately, the data is so voluminous, and the analysis process so time consuming, that the administrators have problems analyzing the data. Data Mining adds additional depth to the administrator's defenses, and allows them to more accurately determine what the threats against their network are. Hence, activity that it is not detected in near real-time in an online NID, can now be identified. Some examples of attacks that mining could detect, that online NIDS cannot detect, include certain types of malicious activity, such as low and slow scans, a slowly propagating worm, unusual activity of a user based on some new pattern of activity. Ideally, such a system should be able to derive a threat level for the network activity that it analyzes, and predict future attacks based on past activity.

Designing and implementing an anomaly detector is the major goal of this project. Data mining has become a very useful technique to reduce information overload and improve decision making by extracting and refining useful knowledge through a process of searching for relationships and patterns from the extensive data collected by organizations. The extracted information is used to predict, classify, model, and summarize the data being mined. In recent years data mining techniques have been successfully used for intrusion detection[9].

Data Mining, KDD and Related fields

The term knowledge discovery in databases (KDD) is used to denote the process of extracting useful knowledge from large data sets. Data mining, by contrast, refers to one particular step in this process. Generally, data mining is the process of extracting useful and previously unnoticed models or patterns from large data stores [3,6, 7,10,14]. Specifically, the data mining step applies so-called data mining techniques to extract patterns from the data. Data mining is a component of the Knowledge Discovery in Databases (KDD) process[7]. It is preceded and followed by other KDD steps, which ensure that the extracted patterns actually correspond to useful knowledge. Indeed, without these additional KDD steps, there is a high risk of finding meaningless or uninteresting patterns. In other words, the KDD process uses data mining techniques along with any required pre- and post-processing to extract high-level knowledge from low-level data. In practice, the KDD process is interactive and iterative, involving numerous steps with many decisions being made by the user.

Data mining is focused in finding relatively simple models in an efficient and scalable manner. It emphasizes the efficient discovery of simple, but understandable models that can be interpreted as interesting or useful knowledge. Data mining is just a step in the KDD process. As such, it has to contribute to the overall goal of knowledge discovery. Data mining techniques essentially are pattern discovery algorithms. Some techniques such as association rules[1] are unique to data mining, but most are drawn from related fields such as machine learning or pattern recognition. Typically, the best features of these techniques and detection models are combined to obtain a high detection performance and a complex profile for intruders. Recognized practices merge models for new activity (attacks or normal events) and the existing models, to generate adaptive processes able to learn inductively the existing correlations: this Meta-Learning capability and its adaptability with other techniques have been evaluated empirically as effective and scalable[14]. The rules reduce substantially the impractical manual development process of patterns and profiles, computing statistical patterns from the collected data.

Misuse Detection:

In Misuse Detection each data record is classified and labeled as normal or anomalous activity. This process is the basis for a learning algorithm able to detect known attacks and new ones if they are cataloged appropriately under a statistical process. The basic step known as discovery outliers, matches abnormal behavior against attack patterns knowledge base that capture behavioral patterns of intrusion and typical activity. To do this, it is needed to compute each measure with random variables implying more updating effort as more audit records are analyzed. Although the activity needs to be analyzed individually, complementary visualization and data mining techniques can be used to improve performance and reduce the computational requirements. Some projects using this concept are JAM (Java Agents for Meta-learning), MADAM ID (Mining Audit Data for Automated Models for Intrusion Detection) and Automated Discovery and Concise Predictive Rules for Intrusion Detection. JAM (developed at Columbia University), uses data mining techniques to discover patterns of intrusions[2]. It then applies a meta-learning classifier to learn the signature of attacks.

Researchers at Iowa State University report on Automated Discovery of Concise Predictive Rules for Intrusion Detection[2]. This system performs data mining to provide global or temporal views of intrusions on a distributed system. The rules detect intrusions against privileged programs (such as Sendmail) using feature vectors to describe the system calls executed by each process. A genetic algorithm selects feature subsets to reduce the number of observed features while maintaining or improving learning accuracy.

Anomaly detection, on the other hand, uses a model of normal user or system behavior and flags significant deviations from this model as potentially malicious. This model of normal user or system behavior is commonly known as the user or system profile. Strength of anomaly detection is its ability to detect previously unknown attacks. The most popular anomaly detection system using data mining is ADAM (Audit Data Analysis and Meaning). One of the most significant advantages of ADAM is the ability to detect novel attacks, without depending on attack training data, through a novel application of the pseudo-Bayes estimator[4].

In this project, the focus is on implementing an off-line "Anomaly Detection Intrusion Detection Systems" (AD-IDS) to periodically analyze or audit batches of TCP/IP network log data. While offline processing would seem to be solely a compromise between efficiency and timeliness, it provides for some unique functionality. For instance, periodic batch processing allows the related results (such as activity from the same source) to be grouped together, and all of the activity can be ranked in the report by the relative threat levels. Another feature unique to the offline environment is the ability to transfer logs from remote sites to a central site for correlation during off-peak times. Offline processing also permits to more easily overcome shortcomings in real-time IDSs. For example, many IDSs will start to drop packets when they are flooded with data at a rate that is faster than the information they can process. Other forms of denial of service can also be launched against an active (real-time) IDS system, such as flooding it with fragmented IP packets. This attack will consume time and memory attempting to reconstruct bogus traffic. Meanwhile, the attacker can break into the real target system without fear of detection. The off-line environment is significantly less vulnerable to these attacks, especially if they are given a high degree of assurance that any traffic admitted to the local network is logged (such as by the firewall responsible for the admittance of such traffic). In principle, an AD-IDS "learns" what constitutes "normal" network traffic, developing sets of models that are updated over time. These models are then applied against new traffic, and traffic that doesn't match the model of "normal" is flagged as suspicious. It is important to note that no software can monitor all network traffic because the data processing becomes prohibitive.

Building Intrusion Detection Models Using Data Mining

The first step is to train the anomaly detector using training data. Once the systems learns, it classifies the data as normal or anomalous. Given a training set, which is a set of labeled sequences, and a test set, which is another set of labeled sequences unseen by the system, it classifies each of the test sequences as either 'Normal' or 'Anomalous'. For any given anomaly detector, its performance is measured by calculating its hit ratio, miss ratio and the false alarm ratio.

TCP/IP traffic data is generated using a data collection application, which is referred to as a **collector**. A collector can be a "home grown" application or purchased from one of many network performance vendors available in the market. When network data is generated it is stored in files commonly known as network data logs. These data logs can be represented in a wide variety of formats, such as ASCII, binary, or RDBMS tables. Pre-processing of the network data within the logs is usually required so that the data can be standardized. This will ensure that the data is processed correctly and that there are no discrepancies in the representation of the data.

Network audit data was collected under an isolated network and on a real network using Snort 2.4.3. Snort[13] is a cross-platform, lightweight network intrusion detection tool that can be deployed to monitor TCP/IP packets and detect a wide variety of suspicious network traffic as well as outright attacks. Snort provides administrators with enough data to make informed decisions on the proper course of action in the face of suspicious activity. Snort can perform protocol analysis and content searching/matching. It can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and more. Snort uses a flexible rules language to describe traffic that it should collect or pass, and includes a detection engine utilizing a modular plug-in architecture. Snort has real-time alerting capability as well, incorporating alerting mechanisms for Syslog, user- specified files, a UNIX socket, or WinPopup messages to Windows clients using Samba's smbclient. Snort has three primary uses. It can be used as a packet sniffer like tcpdump or as a packet logger that is useful for network traffic debugging. It can also be used as a network intrusion detection system. Snort logs packets in either tcpdump binary format or in Snort's decoded ASCII format. Plug-ins allow the detection and reporting subsystems to be extended. Available plug-ins include database logging, small fragment detection, portscan detection, and HTTP URI normalization. [Snort]

Analysis Tools

To solve the complex issues involved in turning TCP/IP network transactions into data suitable for mining and exception reporting the tools *tcpreplay* and Bro-IDS were utilized. Tcpreplay is a suite of utilities for *NIX systems for editing network traffic collected using tcpdump and ethereal. Tcpreplay provides the ability to classify traffic as client or server, edit packets at layers 2-4, and replay the traffic at arbitrary speeds onto a network for sniffing or through a device. Since the study of Internet traffic requires working with large quantities of data, selecting an appropriate tool for data analysis is crucial. This project utilizes the Bro intrusion detection system. Bro is an open-source, Unix-based Network Intrusion Detection System (NIDS) that passively monitors network traffic and looks for suspicious traffic. Bro detects intrusions by comparing network traffic against a customizable set of rules describing events that are deemed troublesome. These rules might describe specific attacks (including those defined by "signatures") or unusual activities (e.g., certain hosts connecting to certain services or patterns of failed connection attempts). Bro uses a specialized policy language that allows a site to tailor Bro's operation, both as site policies evolve and as new attacks are discovered. If Bro detects something of interest, it can be instructed to either generate a log entry, alert the operator in real-time, or initiate the execution of an operating system command (e.g., to terminate a connection or block a malicious host on-the-fly). In addition, Bro's detailed log files can be particularly useful for forensics. Bro targets high-speed (Gbps), high-volume intrusion detection. By using packet-filtering techniques, Bro is able to achieve the necessary performance while running on commercially available PC hardware, and thus can serve as a cost-effective means of monitoring a site's Internet connection. Although its primary purpose is for detection of network intrusions, Bro comes with powerful scripting capabilities that make analyzing large volumes of data more manageable.[5]

System Design

Data Mining is a resource consuming computing process. An efficient deployment for an Intrusion Detection system employing Data Mining requires an adaptive and scalable *architecture* and *infrastructure* able to support the storage for the audited data, its processing, the model generation and distribution, as well as the interaction with the pre-existing elements in the organizational security infrastructure. Data collected by Snort is transformed into connection records by Bro-ids that are used to generate models using the Data Mining tool WEKA. This study uses data collected on an isolated network and at the main router connected to the Internet. Data is collected using a Snort sensor that runs on a dual-processor Dell server, with 2GB of RAM and 140GB of disk space. The operating system on this machine is Red Hat Enterprise Linux 4. This system has minimal number of packages installed, sufficient for a usable system with the unessential services turned off. By hardening the OS and further securing the system, it will be ideal to act as a Snort[13] sensor. Snort ver. 2.4.3 is installed on this system along with Apache, SSL, PHP, MySQL, and BASE.

Since a Snort sensor is fundamentally passive i.e. it receives data but does not send any, it makes sense security wise to run it in a stealthy mode. Two network interface card's are used, one for management and the other for sniffing. The NIC used for sniffing in stealthy mode is not given any IP address whereas the NIC used for management is provided with an IP address. The NIC with the IP address is connected to a network different from the sniffing interface for administrative purposes. Attack free data is collected on an isolated network and on the main router for data with attacks by running Snort in Packet Logging mode. The packets are logged to a single log file in tcpdump format using log_tcpdump module. Log_tcpdump logs packets in the tcpdump file format. There are a wide assortment of applications and tools designed to read tcpdump output.

Normal data without any intrusions and attacks was collected on an isolated simulated network which was set up solely for this purpose. This network consisted of five Dell Pentium III machines, with 256 MB RAM and 20 GB of disk space. This network consisted of a Windows 2003 server, three Windows XP machines and a Linux machine. The Windows 2003 server runs the IIS, FTP, SMTP and other services. Samba smb client was turned on the Linux machine to communicate with the Windows network. All the computers were connected to a 3Com® Baseline Dual Speed 16-Port hub with CAT5 cables. The computer running Snort was connected to this hub in a stealthy mode. An 8-port KVM switch was utilized to control multiple computers from a single keyboard, video monitor and mouse.

Network traffic is generated on the isolated network using the open source network traffic generator "Traffic". Traffic is a network traffic generator following a server/client model for generating high volumes of traffic on a network. The server module is run on the Windows 2003 server and the client module is run on two Windows XP machines. The client permits the user to choose the protocol, number of packets and the time interval between the packets. Data was collected on the isolated network over a period of one week by running the Network traffic generator “Traffic”. The Traffic client was installed on two Windows XP machines. Snort was used in packet logging mode to collect the data. The data passing through the 3Com hub is captured by Snort.

The internal network of the organization where the data was collected is connected to the Internet through a single high speed internet connection. Snort was deployed in such a way that it can monitor all the traffic coming (and leaving) an isolated network. One way to accomplish this is to deploy a passive tap with minimal effect on network operations. The sensor is deployed on this tap between the external firewall and the internal network such that it can monitor all the traffic that enters (and departs) over that connection. This allows the sensor to examine all of the data associated with the external link so that it can be effectively used to monitor incoming (and outgoing) attacks. The snort machine is located at the main router on campus, which is connected to the Internet by a 100Mb/s full-duplex Ethernet link. Data was collected on the network tap over a period of one week by running Snort in a stealthy packet logging mode.

The goal of the analysis is to create descriptive information from the raw TCPDUMP files, then to mine the data in order to determine likely intrusive TCP/IP connections. The raw data consists of packet level transmission data including source and destination IP address and ports; flags, acknowledgements and packet sequence numbers; and window, buffer and optional information. In the audit data no single record represents a complete conversation between two IP addresses. In fact, each record is only a portion of the conversation: the source sending information to the destination, or vice versa. In order to create useful inputs for data mining, we must first determine which records are parts of the same conversation. After the conversation reconciliation, we can compute meaningful variables such as the number of connections made to a one or more destination IP addresses within a two-second time window.

For data mining the data preparation goal is to establish the final state of a conversation and to understand the behavior of each source IP address on the network in relation to destination IP addresses and destination ports. Information about the number of destination IP addresses, and time differences to destination address fields by the source IP address are collected. This information provides for each source IP address, counts of the number of times each final state occurred, how many IP addresses were connections made, and counts of the time difference groupings. Information is also collected for destination port types and time differences to destination ports. This information provides, for each unique IP source to destination IP connection, the number of times each specific action was attempted, and the number of times port hits occurred within the specified intervals. To get information on what action the user was attempting, the destination ports to specific user actions were mapped. The destination port determines the function the user is trying to access. By transforming port numbers into action types, it is possible to determine basic user information. For this purpose, a high level groupings of the port types as login, email, system status check, SNMP, date, who, chat, and other were created.

In order to consider automated versus manual input stream, the amount of time elapsed between connections to different destination IP addresses from a single IP source address is needed. Indicators were created for measuring the elapsed time within specific ranges. For example, it was recorded if the elapsed time was less than 5 seconds, between 5 and 30 seconds, greater than

30 seconds, or undeterminable. This information creates the time difference to destination address and time difference to destination port fields. Tcpreplay, installed on the Snort sensor, was used to replay the data collected by Snort. A network was set up using the 3Com hub with just the Snort sensor and the Bro-ids machine. The network packets broadcasted by tcpreplay on to the hub were captured by the Bro-ids and transformed into the Network connection data suitable for data mining.

Bro is an intrusion detection system that works by passively watching traffic seen on a network link. It is built around an *event engine* that pieces network packets into events that reflect different types of activity. Some events are quite low-level, such as the monitor seeing a connection attempt; some are specific to a particular network protocol, such as an FTP request or reply; and some reflect fairly high-level notions, such as a user that was successfully authenticated during a login session. Bro runs the events produced by the event engine through a *policy script* supplied to it by the administrator. Bro scripts are made up of *event handlers* that specify what to do whenever a given event occurs. Event handlers can maintain and update global state information, write arbitrary information to disk files, generate new events, call functions (either user-defined or predefined), generate *alerts* that produce *syslog* messages, and invoke arbitrary shell commands[5.]

Bro performs the generic connection analysis like the connection start time, duration, hosts etc. using the conn analyzer. The Connection record data type associated with the conn analyzer keeps track of the state associated with each connection in the form of one line connection summaries. A connection is defined by an initial packet that attempts to set up a session and all subsequent packets that take part in the session. Initial packets that fail to set up a session are also recorded as connections and are tagged with a failure state that designates the reason for failure. Each entry contains the following data describing the connection: date/time, the duration of the connection, the local and remote ip addresses and ports, bytes transferred in each direction, the transport protocol (udp, tcp), the final state of the connection, and other information describing the connection. The conn analyzer also has a list of callable connection functions that help in the generic connection analysis. The data was extracted from the tcpdump data by making changes to the conn analyzer. The feature list employed was obtained from Darpa feature set. The main output of *conn* analyser is a one-line ASCII summary of each connection. These summaries are written to a file with the name *conn.tag.log*, where *tag* uniquely identifies the Bro session generating the logs

Mining Data for Complex Relationships

Classification involves data that is divided into two or more groups, or classes to predict the category of data by building a model based on some predictor variables. Classification learning is sometimes called Supervised because this method operates under supervision by being provided with the actual outcome for each of the training examples. This outcome is called the class of the example. The Classification algorithm is inductively learned to construct a model from the pre-classified data set. Each data item is defined by values of the attributes. Classification may be viewed as mapping from a set of attributes to a particular class. The decision tree classifies data using the values of its attributes. The decision tree is initially constructed from a set of pre-classified data. The main approach is to select the attributes, which best divides the data items into their classes. According to the values of these attributes the data items are partitioned. This process is recursively applied to each partitioned subset of the data items. The process terminates when all the data items in current subset belongs to the same class. A node of a decision tree specifies an attribute by which the data is to be partitioned. Each node has a number of edges, which are labeled according to a possible value of the attribute in the parent node. An edge connects either two nodes or a node and a leaf. Leaves are labeled with a decision value for categorization of the data. The success of classification learning can be judged by trying out the concept description that is learned on an independent set of test data[17.]

WEKA[17] has implementations of numerous classification and prediction algorithms. The basic ideas behind using all of these are similar. The dataset in this experiment is classified by using the J4.8 algorithm that is Weka's implementation of C4.5 decision tree learner. Since J4.8 algorithm can handle numeric attributes, there is no need to discretize any of the attributes. The experiment was carried out using the normal data as the training data and a subset of attack data as the test data initially and subsequently using attack data as both training and test data. Weka cannot handle large datasets since it loads everything into memory before it processes the data. To overcome this shortcoming, 5,900 records were generated randomly from the normal data and two sets of 6,600 and 3,400 records were generated randomly from the attack data. The subset of attack data with 3,400 records was used as test data in both cases. The results of using J4.8 decision tree on the data are presented in Table 1.

Normal data as training data and attack data as test data: Here the normal data (attack-free subset) was used as training data and attack data (3400 records) was used as testing data.
Correctly Classified: 622 (18.2 %) Incorrectly Classified: 2778 (81.7 %)

Attack Data as both training and testing data: In this set up the subset of attack data with 6600 records was used as training data and the subset with 3400 records was used as testing data.
Correctly Classified: 3298 (97 %) Incorrectly Classified: 102 (3%)

Results Analysis:

By comparing the percentage of correctly classified instances it is clear that using attack data (97%) as the training set is way more efficient than using the attack free data (18.2%) as the training set.

Table 1 Summary of classification testing

	Training Time	Testing Time	Accuracy
Normal Data as Trainer	0.07 sec	0.01sec	18.2%
Attack Data as Trainer	4.08 sec	3.98sec	97%

Clustering

Cluster analysis can be defined as "a wide variety of procedures that can be used to create a classification. These procedures empirically form "clusters" or groups of highly similar entities."

Cluster analysis defines groups of cases through a number of procedures that are similar. Clustering algorithms segment the data into groups of records, or clusters that have similar characteristics. Clustering discovers complex intrusions that occur over an extended periods of time and at different places, correlating independent network events. The sets of data belonging to the cluster (attack or normal activity profile) are modeled according to pre-defined metrics and their common features. Clustering is shown to provide high performance but is supposedly computationally expensive. Weka's SimpleKMeans algorithm was used for clustering the data. The WEKA SimpleKMeans algorithm uses Euclidean distance measure to compute distances between instances and clusters. SimpleKMeans clusters data using *k*-means; the number of clusters is specified by a parameter.

Results Analysis:

Comparing the percentage of the number of clustered instances it is clearly evident that using attack data (20%) for training is more efficient than the normal data (9%). An association rule identifies a combination of attribute values that occur with greater frequency than it might be expected if the values were independent of one-another. Association rules like classification rules attempts to predict the class given a set of conditions on the Left Hand Side (LHS.) The conditions are typically attribute value pairs, also referred to as item-sets. However, the main difference is that the prediction associated with the Right Hand Side (RHS) of the rule is not confined to a single class attribute, instead it can be associated with one or more attribute combinations. Weka's *Apriori* starts with a minimum support of 100% of the data items and decreases this in steps of 5% until there are at least 10 rules with the required minimum confidence of 0.9 or until the support has reached a lower bound of 10% which ever occurs first. There are four metrics for ranking rules: *Confidence, Lift, Leverage* and *Conviction.*

Testing and validation

Evaluating detection systems is a difficult undertaking, complicated by several common practices. For example, most evaluations are done according to a black-box testing régime. While black-box testing can demonstrate the overall performance capabilities of a detection system, it reveals almost nothing about the performance of components inside the black box, such as how phenomena affecting the components (e.g., a feature extractor or an anomaly detector) or the interactions among them will influence detection performance. If the performance aspects of components like anomaly detectors are not fully understood, then the performance aspects of any system composed of such elements cannot be understood either. Testing was carried out by using different attacks on the system trained with real time data using J48, Weka's version of decision tree algorithm C4.5. The results are presented in table 2.

Using one type of Attack (Mailbomb):
    Correctly Classified Instances: 41
    Incorrectly Classified Instances: 0

Using two types of attacks (Mailbomb and Neptune):
    Correctly Classified Instances:160
    Incorrectly Classified Instances: 0

Using three types of attacks (Mailbomb, Neptune and Smurf):

Correctly Classified: 236 (99.5%)

Incorrectly Classified: 1(0.4219 %)

Table 2 Summary of the testing:

Test	Time taken	Accuracy (%)
1. Using one type of attack	1.8	100
2. Using two types of attacks	1.59	100
3. Using three types of attacks	1.61	99.57

Conclusion

The integration of data mining in Intrusion Detection is in an interesting phase of development where data mining has made it possible to obtain performance improvements over current commercial products, with special features as:

- Better feature extraction and meaningful information from large amount of raw data, getting more accurate models from applying DM techniques and correlation algorithms. These practices reduce the analysis overload to human operators and the associated issues.
- Potential for predictive analysis of suspicious activity, enabling defensive action before a severe injury occurs or the system is totally compromised.
- The data mining adaptability for specific environments allows recognizing individual trends more efficiently making appropriate statistical recognition of new or hidden malicious activity.

The complexity involved in setting up this kind of system provides new challenges for implementing scalable and incremental deployments. The goal is to ensure accuracy with efficient time and resource investments specifically improving data selection, data preparation, data quality (from all the information sources), resulting in precision and reduction in the associated computing costs

Bibliography

1. Agarwal,R.,Imielinski,T.,and Swami,A. Mining Associations between Sets of Items in Massive Databases. *In proceedings of the ACM-SIGMOD 1993 International Conference on Management of Data*, pages 207-216.
2. Allen ,J.,Christie,A.,Fithen,W.,McHugh,J.,Pickel,J.,and Stoner, E. State of the Practice of Intrusion Detection Technologies. Technical report, Carnegie Mellon University.
3. Gurley B. R. *Intrusion Detection*. Macmillan Computer Publishing. (MCP), Indianopolis. 2000.
4. Barbara D. and Jajodia S. *Applications of Data Mining in Computer Security.* Kluwer Academic Publishers, Norrwell, MA. 2002.
5. bro-ids.org. Home page. Available from www.bro-ids.org(Visited on Jan 24, 2006).
6. Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. (1996c) From data mining to knowledge discovery: an overview. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 1-34. AAAI Press / MIT Press, Menlo Park, CA.
7. Goebel M. and Gruenwald L.: A Survey of Data Mining and knowledge Discovery Software Tools. SIGKDD

Explorations 1(1): 20-33 (1999)
8. Witten I and Eibe F. Data Mining Practical Machine Learning Tools and Techniques. Morgan Kauffman Publishers, San Francisco ,CA ,2005.
9. Wenke L. Applying Data Mining to Intrusion Detection: The Quest for Automation, Efficiency, and Credibility. *SIGKDD Explorations*, 4(2), December 2002.
10. Mannila H., Local and Global Methods in Data Mining: Basic Techniques and Open Problems, proceedings of 29th International Colloquium on Automata, Languages and Programming, Lecture Notes on Computer Science, pages 57-68, Springer-Verlag, 2002
11. Traffic. Homepage. Available from http://robert.rsa3.com/traffic.html
12. Peddabachigari, Ajith A. and Johnson T. Intrusion Detection Systems Using Decision Trees and Support Vector Machines. *Information Management Journal* v.4: 635-660, 2001.
13. Snort.org. Home page. Available from www.snort.org (Visited on Jan15, 2005)
14. Wenke L. and Stolfo S.. Data Mining Approaches for Intrusion Detection. *Proceedings of the Seventh USENIX Security Symposium (SECURITY '98)*, San Antonio, TX, January 1998.
15. TCPreplay. Homepage. Available from http://tcpreplay.synfin.net/
16. Weka. Home page. Available from http://www.cs.waikato.ac.nz/ml/weka/
17. Junxin Z, Wenke L, Sal S, Phil C, Eskin E, Fan W, Miller M, and Hershkop S. Real Time Data Mining-based Intrusion Detection. In *Proceedings of the 2001 DARPA Information Survivability Conference and Exposition (DISCEX II)* (selected for presentation), Anaheim, CA, June 2001.