ABSTRACT

DETECTION OF DENIAL OF SERVICE ATTACKS USING DATABASE QUERIES

by
Dmytro Zakhalyavko

In the current intrusion detection world, most intrusion detection systems output data into flat files. This project was conducted in order to improve intrusion detection data and alerts by writing them into a database system and analyzing them with SQL. A database plug-in was developed that helps to transition the data from an intrusion detection system to a database. Storing, analyzing, categorizing, and accessing data are major advantages and reasons for using databases in intrusion detection. Security analysts have to constantly perform the difficult task of sorting through a haystack of attack alerts, many of which turn out to be inaccurate. It is possible to make the job of managing these alerts, analyzing data with high precision, and searching for attacks or intrusions easier by using SQL based analysis. In addition, a statistical analysis was conducted and proved that such a method can be effective in detecting intrusions and increasing the security of the network.

DETECTION OF DENIAL OF SERVICE ATTACKS USING DATABASE QUERIES

by

Dmytro Zakhalyavko

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Department of Electrical and Computer Engineering

August 2004

APPROVAL PAGE

DETECTION OF DENIAL OF SERVICE ATTACKS USING DATABASE QUERIES

Dmytro Zakhalyavko

Dr. Constantine Manikopoulos, Thesis Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Sotirios G. Ziavras, Committee Member — Date
Professor of Electrical and Computer Engineering and Computer Science, NJIT

Dr. George Antoniou, Committee Member — Date
Professor, Montclair State University

BIOGRAPHICAL SKETCH

Author:	Dmytro Zakhalyavko
Degree:	Master of Science
Date:	August 2004

Undergraduate and Graduate Education:

- Master of Science in Computer Engineering,
  New Jersey Institute of Technology, 2004

- Bachelor of Science in Computer Engineering,
  New Jersey Institute of Technology, 2002

Major:	Computer Engineering

This work is dedicated to

my parents who stimulated me in

my education and throughout my life

and

teachers who

taught me everything I know in my profession.

ACKNOWLEDGMENT

I want to express deep gratitude to my thesis advisor, Dr. Constantine Manikopoulos, for his ideas, encouragement and guidance in all phases of my studies at NJIT.

Sincere thanks to Dr. Sotirios G. Ziavras and Dr. George Antoniou for serving as members of the thesis committee.

Special thanks to my parents and everybody who were with me through everything, never let me down, and encouraged me.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

There are multiple ways to analyze [1,2], store and retrieve intrusion detection data. The most logical one would be to store it inside of a database. This approach also introduces a convenient and easy way of detecting attacks through writing SQL queries. A recent survey [3] by CERT/CC (Computer Emergency Response Team/Coordination Center) showed that the rate of cyber attacks has more than doubled every year since 1998 (see Figure1). We need to find the best ways to make information systems immune to such attacks.

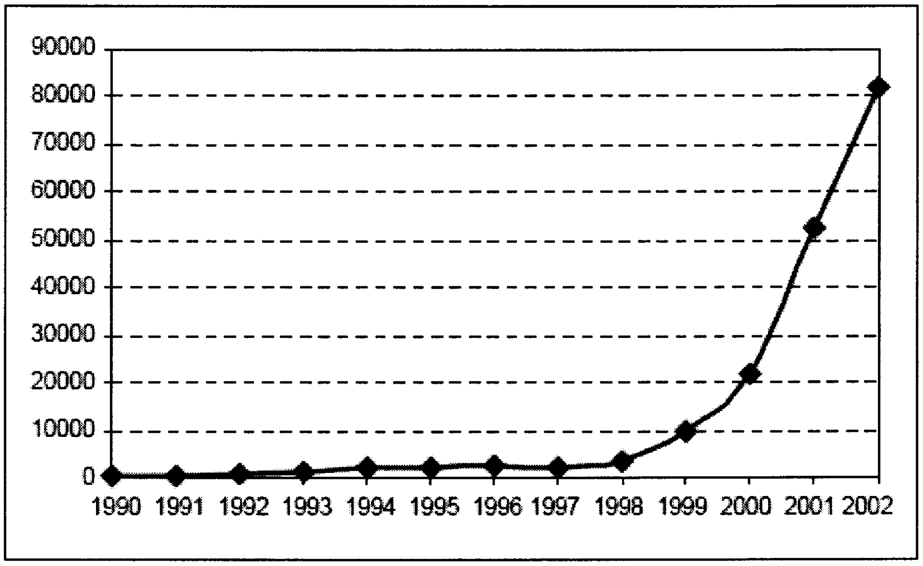


Figure 1. Attack incidents that were reported to Computer Team/Coordination Center (CERT/CC) during the last decade.

Many researches [4,5,6,9] proved that data mining is not only useful but necessary in detecting attacks and analyzing intrusion detection data. However, few have tried to integrate data mining techniques into detecting attacks with the help of database collecting information about the network traffic. In this paper I tried to prove that the method of using databases in intrusion detection and detecting attacks using SQL queries is worthwhile doing and can compete with other techniques [2].

CHAPTER 2

NETWORK INTRUSION DETECTION

2.1 Introduction to Network Intrusion Detection

Intrusion detection in networks is mainly done by means of Network Intrusion Detections Systems [8]. Such a system monitors the traffic on its network segment as a data source. This is generally accomplished by placing the network interface card in promiscuous mode to capture all network traffic that crosses its network segment. Network traffic on other segments and traffic on other means of communication (like phone lines) cannot be monitored.

There are basically two types of intrusion detection techniques: *Anomaly detection* and *Misuse detection*. The most current intrusion detection systems use both techniques.

2.2 Anomaly Detection

Anomaly detection models the normal behavior profile of the traffic in a network. Any events that violate this model are flagged as an intrusion. A profile usually consists of a number of statistical measures on network traffic activities, such as bursts of certain traffic parameters that were not recorded in the normal profile. Deviation from a profile can be computed as the weighted sum of deviations of the constituent statistical measures [6]. Profiles can be aged so that shifts of normal behavior are accounted for. The advantage of anomaly detection is that it can detect unknown intrusions without any prior knowledge about specific intrusions [7]. Intrusive activity, however, does not always coincide with anomalous activity. Four types of possible activities are listed below:

1. Intrusive but not anomalous. These classify as false negatives or type I errors, meaning that the activity is intrusive but because it is not anomalous, the Intrusion Detection System (IDS) fails to detect it. They are called false negatives because the system falsely reports that there was no intrusion.

2. Not intrusive but anomalous. These are false negative or type II errors. The activity is not intrusive but because it is anomalous, IDS reports it as intrusive. It is named in such a way because the intrusion is falsely reported.

3. Not intrusive and not anomalous. These are definitely negatives; the activity is not intrusive and is not reported as intrusive.

4. Intrusive and anomalous. These are definitely positives; the activity is intrusive and is reported as such because it is anomalous.

Anomaly detection suffers from accuracy problems, as building an accurate profile to avoid false negatives may not fully reflect the complex dynamic nature of computer systems leading to false positives.

2.3 Misuse Detection

The main assumption of misuse intrusion detection is that there are attacks that can be precisely encoded in a manner that captures rearrangements and variations of activities that exploit the same vulnerability. In other words it uses patterns of known intrusions called signatures that are first given to IDS to identify intrusions. The key advantage of misuse detection is that once the patterns of known intrusions are stored, future instances of these intrusions can be detected effectively and efficiently. Newly invented attacks, however, will likely go undetected, which leads to unacceptable false negative error rates.

Network-based Intrusion Detection that uses exclusively misuse detection involves looking at the packets on the network as they pass by some sensor. The sensor can only see the packets that happen to be carried on the network segment to which it is attached. Packets are considered to be of interest if they match a signature.

Three primary types of signatures are string signatures, port signatures, and header condition signatures.

CHAPTER 3

DATA MINING IN INTRUSION DETECTION

In today's world, nearly every company is dependent on the Internet to survive, and it is therefore not surprising that the role of network intrusion detection has grown so rapidly [9]. While there may still be some argument as to what is the best way to protect companies' networks (i.e. firewalls, patches, intrusion detection, training, etc.), it is certain that the Intrusion Detection Systems will likely maintain an important role in providing for a secure network architecture. Unfortunately, Intrusion Detection Systems are currently not at their best because they are not capable of identifying all the intrusions (or attempted intrusions).

3.1 Data Mining Definitions

R.L. Grossman in "Data Mining: Challenges and Opportunities for Data Mining during the Next Decade", defines data mining as being "concerned with uncovering patterns, associations, changes, anomalies, and statistically significant structures and events in data." Simply put it is the ability to take data and pull from it patterns or deviations which may not be seen easily to the naked eye. Another term sometimes used is knowledge discovery.

Many different types of data mining algorithms exist today, including link analysis, clustering, association, rule abduction, deviation analysis, and sequence analysis.

Most, if not all, IDS that can be purchased today are based on misuse detection. Current IDS products come with a large set of signatures, which have been identified as unique to a particular vulnerability or exploit. Most IDS vendors also provide regular signature updates in an attempt to keep pace with the rapid appearance of new vulnerabilities and exploits.

3.2 Shortfalls of Data Mining

While the ability to develop and use signatures to detect attacks is a useful and viable approach, there are shortfalls to only using this approach, which should be addressed.

Variants. As stated previously signatures are developed in response to new vulnerabilities or exploits, which have been posted or released. A signature must be unique enough to only alert on malicious traffic and rarely on valid network traffic. The difficulty here is that exploit code can often be easily changed. It is not uncommon for an exploit tool to be released and then have its defaults changed shortly thereafter by the hacker community.

False positives. A common complaint is that the IDS generates too many false positives. It is much more difficult to pick out a valid intrusion attempt if a signature also alerts regularly on valid network activity. The difficulty here is figuring out how much to filter without potentially missing an actual attack.

False negatives. False negatives arise from detecting attacks for which there are no known signatures. This leads to the other concept of false negatives where the IDS does not generate an alert when an intrusion is actually taking place.

Data overload. Another aspect which does not relate directly to misuse detection but is extremely important is the quantity of data that an analyst effectively and efficiently analyze. That being said the amount of data he/she needs to look at seems to be growing rapidly. Depending on the intrusion detection tools employed by a company and its size there is the possibility for the number of logs to reach millions of records per day.

3.3 How Can Data Mining Help?

Data mining can help improve intrusion detection by adding a level of focus to anomaly detection. By identifying bounds for valid network activity, data mining will aid an analyst in his/her ability to distinguish attack activity from common everyday traffic on the network.

Variants. Since anomaly detection is not based on pre-defined signatures, the concern with variants in the code of an exploit is not as great because we are looking for abnormal activity versus a unique signature. An example might be a Remote Procedure Call (RPC) buffer overflow exploit whose code has been modified slightly to evade an IDS using signatures. With anomaly detection, the activity would be flagged since the destination machine has never seen an RPC connection attempt and the source IP was never seen connecting to the network.

False positives. In regards to false positives, there has been some work to determine if data mining can be used to identify recurring sequences of alarms in order to help identify valid network activity, which can be filtered out.

False negatives. By attempting to establish patterns for normal activity and identifying the activity which lies outside identified bounds, attacks for which signatures have not been developed might be detected. An extremely simple example of how this would work would be to take a web server and develop a profile of the network activity seen to and from the system. Let us say that the web server is locked down and only connections to ports 80 and 443 are ever seen to the server. Thus, whenever a connection to a port other than 80 or 443 is seen, the IDS should identify that as an anomaly. While this example is quite simple this could be extended to profiling, not only individual hosts, but entire networks, users, traffic based on days of the week or hours in a day, etc.

Data overload. The area where data mining is sure to play a vital role is in the area of data reduction. With current data mining algorithms there exists the capability to identify or extract data, which is most relevant, and provide analysts with different "views" of the data to aid in their analysis.

3.4 Difficulties with Data Mining in Intrusion Detection

The concept of data mining has been around for many years. Despite this, data mining in intrusion detection is a relatively new concept. Thus, there will likely be obstacles in developing an effective solution. One is the fact that even though the concept of data mining has been around for some time, the amount of data to be analyzed and its complexity is increasing dramatically. As stated previously, it is possible for a company to collect millions of records per day, which need to be analyzed for malicious activity. With this amount of data to analyze, one can guess that data mining will become quite computationally expensive. Unfortunately, processing power or memory is not always cheap or available. Of course, there may be the argument that you only need samples of the data in order to generate profiles, but analyzing anything, especially network traffic, without all the data could lead to false conclusions. Another obstacle will be tailoring data mining algorithms and processes to fit intrusion detection. An effort to identify how the data needs to be looked at in order to provide us with a better picture is integral in providing accurate and effective results.

Obviously data mining and anomaly detection is not a silver bullet for intrusion detection, nor should it be a replacement for misuse detection. The goal should be to effectively integrate anomaly detection and misuse detection to create an IDS, which will allow an analyst to more accurately and quickly identify an attack or intrusion on their network.

CHAPTER 4

INTRUSION DETECTION WITH SQL ANALYSIS AND DATABASE IDS INTEGRATION

4.1 Novelties of Intrusion Detection Systems

Intrusion detection systems even well configured ones that utilize data-mining techniques, are rather common in companies that care about their security and non-vulnerability. Just like anything else in the world, intrusion detection systems are not perfect.

Here are some basic drawbacks of current IDS products that limit effectiveness of this security solution [10]:

Detection Accuracy – Current IDS products are noted for generating a high rate of false positives.

Enterprise Scalability – Network IDS deployments are now enterprise scale. The concept of "small" and "big" networks has significantly changed in the past years. Two years ago, a network of 50 sensors was considered large, but today networks consisting of hundreds of sensors are not uncommon. Network IDS has to keep up with the enterprise scale in order to function appropriately.

Product complexity – Today's IDS are still difficult to install, update and manage on a daily basis.

Growing IDS Evasion – new and polymorphic attacks are used very widely and can easily go around most IDS products.

Performance barriers – keeping up speed of traffic and attack detection.

Passive device – IDS are considered to be non-critical devices because of their passiveness while detecting attacks. Preventing attacks is considered more important. It needs to be understood that detection is much more valuable than prevention.

Last but not least is intrusion detection management – with millions of records per day, hundreds or thousands of alerts are certainly overwhelming to security administrators, and of course, this prevents them from doing their primary job efficiently.

There is a clear need for development and improvement in this area. One of the improvements that is just starting make its way into the intrusion detection world is logging to database system. Most intrusion detection systems log to a flat file, which is later analyzed by some specially designed programs or manually browsed by security analysts of the company.

4.2 Advantages and Disadvantages of Database Logging

Clearly, nobody wants to manually go through hundreds of thousands or millions of lines of logs and alerts, and even worse try to make sense out of them. Some systematic advanced technological process has to be developed. I did extensive research in this area and discovered that logging intrusion detection data to a database would create multiple advantages. Information that is being written into the database can successfully and efficiently be:

Stored – no limit on the size of the database, such database systems as Oracle are very robust and can handle enormous chunks of data. A flat file can grow uncontrollably and eventually take up all of the memory, causing a system crash or an intrusion detection system malfunction.

Analyzed – by means of Structured Query Language (SQL), data in the database can be easily found, classified, analyzed and researched.

Categorized – multiple tables for multiple periods of time could be created, then linked together for more functional design.

Accessed – data access is also a very important aspect. Database systems provide easy access to the required information at any time of day or night from any location, whereas it might be hard and insecure to share a file on the network.

Disadvantage of such an improvement is an insignificant increase of local traffic and response time. It is understandable that if a database system includes numerous amounts of records, the response time would not be immediate, but would nevertheless be within a reasonable limit.

4.3 Project Description and Analysis

4.3.1 Two Models

I propose and describe two models in this project. The first model is a local area network or a segment of a computer network that has at least one sensor, an intrusion detection system running on it, and a database server that will accept logs and alerts written into it. One or more sensors detect attacks, while intrusion detection system outputs data into the database system as shown on Figure 4.1. A security analyst has access to the database server, and will analyze data collected from the whole network.

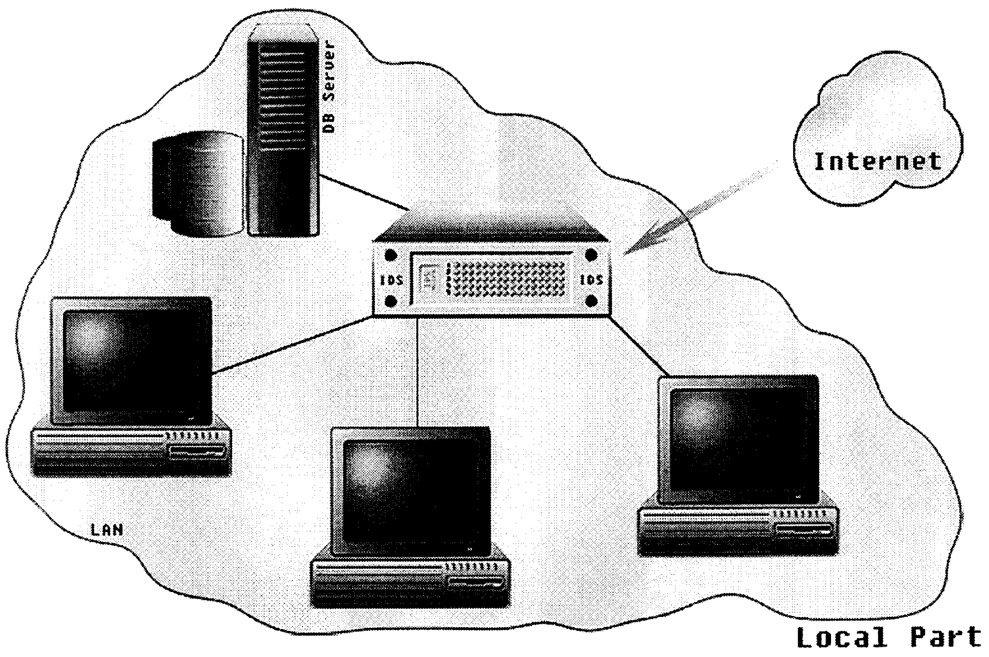


Figure 4.1. Model 1 LAN with IDS and database system on it.

The second model is a little bit more complicated. It has one secure centralized database server and multiple intrusion detection systems (they do not even have to be of the same vendor) on different networks. ID systems connect to the database server and write alerts into it. This model is designed for detecting attacks on a larger scale and can give a better understanding of the types of attacks that are launched and possible ways to stop them. This model can be seen in Figure 4.2.

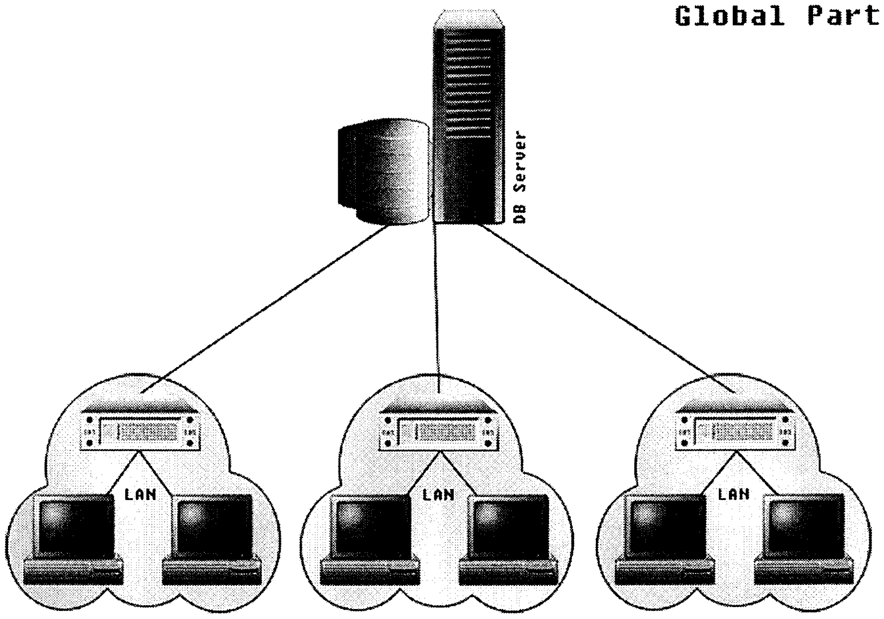


Figure 4.2. Model 2 – Global database server with multiple IDS writing into it.

Let us consider an example. A fairly large company has multiple sites at different geographic locations and, therefore, multiple networks. Let us assume that each network has an intrusion detection system running on it. In a regular environment, intrusion detection systems detect some of the attacks that are launched against that particular network. They are not detecting attacks on a global basis, however, where a simple port sweep on one segment of the network, might be part of an attack on the whole enterprise. Therefore, if all intrusion detection systems would write data into a centralized database server, it would be much easier and possible to detect global attacks. Failure to realize global attacks can lead to catastrophic consequences and loss of money. This is also a very good way of detecting new types of viruses, worms and security holes – if each IDS reports unusual number of similar attacks, it may be that a new virus or a worm is trying to compromise the network or individual machines. This system can help eliminate the virus spread and allow a faster response to obtain and apply security updates.

It may be argued that the security of the database server would be of question. It is understandable that information is of a high value and has to be protected very well. One of the ways to protect the flow of the data between the intrusion detection system and database system is secure logging using Stunnel. This approach involves encryption of traffic between the IDS and the database server using Stunnel or Secure Tunnel, which is an open source package [11] that provides a secure tunnel between two hosts. It can even serve as a simplification, because after the configuration of Stunnel, IDS can be configured in such a way that it assumes that the database system is running on the local machine and Stunnel transfers all the communication to the remote database server.

4.3.2 Sharing Database System with Firewalls

A combination of an intrusion detection system, a database server, and a firewall is also a very good example of improvement. This approach gives the advantage of using existing investments in firewall and IDS technology. If an attack was detected, an intrusion detection system logs an alarm into the database, and from the alarm is sent to the firewall. The firewall, inspects the alarm, finds the necessary information, and blocks the offending traffic. This approach is effective against some known attacks as well as brand new types of attacks. It is especially effective against long-time running attacks that occur over multiple packets, so that identification can be made earlier. Unfortunately, this method is not very effective against initial single packet attacks. This is so because by the time the attack is detected, the single offending packet has already passed. It is possible however, to block subsequent attacks.

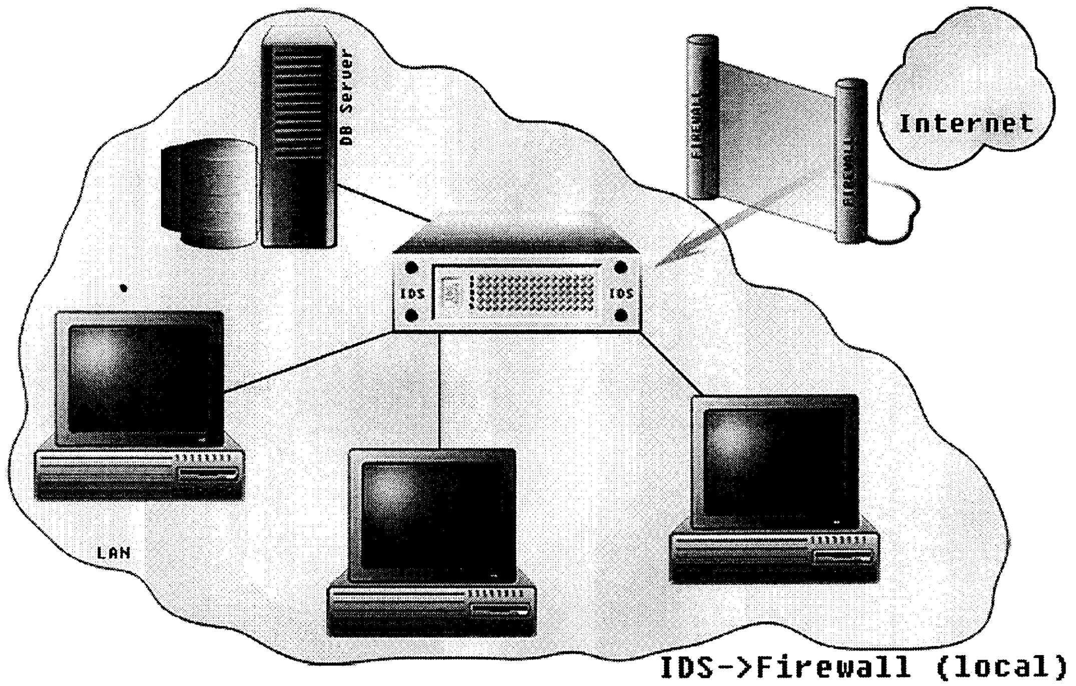


FIGURE 4.3 IDS and firewall integration.

4.3.3 Data Mining Capabilities of SQL

SQL based analysis can be classified as a special data mining technique. As mentioned earlier, the simplest definition of data mining is "discovering interesting patterns". Researching data in the database with the SQL technique matches this definition exactly. In addition, different data types require different analysis methods. In this project's database, each column already has a specific data type; so one part of work is conveniently done already. The data is ready to be analyzed.

Database mining[11], which incorporates the ability to directly access data stored in a database, is one step beyond the core technology of data mining. The distinction (database rather than data) might seem to be a trivial improvement but like most transitions from technology to solutions, it requires a major leap for developers.

4.3.4 Preliminary Research

I performed extensive research in order to pick tools, known programming languages, and a database system for this project. I first researched different database systems that match the project's needs and are freely available. Two database systems supported by NJIT were Oracle and MySQL. I considered installing my own database server but because of lack of time and support I decided to go with a system that is already installed, configured properly, and supported and maintained professionally and constantly. Conveniently, NJIT offered all of the above, so the Oracle database system was chosen, mainly because of its known advances in robustness, stability, properties and functionality over other database systems.

After the choice of Oracle was finalized, I found out that it supports only two types of connections from outside, JDBC and Pro*C/C++. Unfortunately, I did not know JDBC or the Java programming language, so I had to choose Pro*C/C++.

4.3.5 Programming

In order to make the intrusion detection system write data to the Oracle database system, I had to write a program, using the Pro*C/C++ precompiler. Oracle Pro*C/C++ Precompiler is a programming tool that allows embedding SQL statements into a high-level (C++) source program. The precompiler accepts the source program as input, translates the embedded SQL statements into standard Oracle runtime library calls, and generates a modified source program that you can compile, link, and execute in the usual way.

This program takes outputted alert logs from an IDS, connects to the database, logs in with username and password, and uses SQL statements to write data into the table of Oracle database. Although it seemed like a trivial task, the program was rather difficult to complete, primarily because of low informational resources and references. The source code for the program is available (in the appendix) at the end of the paper.

The program can also be thought as Database Plugin for an intrusion detection system. What is valuable is that the plugin can be applied not to a specific intrusion detection system, but to various types of IDS (misuse or anomaly type), with only minor code modifications to compensate for output differences. In our case, the plugin is specific to the Oracle database system. It should not be a problem, in any case, because Oracle is very widely used in the world of database systems.

4.3.6 Query-Based Analysis

Once the data from an intrusion detection system is fed into the database, we can do something with it. Because SQL is proven to be the most reliable way to pull the data from a database, it seems logical to use it for this project.

4.3.6.1 Alert Management. There are two datasets that were used in this project. One is a sample alert log that was obtained from one of the graduate students, who developed a custom intrusion detection system at NJIT. The format of the alert log is the following:

Date - date of the attack was attempted;

Time – time of the attack was attempted;

Server IP – IP address of the target machine;

Client IP – IP address of the source machine;

Protocol – indicates the ATTACK protocol (1—ICMP, 2 – IGMP, 6 -- TCP, 9 – IGRP, 17 – UDP, 47 – GRE, 50 – ESP, 51 – AH, 57 – SKIP, 88 – EIGRP, 89 – OSPF, 115 – L2TP)

Server port – Port of the target machine;

Client port – Port of the source machine;

Session entropy - the entropy of a server port being accessed – the closer the value is to zero, the more likely that the port is accessed;

Service definition - a Boolean, indicating whether the server port is actually opened (0: not open, 1: open);

Session suspiciousness - the likeliness that the session is actually an attack session (<0 attack, >= 0 normal).

Using the mentioned fields and their meanings, appropriate queries can be constructed.

Here are just some that are possible:

- Output all/some alerts for a particular date

```
select *
from LOG
where DT = 'MM/DD/YYYY';
```

- Output all/some alerts for a particular timeframe on a particular date; insert '??' if seconds or minutes are not essential

```
select *
from LOG
where TIME = 'HH:MM:SS'
and DT = 'MM/DD/YYYY';
```

- Output all/some alerts for IPs for a particular period of time

    - what are the most aggressive IPs

```
select DISTINCT S_IP
from LOG
where DT = 'MM/DD/YYYY';
```

    - what IPs have been victims the most – that is of the greatest interest of intruders

```
select DISTINCT C_IP
from LOG
```

```
        where DT = 'MM/DD/YYYY';
```

- what is the most popular protocol used by intruders for a particular source IP, date range, timeframe, etc.

```
        select *
        from LOG
        where PROT = 'X' and
        DT = 'MM/DD/YYYY' and
        TIME like 'HH:MM:SS';
```

- Output all/some server/client ports (most popular port to be scanned, least popular)

```
        select DISTINCT S_PORT, C_PORT
        from LOG
        where DT = 'MM/DD/YYYY'
        order by S_PORT asc, C_PORT asc;
```

- Output if any ports were open for particular date

```
        select *
        from LOG
        where SERVICE_DEF = 1 and
        DT = 'MM/DD/YYYY';
```

The queries listed above are relatively simple, however due to broad SQL capabilities, more complex queries, such as nested queries, can be in fact constructed. For example, we can output the IP addresses of the attacker and target machine for a particular date, specific time (to hour), for a particular protocol used, and whether the attack was successful.

```
    select C_IP, S_IP
    from LOG
    where DATE in
        (select date from LOG
        where DT='MM/DD/YYYY')
    and TIME like 'HH:%'
    and PROT in
```

```
        (select PROT from LOG
        where PROT = '6')
    and SESSION_S >= 0
```

Above data and queries are strictly for the purpose of helping administrators and network security personnel manage alerts.

In addition to creating queries, stored procedures and triggers can also be programmed for better analysis purposes. Needless to say, the time it takes for a query to be executed and to return results is very short. Considering the amount of records in an intrusion detection database, this is a very good way to obtain needed information, especially when it is urgently needed.

4.3.6.2 Network Data and Query-Based Statistical Analysis. To prove that database systems and SQL techniques are useful in intrusion detection and worthwhile doing, statistical analysis of the intrusion detection data was conducted.

The second data set that was used is the DARPA'98 IDEVAL data, generated by MIT Lincoln labs in 1998 as the intrusion detection offline evaluation project sponsored by DARPA [13]. This data was collected in an isolated controlled network environment with simulated normal background traffic and attacks. Only DOS and probing attacks are considered in our experiment. The original DARPA data is organized by a total of 7 weeks. I analyzed all of seven weeks of this data.

There are two approaches to represent traffic features of intrusion detection systems [2]. Most systems measure the average rates, or the number of occurrences of the selected traffic parameters and normalize these parameters using the means and standard deviations, which are maintained in a reference models. In data, that I used, network features are represented in probability density functions (PDF) and using similarity metrics. This data is being called DISTANCE data.

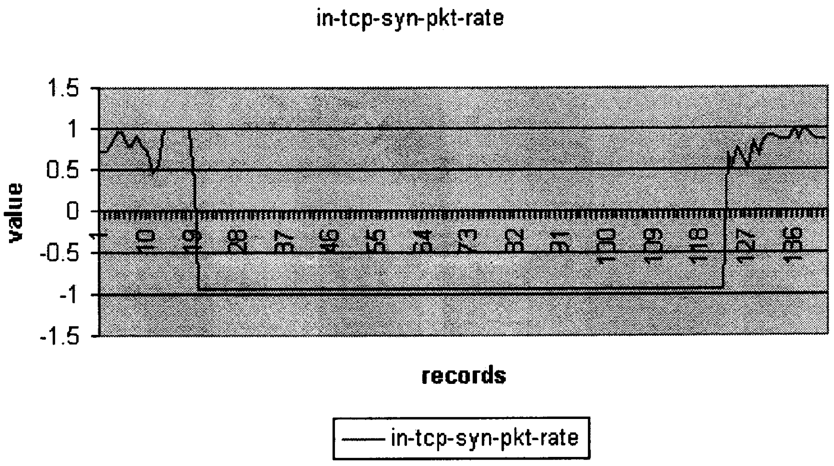
There are 45 different network parameters in this data set and one label that signifies whether a particular record is an attack or valid network traffic. A value of '-1' means that it is an attack and a value of '1' means that it is non-attack.

The purpose of doing the analysis is to try to see how many attacks we can successfully predict, matching our predictions to attack truths that are available to us in the 'label' column.

A new database table was created and with a few modifications to the program that was described earlier all the data in the data set was loaded into the Oracle database.

However, before creating SQL queries for statistical analysis, I needed to know which fields and how, influences the fact if there was an attack or no. Therefore, I performed a field analysis – I looked for patterns in the data and tried to identify which fields are most important in identifying attacks.

Each of the 45 fields was graphed for each of the seven weeks of DARPA data set. Because SQL itself does not provide graphical capabilities and Oracle visualization modules were not available during this project, the data was graphed in MS Excel. An example of a valid useful pattern can be seen on Figure 4.4. The figure represents two graphs of the same field; with the top graph depicted attack data and the bottom depicting valid traffic. The difference is clearly noticeable – most of the attacks have negative value for this field, however non-attack data has only positive values.



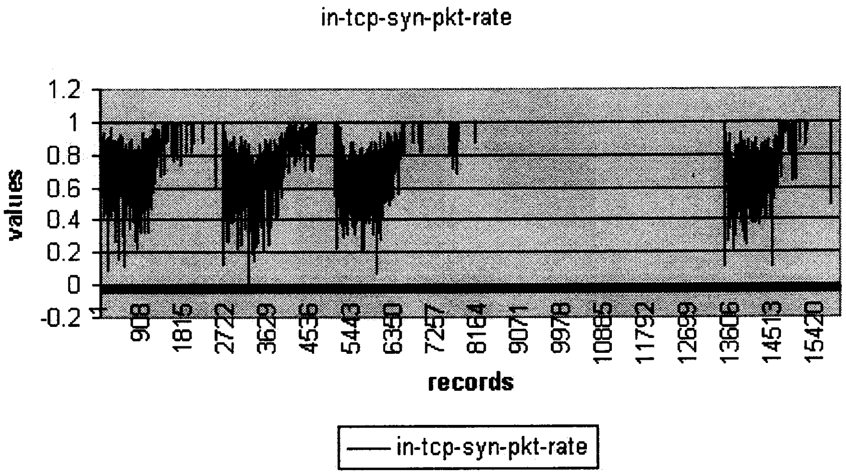


Figure 4.4 Field 'Input TCP SYN Packet Rate' for attack data (top graph) and non-attack data (bottom graph).

As I graphed each field and queried for it in the database to catch exceptions, I constructed the following list along with data description and my observations.

in_ip_pkt_len = 1 only if it is legitimate traffic (except for week 5 – one occurrence and week 7 – three occurrences)

in_ip_pkt_rate = 1 only if it is legitimate traffic

in_ip_byt_rate and in_ip_pkt_rate are almost the same in_ip_frag_rate is <0 only in case of attack (finds very small number of attacks)

in_ip_defrag_rate and in_ip_frag_rate are almost the same

in_ip_csum_error_rate <0 only in case of an attack

in_tcp_pkt_len = 1 – means legitimate traffic
(except for week1 – one occurrence,
week3 – one occurrence,
week4 – three occurrences,
week5 – 149 occurrences,
week6 – 55 occurrences,
week7 – 9 occurrences)

in_tcp_pkt_rate = 1 only in case of legitimate traffic

in_tcp_syn_pkt_rate < 0 in case of attack (except for week7 – one occurrence, week5 – one occurrence, week3 – one occurrence)

in_tcp_fin_pkt_rate and in_tcp_syn_pkt_rate are very similar

in_tcp_rst_pkt_rate – attack and legitimate traffic data is very similar

in_tcp_con_new_opened < 0 only in case of an attack

in_tcp_con_new_closed and in_tcp_con_new_opened are very similar

in_tcp_con_new_aborted < 0 in case of an attack
week7 attacks=86, FP = 13
week6 attacks= 295, FP = 10
week5 attacks= 174, FP = 25
week4 attacks= 0, FP = 0
week3 attacks= 84, FP = 14
week2 attacks= 0, FP = 0
week1 attacks= 0, FP = 0

in_tcp_con_half_opened_ratio (no significant patterns found)

in_tcp_con_duration (no significant patterns found)

in_tcp_con_diff_src and in_tcp_con_duration are very similar

in_tcp_con_diff_src and in_tcp_con_diff_dst are very similar in_tcp_con_anomalous_entropy and in_tcp_con_diff_dst are very similar

in_icmp_pkt_len (no significant patterns found)

in_icmp_pkt_rate < 0 in case of an attack

in_icmp_byt_rate and in_icmp_pkt_rate are very similar

in_icmp_diff_src similar to in_icmp_byt_rate – not of significant value

in_icmp_diff_dst < 0 in case of an attack, except:
week7 attacks=4, FP = 11
week6 attacks= 556, FP = 1
week5 attacks= 327, FP = 2
week4 attacks= 2, FP = 4
week3 attacks= 0, FP = 1
week2 attacks= 0, FP = 0
week1 attacks= 3, FP = 0

in_udp_pkt_len (no significant patterns found)

in_udp_pkt_rate < 0 is in case of an attack except for:
week5 one occurrence
week3 one occurrence
week1 two occurrences
(not of very significant value)

in_udp_byt_rate and in_udp_pkt_rate are very similar

in_udp_diff_src - minor influence

in_udp_diff_dst < 0 in case of an attack
week7 attacks= 5, FP = 10
week6 attacks= 3, FP = 0
week5 attacks= 12, FP = 8
week4 attacks= 2, FP = 73
week3 attacks= 0, FP = 13
week2 attacks= 5, FP = 2
week1 attacks= 0, FP = 1
(as it is seen, it is not worth taking this field into account, because ratio of attacks detected and false positives is too high)

out_ip_pkt_len – no significant patterns found

out_ip_pkt_rate – no significant patterns found

out_ip_byt_rate and out_ip_pkt_rate are very similar – no significant patterns found

out_tcp_pkt_len – no significant patterns found

out_tcp_pkt_rate = 1 when the traffic is legitimate, except in week6 one occurrence

out_tcp_con_diff_src – no significant patterns found

out_tcp_con_diff_dst – no significant patterns found

out_icmp_pkt_len – no significant patters found

out_icmp_pkt_rate < 0 in case of an attack

out_icmp_diff_src – no significant patterns found

out_icmp_diff_dst – no significant patterns found

out_udp_pkt_len – no significant patterns found

out_udp_pkt_rate – no significant patterns found

out_udp_diff_src – no significant patterns found

out_udp_diff_dst – no significant patterns found – redundant, similar graph was observed before

io_icmp_anomalousecho_reply < 0 in case of attacks in all cases

After analyzing all DARPA data with the help of SQL queries and Excel graphs, a list of most significant columns that are exclusively negative in case of attacks was acquired:

in_ip_frag_rate, in_ip_csum_error_rate, in_tcp_syn_pkt_rate, in_tcp_con_new_opened, in_icmp_pkt_rate, out_icmp_pkt_rate, io_icmp_anomalousecho_reply

Based on the fields above, the following query was constructed:

```
SELECT io_icmp_anomalousecho_reply, label
FROM week7
WHERE
in_ip_frag_rate<0 OR in_ip_csum_error_rate<0 OR
in_tcp_syn_pkt_rate<0 OR
in_tcp_con_new_opened<0 OR
in_icmp_pkt_rate<0 OR
out_icmp_pkt_rate<0 OR
io_icmp_anomalousecho_reply<0;
```

This query does not return any False Positives, which is good, however it has higher number of False Negatives. Results of attack discovery, is shown in the Table 4.1.

	Total records	**Total attacks**	**Valid traffic**	**False positives**	**False negatives**	**False positive rate**	**False negative rate**	**Misclassification rate**
Week 1	2650	3	2647	0	2	0	0.666666667	0.000754717
Week 2	8368	186	8182	0	186	0	1	0.022227533
Week 3	24395	115	24280	0	7	0	0.060869565	0.000286944
Week 4	7774	42	7732	0	39	0	0.928571429	0.005016722
Week 5	8724	809	7915	0	466	0	0.576019778	0.053415864
Week 6	15905	1058	14847	0	626	0	0.59168242	0.039358692
Week 7	16066	142	15924	0	27	0	0.190140845	0.001680568
Total	83882	2355	81527	0	1353	0	0.574522293	0.016129801

Table 4.1 Table of results of the first query.

The formulas for the above columns are the following:

False Positives – the number of valid traffic samples classified as attacks.

False Negatives – the number of attacks classified as valid traffic.

False Positive Rate – the ratio between the false positives and total number of valid traffic samples.

False Negative Rate – the ratio between the false negatives and the total number of attack samples.

Misclassification Rate – the ratio between the total misclassifications and the total number of samples. (FP + FN)/total

As we can see, that results are ambiguous. For weeks one, three and seven, results were acceptable and even good, however for other weeks despite the fact that there are no false positives, a lot of attacks still left unidentified.

In order to improve results, based on the field analysis research, I added more fields and obtained the following query:

```
SELECT io_icmp_anomalousecho_reply, label
FROM week7
WHERE
in_ip_frag_rate<0 OR
in_ip_csum_error_rate<0 OR
in_tcp_syn_pkt_rate<0 OR
in_tcp_con_new_opened<0 OR
in_icmp_pkt_rate<0 OR
out_icmp_pkt_rate<0 OR
io_icmp_anomalousecho_reply<0 OR
in_tcp_con_new_aborted<0 OR
in_icmp_diff_dst<0 OR
in_udp_pkt_rate<0;
```

Although this query introduces False Positives, it eliminates a lot of False Negatives. Results of this query analysis are displayed in Table

4.2. Obvious improvements can be seen after including the above listed fields.

	Total records	Total attacks	Valid traffic	False positives	False negatives	False positive rate	False negative rate	Misclassification rate
Week 1	2650	3	2647	2	0	0.0007556	0	0.0007547
Week 2	8368	186	8182	0	186	0	1	0.0222275
Week 3	24395	115	24280	17	3	0.0007002	0.026087	0.0008198
Week 4	7774	42	7732	4	37	0.0005173	0.8809524	0.005274
Week 5	8724	809	7915	29	151	0.0036639	0.1866502	0.0206327
Week 6	15905	1058	14847	10	111	0.0006735	0.1049149	0.0076077
Week 7	16066	142	15924	25	21	0.00157	0.1478873	0.0028632
Total	83882	2355	81527	87	509	0.0010671	0.2161359	0.0071052

Table 4.2 Results of the query-based statistical analysis conducted.

I am at present unaware of existence of similar analyses, and therefore do no have at my disposal the results from such analysis. However the evaluation of Feature-Extraction Algorithms has been done in this area, on the same data set by Zheng Zhang [2].

We can compare our results against results of that study.

There were different classification architectures used. Table 4.3 shows result of Feature-Extraction Algorithm study.

Type of classification architecture	False positives	False negatives	False positive rate	False negative rate	Misclassification rate
Backpropagation	2	23	0.000126	0.168	0.00156
Decision Tree	8	5	0.000502	0.0365	0.000809
K-Nearest Neighbor Tree	0	16	0	0.117	0.000996
Linear Vector Quantization	7	21	0.000439	0.153	0.00174
Support Vector Machine	5	21	0.000311	0.153	0.00162

Table 4.3 Result of feature-extraction algorithm study.

In order to improve results, I tried to figure out what would be the best threshold that should be in the query. Initially, it was zero. I experimented with values from –0.5 to 0.4 and I obtained Table 4.4

Threshold	Misclassification rate
-0.5	0.011993038
-0.4	0.011969195
-0.35	0.006914475
-0.3	0.006604516
-0.25	0.006687966
-0.2	0.006783338
0	0.007164827
0.1	0.007140984
0.3	0.007939725
0.4	0.011432727

Table 4.4 Experimental values of threshold and corresponding misclassification rate.

For visualization purposes, I also constructed the graph that can be observed in Figure 4.5

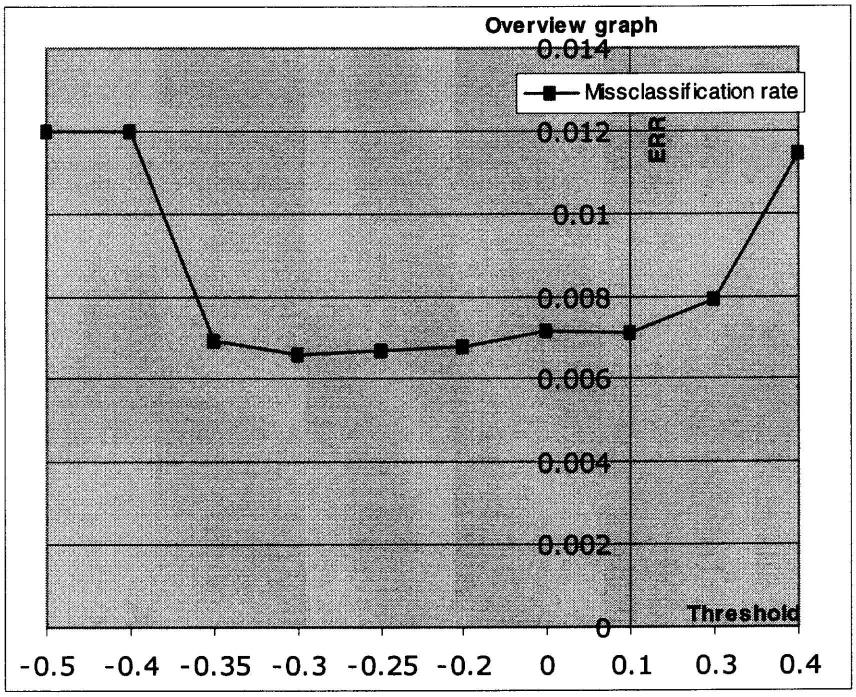


Figure 4.5 Graph of experimental values of threshold and corresponding misclassification rate.

It is seen from the table and the graph that the threshold value with the smallest Misclassification rate is - 0.3, which equals to 0.006604516. So the final accumulation of results can be seen in Table 4.5

	Total records	Total attacks	Valid traffic	False positives	False negatives	False positive rate	False negative rate	Misclassification rate
week 1	2650	3	2647	0	0	0	0	0
week 2	8368	186	8182	0	186	0	1	0.02222753
week 3	24395	115	24280	1	7	4.11862E-05	0.060869565	0.00032794
week 4	7774	42	7732	3	39	0.000387998	0.928571429	0.00540262
week 5	8724	809	7915	7	158	0.000884397	0.195302843	0.01891334
week 6	15905	1058	14847	1	123	6.73537E-05	0.116257089	0.00779629
week 7	16066	142	15924	5	24	0.000313991	0.169014085	0.00180505
Total	83882	2355	81527	17	537	0.00020852	0.228025478	0.0066045

Table 4.5 Final table of results.

As we can see the above SQL-based analysis along with field study, gives the same order of results as such feature extraction algorithms as Backpropagation Neural Network, Linear Vector Quantization and Support Vector Machine for DISTANCE data.

Although the time that it takes for writing final SQL queries may vary depending on the attack complication, once most of the known attacks are figured out and SQL signatures are finished, the speed of detecting attacks should increase.

4.3.6.3 Result Analysis. As can be seen from Table 4.5 that results of the query-based SQL analysis vary from week to week. For weeks one and three the results are very good, even better than Feature-Extraction Algorithm study numbers. Weeks four, six and seven have average results but weeks two and five have really high false negative rates. What is interesting in this study is that there are not many false positives but many more false negatives. This may be because of the way we analyzed the data – we considered only network features that had the most impact on the detection of the attack, neglecting features with smaller attack detection rates. One way to improve these results would be to analyze more data, and try to identify more patterns out of existing network features. Another way would be to include more fields to reach the maximum level of detection of attacks, but then the false positive rate will increase. Another good way of improving results and automating the process of finding interesting patterns would be writing stored procedures in our database. It would, however, require higher-level programming and database administration skills.

CHAPTER 5

CONCLUSIONS

The objective of this project was to integrate a database system with an intrusion detection system and with the help of SQL analysis, prove that it will bring improvements to network security.

Leveraging the power of today's commercial database systems, it is possible to simplify and improve the task of managing and searching intrusion detection data. This project is an inexpensive and well-performing way to make it easier for the administrators and security personnel to archive, manipulate the information, audit and mine the data, and easier for managers to interpret it. The results of the analysis show that this can be an effective system with only a 0.001 misclassification rate. The system captured approximately 82% of the attacks from a particular data set. Programming languages such as Pro*C/C++ or Perl can create programs that would utilize SQL queries in their statistical analysis algorithms to further automate the process of detecting intrusions. If the latest statistical intrusion detection algorithms are used in conjunction with such a system, the capture rate will increase.

SQL and database systems alone cannot entirely perform statistical analysis for us. They are, however, a means of storing, accessing, searching, and presenting data to analysts, allowing them to be more efficient at their jobs.

APPENDIX

SOURCE CODE OF THE PROGRAM THAT CONNECTS IDS TO THE DATABASE MANAGEMENT SYSTEM.

How to compile:

First Pro*C file has to be compiled:

```
proc dbaccess.pc auto_connect=true
```

After it generates C file, we need to compile it:

```
gcc dbaccess.c –c
```

C file is compiled into the library and ready to be compiled into the final program:

```
g++ sample5.cc dbaccess.o $ORACLE_HOME/lib32/libclntsh.so
```

Some environmental variables have to be set in order for the program to work:

```
setenv LD_LIBRARY_PATH
"$ORACLE_HOME/lib32:$LD_LIBRARY_PATH"
```

Source code of Pro*C file.

```
/*  dbaccess.pc */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define PWD_LEN        17
typedef char asciiz[PWD_LEN];
struct log_entry
{
        asciiz Date;
        asciiz Time;
        asciiz sIP;
        asciiz cIP;
        asciiz Prot;
        asciiz sPort;
        asciiz cPort;
        asciiz sEnt;
        asciiz sDef;
        asciiz sS;
};
void init();
void uninit();
void insert_log(asciiz[2], asciiz[2], asciiz[2], asciiz[2], asciiz[2],
asciiz[2], asciiz[2], asciiz[2], asciiz[2], asciiz[2], int);

EXEC SQL INCLUDE SQLCA;

EXEC SQL TYPE asciiz IS CHARZ(PWD_LEN) REFERENCE;

/* Declare function to handle unrecoverable errors. */
void sql_error(char *);

void init()
{
/* Connect to ORACLE. */

   EXEC SQL WHENEVER SQLERROR DO sql_error("ORACLE error--");

   printf("\nConnected to ORACLE!\n");
}

void uninit()
{
    EXEC SQL COMMIT WORK RELEASE;
}


void  sql_error(msg)
  char *msg;
{
    char err_msg[512];
    size_t buf_len, msg_len;

    EXEC SQL WHENEVER SQLERROR CONTINUE;

    printf("\n%s\n", msg);

/* Call sqlglm() to get the complete text of the error message. */
    buf_len = sizeof (err_msg);
    sqlglm(err_msg, &buf_len, &msg_len);
    printf("%.*s\n", msg_len, err_msg);

    EXEC SQL ROLLBACK RELEASE;
    exit(EXIT_FAILURE);
}



/*void insert_log(asciiz[2], asciiz[2], asciiz[2], asciiz[2], asciiz[2],
asciiz[2], asciiz[2], asciiz[2], asciiz[2], asciiz[2])
        asciiz DT[]; asciiz TIME[]; asciiz S_IP[]; asciiz C_IP[]; asciiz
PROT[];
        asciiz S_PORT[]; asciiz C_PORT[]; asciiz S_ENTROPY[]; asciiz
SERVICE_DEF[]; asciiz SESSION_S[];
{
        EXEC SQL INSERT INTO LOG (DT, TIME, S_IP, C_IP, PROT,
S_PORT, C_PORT, S_ENTROPY, SERVICE_DEF, SESSION_S)
                                        VALUES (:DT, :TIME, :S_IP, :C_IP,
:PROT, :S_PORT, :C_PORT, :S_ENTROPY, :SERVICE_DEF,
:SESSION_S);
}

*/


void insert_log(DT, TIME, S_IP, C_IP, PROT, S_PORT, C_PORT,
S_ENTROPY, SERVICE_DEF, SESSION_S, size)
```

```
        asciiz DT[]; asciiz TIME[]; asciiz S_IP[]; asciiz C_IP[]; asciiz
PROT[];
        asciiz S_PORT[]; asciiz C_PORT[]; asciiz S_ENTROPY[]; asciiz
SERVICE_DEF[]; asciiz SESSION_S[]; int size;
{
        int f;
        for(f=0; f<size; f++)

        EXEC SQL INSERT INTO LOGS (DT, TIME, S_IP, C_IP, PROT,
S_PORT, C_PORT, S_ENTROPY, SERVICE_DEF, SESSION_S)
                                        VALUES (:DT[f], :TIME[f], :S_IP[f],
:C_IP[f], :PROT[f], :S_PORT[f], :C_PORT[f], :S_ENTROPY[f],
:SERVICE_DEF[f], :SESSION_S[f]);

}
```

Source code of C++ file.

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <cstdlib>
#include <string.h>
#include <cstring>

extern "C" {
#include "dbaccess.h"
}

using namespace std;

int main()
{

  ifstream in;
  string str="";
  string tmp;
  int counter = -1;
  vector<string> V0;
  vector<string> V1;
  vector<string> V2;
```

```
vector<string> V3;
vector<string> V4;
vector<string> V5;
vector<string> V6;
vector<string> V7;
vector<string> V8;
vector<string> V9;
    in.open("input.dat");

while( !in.eof() )
{

 counter++;
 switch (counter)
 {
 case 0:
  tmp = str;
  in>>str;
  if (tmp != str)
    V0.push_back(str);
  break;

 case 1:
  in>>str;
  V1.push_back(str);
  break;

 case 2:
  in>>str;
  V2.push_back(str);
  break;

 case 3:
  in>>str;
  V3.push_back(str);
  break;

 case 4:
  in>>str;
  V4.push_back(str);
  break;

 case 5:
  in>>str;
```

```
    V5.push_back(str);
    break;

  case 6:
    in>>str;
    V6.push_back(str);
    break;

  case 7:
    in>>str;
    V7.push_back(str);
    break;

  case 8:
    in>>str;
    V8.push_back(str);
    break;

  case 9:
    in>>str;
    V9.push_back(str);
    break;

  default:
    counter = -1;

  }

}
     in.close();

     int size = V0.size(); typedef char asciiz[17];
            asciiz DT[size];
            asciiz TIME[size];
            asciiz S_IP[size];
            asciiz C_IP[size];
            asciiz PROT[size];
            asciiz S_PORT[size];
            asciiz C_PORT[size];
            asciiz S_ENTROPY[size];
            asciiz SERVICE_DEF[size];
            asciiz SESSION_S[size];

      for (int i = 0; i < size; i++){
            strcpy(DT[i], V0[i].c_str());
            strcpy(TIME[i], V1[i].c_str());
            strcpy(S_IP[i],    V2[i].c_str());
            strcpy(C_IP[i],    V3[i].c_str());
            strcpy(PROT[i],    V4[i].c_str());
            strcpy(S_PORT[i], V5[i].c_str());
            strcpy(C_PORT[i], V6[i].c_str());
            strcpy(S_ENTROPY[i], V7[i].c_str());
            strcpy(SERVICE_DEF[i], V8[i].c_str());
            strcpy(SESSION_S[i], V9[i].c_str());
      }

init();

insert_log(DT, TIME, S_IP, C_IP, PROT, S_PORT, C_PORT,
S_ENTROPY, SERVICE_DEF, SESSION_S, size);

uninit();

return 0;
}
```

REFERENCES

1. Kanaoka, Akira, Okamoto, Eiji. (2003). Multivariate Statistical Analysis of Network Traffic for Intrusion Detection. IEEE Published paper. University of Tsukuba, Japan.

2. Zheng Zhang, Manikopoulos, C.N. (2003). The Evaluation of Feature-Extraction Algorithms in Statistical Anomaly Intrusion Detection. IEEE publication 7-8.

3. Riptech Inc., (2001). Successful Real-Time Security Monitoring, white paper.

4. Bloedorn, E., Talbot, L., Skorupka, C., Chistiansen, A., Hill, B., Tivel, J. (2001). Data Mining Applied to Intrusion Detection: MITRE Experiences. IEEE paper.

5. Bloedorn, E., Talbot, L., Skorupka, C., Chistiansen, A., Hill, B., Tivel, J. (2001). Data Mining for Network Intrusion Detection: How to Get Started. IEEE paper.

6. Lee, W., Mok, K. W. (2000). Adaptive Intrusion Detection: a Data Mining Approach. Computer Science Department, Columbia University.

7. Kumar, S. (1995). Classification and Detection of Computer Intrusions. Ph. D. Thesis. Purdue University.

8. Northcutt, S. (2000). Intrusion detection FAQ. SANS Institute. Retrieved December 09, 2003 from the World Wide Web: http://www.sans.org/resources/idfaq/network_based.php

9. Phung, M. (2000). Data Mining in Intrusion Detection. SANS Institute. Retrieved December 09, 2003 from the World

Wide Web:
http://www.sans.org/resources/idfaq/data_mining.php

10. Yee, A. (July 2003). The Intelligent IDS: Next Generation Network Intrusion Management Revealed. NFR Security Inc. Research paper.

11. Secure Tunnel. Universal SSL Wrapper. Available from the World Wide Web: http://www.stunnel.org

12. DIG White Paper (October 1995) From Data Mining to Database Marketing. Retrieved May 4, 2004 from the World Wide Web: http://www.thearling.com/text/wp9502/wp9502.htm

13. M. Lincoln Labs (1998). 1998 DARPA intrusion detection evaluation data set. Retrieved December 08, 2003 from the World Wide Web:
http://www.ll.mit.edu/IST/ideval/data/1998/1998_data_index.html