ABSTRACT

AN EXPERIMENTAL STUDY ON NETWORK INTRUSION DETECTION SYSTEMS

by
Peng FU

A signature database is the key component of an elaborate intrusion detection system. The efficiency of signature generation for an intrusion detection system is a crucial requirement because of the rapid appearance of new attacks on the World Wide Web. However, in the commercial applications, signature generation is still a manual process, which requires professional skills and heavy human effort. Knowledge Discovery and Data Mining methods may be a solution to this problem. Data Mining and Machine Learning algorithms can be applied to the network traffic databases, in order to automatically generate signatures.

The purpose of this thesis and the work related to it is to construct a feasible architecture for building a database of network traffic data. This database can then be used to generate signatures automatically. This goal is achieved using network traffic data captured on the data communication network at the New Jersey Institute of Technology(NJIT).

AN EXPERIMENTAL STUDY ON NETWORK INTRUSION DETECTION SYSTEMS

by
Peng FU

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

Department of Computer Science

August 2003

APPROVAL PAGE

AN EXPERIMENTAL STUDY ON NETWORK INTRUSION DETECTION SYSTEMS

Peng FU

8/7/2003

Dr. Jason Tsong Li Wang, Thesis Advisor — Date
Professor, Department of Computer Science, NJIT

8/11/03

Dr. Chengjun Liu, Committee Member — Date
Assistant Professor, Department of Computer Science, NJIT

Aug. 7, 03.

Dr. Michael M. Yin, Committee Member — Date
Senior Technical Staff Member, AT&T Labs

BIOGRAPHICAL SKETCH

Author: Peng FU

Degree: Master of Science

Date: August 2003

Date of Birth:

Place of Birth:

Education:

- Masters of Science, Computer Science
  New Jersey Institute of Technology, Newark, New Jersey, August 2003

- Ph.D. Candidate, Biochemistry and Molecular Biology
  University of Medicine and Dentistry of New Jersey (UMDNJ), 1999-2001

- Master of Science, Molecular Genetics and Molecular Virology
  College of Life Sciences, Nankai University, Tianjin, CHINA, 1997

- Bachelor of Science, Biochemistry
  College of Life Sciences, Nankai University, Tianjin, CHINA, 1994

Major: Computer Science

Publications:

- Deng L, de la Fuente C, Fu P, Wang L, Donnelly R, Wade JD, Lambert P, Li H, Lee CG, Kashanchi F.
  "Acetylation of HIV-1 *Tat* by *CBP/P300* Increases Transcription of Integrated HIV-1 Genome and Enhances Binding to Core Histones", in *Virology, vol. 277 No. 2, November 2000. Pages 278-295.*

- de La Fuente C, Santiago F, Chong SY, Deng L, Mayhood T, Fu P, Stein D, Denny T, Coffman F, Azimi N, Mahieux R, Kashanchi F,
  "Overexpression of *p21(waf1)* in Human *T*-cell Lymphotropic Virus Type 1 Infected Cells and Its Association with *Cyclin A/cdk2*", in *Journal of Virology, vol. 74, No. 16, August 2000. Pages 7270-7283.*

- Clark E, Santiago F, Deng L, Chong S, de La Fuente C, Wang L, Fu P, Stein D, Denny T, Lanka V, Mozafari F, Okamoto T, Kashanchi F,
"Loss of *G1/S* Checkpoint in Human Immunodeficiency Virus Type 1-Infected Cells Is Associated With a Lack of *Cyclin*-Dependent Kinase Inhibitor *p21/Waf1*", in *Journal of Virology, vol. 74, No. 11, June 2000 Pages 5040-5052.*

- FU Peng, LIU Shuhong, Chen Qimin, GENG Yunqi,
"Construction of in vitro Culture System for BIV92044 Strain", in *Acta Scientiarum Naturalium Universitatis Nankaiensis, vol. 31, No. 2, February 1998, Pages 22-27.*

- FU Peng, LIU Shuhong, Chen Qimin, GENG Yunqi,
"Cloning and Sequence Analysis of Functional Fragments of BIV92044", in *Acta Scientiarum Naturalium Universitatis Nankaiensis, vol. 31, No. 2, February 1998, Pages 96-98.*

- FU Peng GENG Yunqi, *et al. ed.*,
"*HIV and Related Viruses*", Nankai University Press, Tianjin, P. R. China, 1997.

- CHEN Qi-min, LIU Shuhong, JI Yonggang, Xue Zhihong, FU Peng, GENG Hairong, MA Ming, SUN Qing, LIANG Dong and GENG Yunqi,
"Effect of the Changes of Amino Acids on the Mature Protein *N*-Terminal Region to the Secretion of α-Amylase in *B.subtilis*", in *Acta Genetica Sinica (Domestic Edition), vol. 25, No. 3, March 1998, Pages 278-285.*

- Peng Fu, Yunqi Geng,
"Immunological Detection of BIV92044 and Cloning and Sequence Analysis of its Structural Genes", *Meeting Communication*; Chinese Society for Biochemistry, 1997 Annual Meeting; July 1997; Haikou City, Hainan Province, P. R. CHINA

To My Beloved Family

ACKNOWLEDGEMENT

First of all, I would like to thank my thesis advisor, Dr. Jason Tsong Li Wang, for his precious direction and guidance. His words always inspire thoughts and bring great complexity down to simple workable ideas. Doing research under his instruction is my honorable privilege. I will benefit from this experience throughout my life. I also would like to acknowledge and thank Dr. Chengjun Liu and Dr. Michael M. Yin, for their kindness to be my committee members, and their time spent on reviewing my research.

I would like to thank Mr. Wenping Yang, who is the System Manager at Department of Computer Science of NJIT. He kindly provided suggestions on the research topic as well as some of the facilities that were used in the research.

I want to thank Ms. Katherine Herbert, who is a Ph.D. candidate in the Data and Knowledge Engineering Lab. Her valuable advice nourishes my mind in many ways. I want to thank Mr. Sen Zhang and Mr. Huiyuan Shan, who are also Ph.D. candidates in the Data and Knowledge Engineering Lab. Their knowledge on network has greatly enriched my research.

Finally, I would like to thank my dear wife, Xing Kuang. The love, support, encouragement and spiritual happiness she brings me always keeps me going.

TABLE OF CONTENTS

Chapter **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

The emergence of the Internet has profoundly and permanently changed the shape of the world. One of the greatest advantages Internet brings is its accessibility, which brings vast and diverse information to users with just a mouse click. With this greatest advantage comes its inherent opposite side, the greatest threat. While it facilitates the usage by all the legitimate users, accessibility also provides convenience for illegal users to probe into networked systems, steal sensitive information or even disable the entire World Wide Web for days. This could happen at any moment to any computer on the Internet. When considering this problem, the biggest issues are when do the intrusions happen, where do they come from and who commits them. These problems are well known to any security professional. Since the early development of data communication protocols developers considered functionalities and efficiency much more than they did network security issues, almost all established and widely used protocols "genetically" contain insecure weakness. This makes solving the above problems much more difficult. It will be a long run for the whole world to have a safe World Wide Web.

With the dramatic growth of the Internet, these kinds of intrusions happen more and more frequently. According to CERT [1], in the Year of 1988 there were only 6 security incidents reported; in the Year of 2002, there were 82,094 incidents; and as of August 2003, there have been 76,404 (Figure 1.1). Since 1988, there have been a total of 258,867 incidents that may involve one site or even hundreds of thousands of sites. Furthermore, with the evolution of technology, the malicious computing techniques and tools have been constantly progressing. Subsequently, intrusions are more complicated and it requires more and more professional skills to find them.

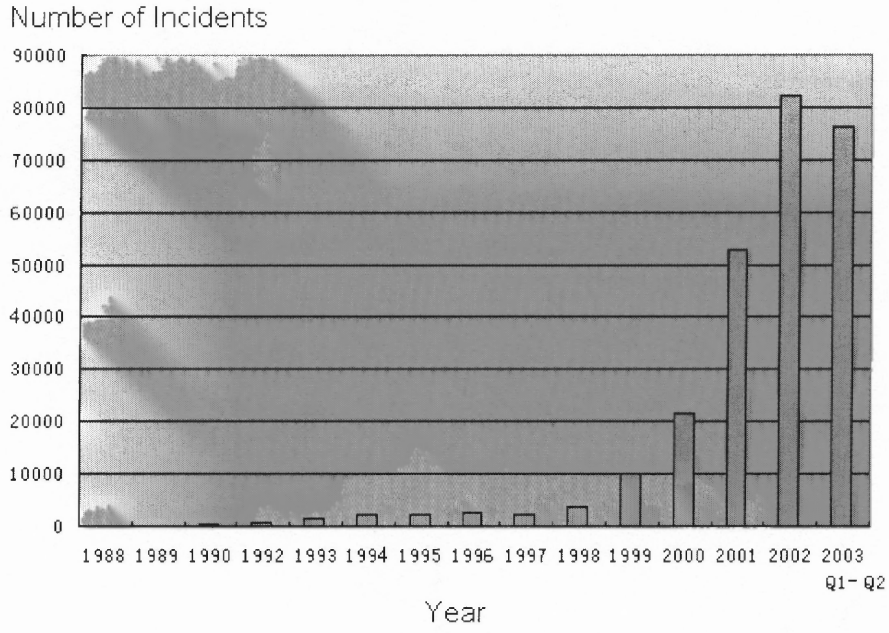

Figure 1.1 Trends of reported security incidents, data obtained from CERT[1].

This severe situation leaves us in great need of countermeasures for network intrusions. First of all, security is a policy issue that has to be always kept in mind by the people who administrate data communication networks. Without the proper regulations and management on security related matters, even the finest technologies can become useless investment. Security policies for data communication networks can also reinforce the technical strength in many ways. For example, publicly emphasizing or even exaggerating that user activity is being closely monitored by administrator may deter an intrusion attempt before it happens.

However, policy management and psychological strategies are not enough to prevent the system from being invaded by experienced attackers. Technically, anti-network intrusion tools that are capable of helping an organization or network secure its information are still needed. These tools could be used to detect an intruder, identify and block the intruders, support investigations to find out how the intruder

got in, stop the exploit that the intruder used from being used by future intruders, and provide forensic evidence to carry out legal actions.

Intrusion Detection System (IDS) is the right tool to meet all these requirements. It utilizes technologies coming from various fields of computer science to discriminate inappropriate, incorrect, or anomalous activities. Detection is not the only function an IDS has. Today's advanced IDSs have the ability to automatically and aggressively counteract intrusions.

CHAPTER 2

INTRUSION DETECTION SYSTEMS

The concept of Intrusion Detection was created by James Anderson in 1980 [2] [3]. Based on the difference between the emphases and methodologies, intrusion detection falls into two categories: anomaly detection and misuse detection.

2.1 Detection Methods

2.1.1 Anomaly Detection

In anomaly detection, observed behaviors are compared against expected normal usage profiles that are developed for individual users, groups, applications, or system resource usage. Activities that are not defined as "normal behavior" are considered anomalies[4]. Instead of defining what is "bad" for the system, anomaly detection defines what is "good", which keeps the control rules in a reasonable size. In practice, thresholds are set to defining acceptable behavior (for example, the acceptable failed logins during a certain period of time), which provides a clear line between "normal" and "anomaly". In order to cover all the major aspects of a system, normal activities for individual user and user groups have to be properly and independently profiled. This process will include analyzing and specifying allowed usage for system recourses and executables. All the users and groups are expected to comply with the system recourse assignments and privileges on executables. Activities otherwise will be treated as anomalies and subject to interruption.

2.1.2 Misuse Detection

In spite of comparing to historically "normal" usage and acting on the entire unknown, misuse detection essentially locates "bad behaviors" based on the summarized and abstracted description of known attack activities [5]. Misuse

detection is signature-based detection. A signature is a pattern that represents possible attacks, which can be found in network traffic. After careful analysis of raw network traffic data, rules are generated to identify a single event that represents a threat to system security, or a series of events that represent a prolonged attack scenario. Obviously, misuse detection requires comprehensive knowledge of a system, especially its weakness and vulnerabilities. Misuse detection could be implemented by developing expert systems, reasoning models, state transition analysis systems, or neural networks. Applying various Knowledge Discovery or Data Mining algorithms to misuse detection model may be a new promising way of making the detection fast, accurate and efficient [10].

2.2 Intrusion Detection Systems

Both anomaly detection and misuse detection can be adopted into a single architecture to form an intrusion detection system (IDS). Because of the steadily increasing number of reported computer security incidents, intrusion detection systems have gained a fast-growing market. For example, Cisco[23], ISS[24], AXENT[25] and NFR[26] all market commercial systems for intrusion detection.

Intrusion Detection System has the following functions [6]:

- Monitoring and analyzing user and system activity
- Auditing of system configurations and vulnerabilities
- Assessing the integrity of critical system and data files
- Recognizing activity patterns reflecting known attacks
- Statistically analyzing activity for abnormal patterns
- Facilitating operating-system audit-trail management, with recognition of user activity reflecting policy violations

A general model for intrusion detection system contains the following discrete components [8]:

- Event Generators. They collect events from outside the intrusion detection system protected environment and present it to other components or store it into event database both in the format of "generalized intrusion detection objects".

- Event Analyzers. They receive events from the Event Generators, analyze the events statistically or compare the events against signatures or rules. Then they provide the result to other components, or store it to even database, both in the format of "generalized intrusion detection objects".

- Event Databases. They are the event repositories for all the components of the system.

- Response Units. They take generalized intrusion detection objects gathered by other components and carry out actions against suspicious events based on the result presented by the Event Analyzers.

  All four kinds of units are logical entities. A component might be implemented as a single process on one computer or a set of distributed processes on different computers.

Based on the needs that different IDSs may exchange data, there are two on-going developments for IDS standards. One standard is "The Intrusion Detection Exchange Protocol (IDXP)", which is being developed by Intrusion Detection Working Group under the Internet Engineering Task Force. (IETF) [7]. The other standard is the "Common Intrusion Detection Framework (CIDF)" [8]. As of the date of writing of this thesis, none of the two above-mentioned frameworks is widely accepted nor accepted as a *de facto* standard.

CHAPTER 3
ESTABLISHMENT OF NETWORK TRAFFIC DATA WAREHOUSE

3.1 The Limitations of Current IDSs

The most widely used and commercially available IDSs are signature-based systems. They often come with a large set of signatures that have been identified as unique to a particular vulnerability or exploit. A signature-based system matches features observed from the network traffic stream to the signatures hand-made by experts and stored in its signature database. When a signature for an attack matches observed traffic, an alert is generated, or the event is otherwise recorded [5].

Signature-based methods have some inherent limitations. The most significant one is that a signature-based method can only detect attacks whose features have existed in the signature database. Therefore, just like anti-virus software, signature based IDSs cannot detect unknown attacks because there is no available signature. But the problem is even worse, as of September 2002, there are only 2,000 IDS signatures that have been defined, while there are more than 30,000 for anti-virus [9]. Most IDS vendors provide regular signature updates in an attempt to keep pace with the rapid appearance of new vulnerabilities and exploits. However, building up the signature database is a tedious manual work, which requires comprehensive knowledge and professional skills. Moreover it is very time consuming. This slow manual process can neither keep pace with the advancement of new attack techniques, nor help solve the severe problems that IDSs are currently having, such as failure to recognize large numbers of attack variants, high probability of generating false positive alerts and/or false negative reports and difficulty in handling overwhelming network traffic data loads.

Based on all these factors, a feasible automated or at least semi-automated mechanisms is greatly needed to respond quickly enough to new attacks and produce reliable signatures for intrusion detection system.

3.2 Data Ming and Signature Generation Framework

Data mining-based methods could be very promising candidates for automated signature generation [10]. The major advantage of these methods is that they utilize the generalization ability of data mining algorithms [11]. In order to detect new and unknown attacks, a data mining-based intrusion detection system employs machine learning and data mining algorithms on a large set of system audit data to build up signature databases.

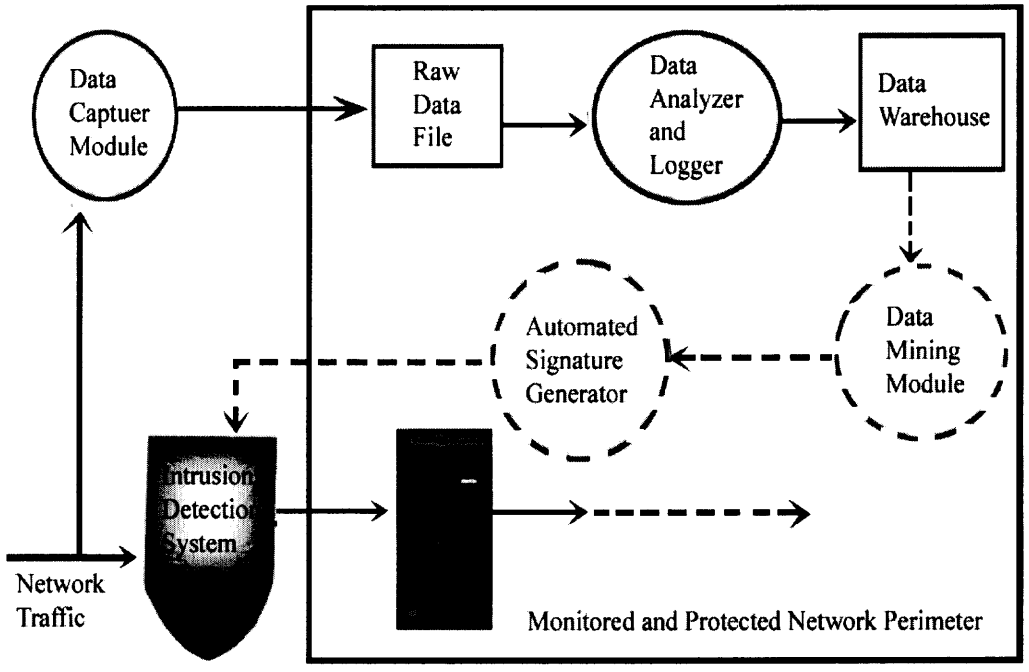

Figure 3.1 An Intrusion Detection Framework with Automated Data Analyzer and Signature Generator.

Based on the needs and the above analysis, an intrusion detection framework is proposed in this thesis as shown in Figure 3.1, the Data Capture Module, Data Analyzer and Logger Module are implemented and Data Warehouse is established

in this thesis. The Data Ming module and Automated Signature Generator will be implemented in further research works.

In Figure 3.1, the IDS is placed on the external side of a monitored and protected network, along with a Data Capture Module. The Data Capture Module captures the network traffic data and writes the data into a raw data file in the format of the original binary stream. Then the Data Analyzer will use network protocol based analysis to extract information in every data field of the packets encapsulated in the binary stream. And the Data Logger will then log the resulted data items into corresponding positions to build up a Data Warehouse. The Data Mining Module takes over here. It queries the Data Warehouse and applies appropriate Data Mining algorithms. The output is then sent to the Automated Signature Generator to produce signatures, which will ultimately enhance the performance of the Intrusion Detection System. Ideally, the whole process needs no human interference. However, in the preliminary implementations of the system, some human interaction is still necessary.

3.3 Construction of the Data Warehouse

Under the Defense Advanced Research Project Agency (DARPA [12]) and Air Force Research Laboratory (AFRL [13]) sponsorship, the Information System Technology Group at MIT Lincoln Laboratory [14] has been evaluating Intrusion Detection Systems. Their offline data sets that were used to carry out 1998 and 1999 evaluation have been widely accepted by researchers in the field of intrusion detection. The 1998 data sets are used to build up the Data Warehouse in this thesis.

The Data Warehouse consists of sets of network traffic databases, namely, training data sets for the training of Data Mining algorithms and Machine Learning mechanisms; and real data sets that the trained algorithms will work on. In these network traffic databases, traffic stream is dissected into fields defined by corre-

3.2 Data Ming and Signature Generation Framework

Data mining-based methods could be very promising candidates for automated signature generation [10]. The major advantage of these methods is that they utilize the generalization ability of data mining algorithms [11]. In order to detect new and unknown attacks, a data mining-based intrusion detection system employs machine learning and data mining algorithms on a large set of system audit data to build up signature databases.

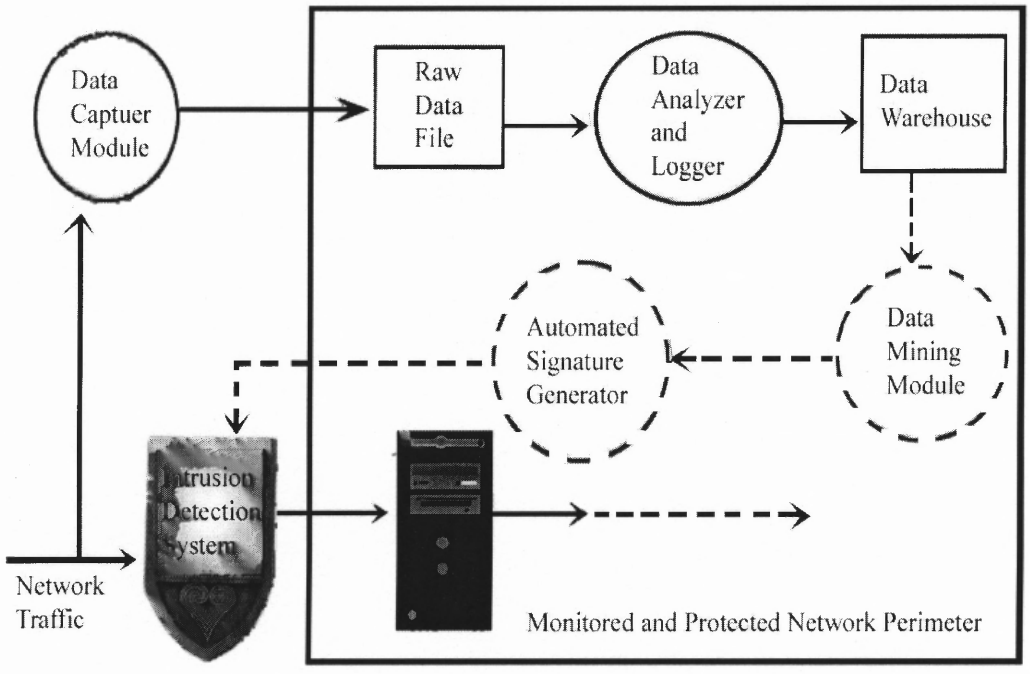

Figure 3.1 An Intrusion Detection Framework with Automated Data Analyzer and Signature Generator.

Based on the needs and the above analysis, an intrusion detection framework is proposed in this thesis as shown in Figure 3.1, the Data Capture Module, Data Analyzer and Logger Module are implemented and Data Warehouse is established

libpcap [15] and *WinPCAP* [18] Programming on UNIX/LINUX systems and Win32 systems, respectively. In the work of this thesis, MySQL 4.0 [19] is used as a cross-platform database server and MySQL Control Center 0.9.2 beta is used as client [19]. The site-specific configuration file of MySQL server is provided in Appendix B.

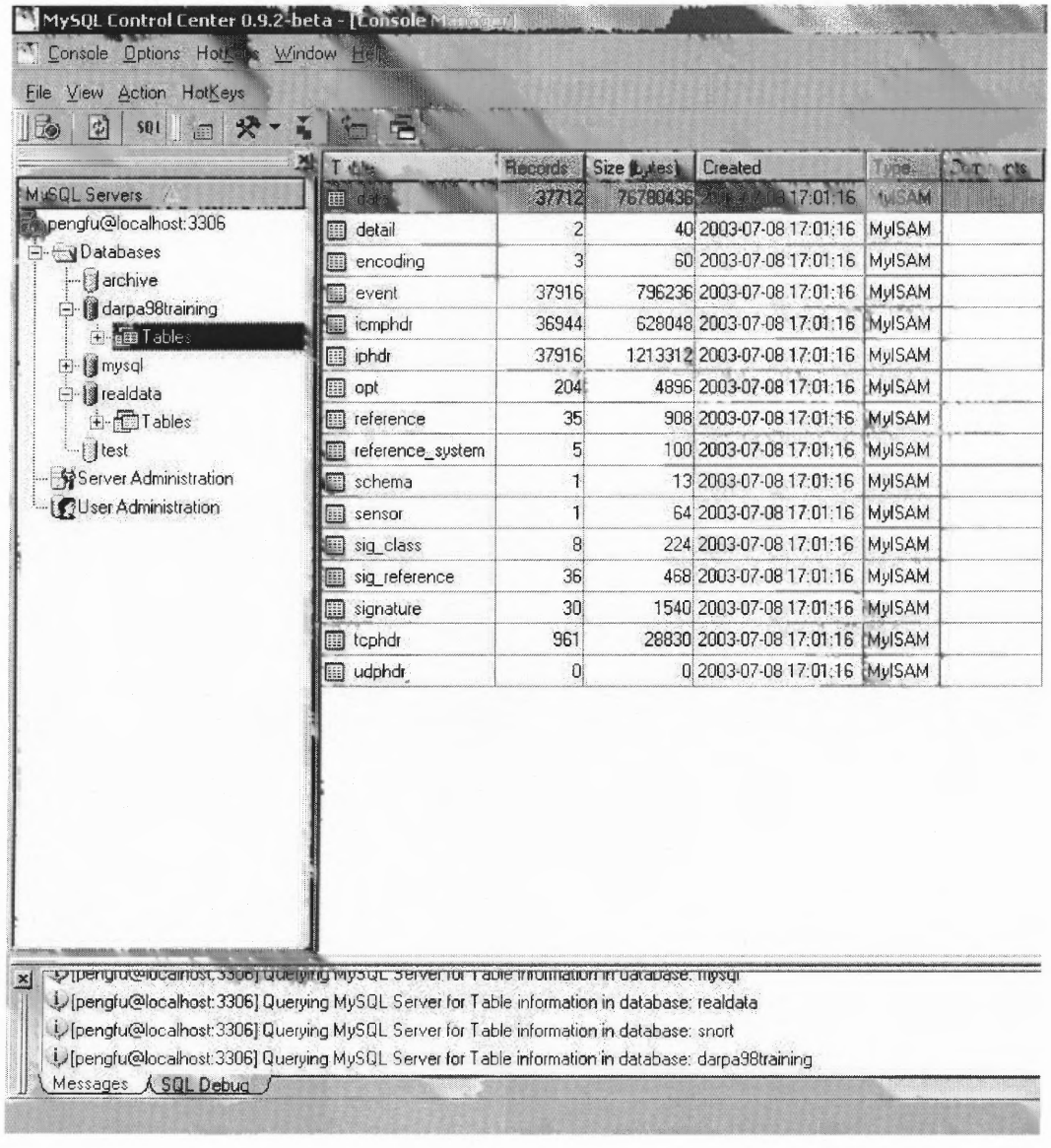

Figure 3.2 Tables for DARPA 98 Training Database, showing in MySQL Control Center.

Part of the database contents is shown in Figure 3.2 and Figure 3.3 after the establishment of the databases.

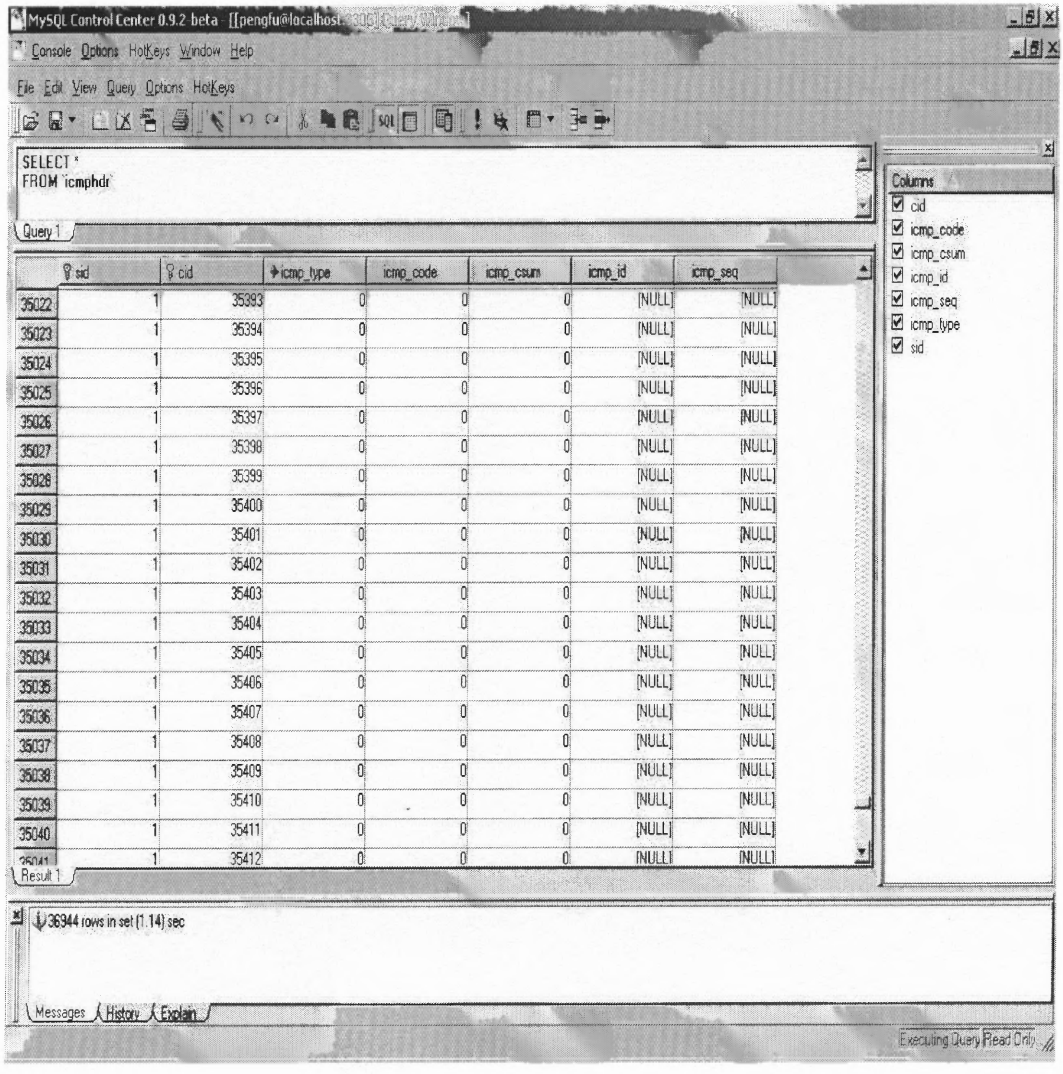

Figure 3.3 Contents of ICMP table in DARPA 98 Training Database, showing in MySQL Control Center.

3.4 Implementation on NJIT's Campus Network

In order to test the previously established procedure, all the involved modules are placed on NJIT network to work with real network traffic data. This implementation

Time	Security Type	Severity	Direction	Protocol	Dst_Host	Src_IP	Count
05/05/2003 03:09:17	Denial of Service	Major	Incoming	TCP	210.187.13.70	128.235.34.132	4
05/05/2003 03:09:18	Denial of Service	Major	Incoming	TCP	210.187.13.70	128.235.34.132	2
05/05/2003 03:09:27	Denial of Service	Major	Incoming	TCP	210.187.13.70	128.235.34.132	2
05/05/2003 03:09:38	Denial of Service	Major	Incoming	TCP	210.187.13.70	128.235.34.132	2
05/05/2003 03:10:00	Denial of Service	Major	Incoming	TCP	210.187.13.70	128.235.34.132	2
05/05/2003 03:10:52	Denial of Service	Major	Incoming	TCP	210.187.13.70	128.235.34.132	2
05/05/2003 13:12:43	Denial of Service	Major	Incoming	TCP	218.244.107.200	128.235.34.132	2
05/05/2003 13:12:51	Denial of Service	Major	Incoming	TCP	218.244.107.200	128.235.34.132	2
05/05/2003 13:14:19	Denial of Service	Major	Incoming	TCP	218.244.107.200	128.235.34.132	2
05/05/2003 13:15:53	Denial of Service	Major	Incoming	TCP	218.244.107.200	128.235.34.132	2
05/06/2003 01:06:05	Denial of Service	Major	Incoming	TCP	128.248.170.138	128.235.34.132	2
05/06/2003 01:06:10	Denial of Service	Major	Incoming	TCP	128.248.170.138	128.235.34.132	2
05/06/2003 01:06:15	Denial of Service	Major	Incoming	TCP	128.248.170.138	128.235.34.132	2
05/06/2003 01:06:31	Denial of Service	Major	Incoming	TCP	128.248.170.138	128.235.34.132	2
05/06/2003 01:06:52	Denial of Service	Major	Incoming	TCP	128.248.170.138	128.235.34.132	2
05/06/2003 01:07:45	Denial of Service	Major	Incoming	TCP	128.248.170.138	128.235.34.132	2
05/06/2003 01:09:13	Denial of Service	Major	Incoming	TCP	128.248.170.138	128.235.34.132	2
05/06/2003 03:13:46	Denial of Service	Major	Incoming	TCP	211.156.96.11	128.235.34.132	2
05/06/2003 03:13:47	Denial of Service	Major	Incoming	TCP	211.156.96.11	128.235.34.132	4
05/06/2003 03:14:02	Denial of Service	Major	Incoming	TCP	211.156.96.11	128.235.34.132	2
05/06/2003 03:14:12	Denial of Service	Major	Incoming	TCP	211.156.96.11	128.235.34.132	2
05/06/2003 03:14:37	Denial of Service	Major	Incoming	TCP	211.156.96.11	128.235.34.132	2
05/06/2003 03:16:59	Denial of Service	Major	Incoming	TCP	211.156.96.11	128.235.34.132	2
05/06/2003 14:34:21	Denial of Service	Major	Incoming	TCP	217.209.63.102	128.235.34.132	4
05/06/2003 14:34:23	Denial of Service	Major	Incoming	TCP	217.209.63.102	128.235.34.132	2
05/06/2003 14:34:29	Denial of Service	Major	Incoming	TCP	217.209.63.102	128.235.34.132	2
05/06/2003 14:34:41	Denial of Service	Major	Incoming	TCP	217.209.63.102	128.235.34.132	2
05/06/2003 14:35:05	Denial of Service	Major	Incoming	TCP	217.209.63.102	128.235.34.132	2

Figure 3.4 Some of the captured network traffic data that are labeled as DoS attacks.

is conducted to comply with the framework proposed in Figure 3.1, with only one exception that the data capture module is placed inside the NJIT campus network to capture traffic data going through a certain domain, instead of the whole campus network.

During capturing, the network traffic is also monitored by a third-party software to calibrate the data labeling. Figure 3.4 shows part of the labeling result.

After the capturing and labeling, the traffic stream is reconstructed by Ethereal [20]. Figure 3.5 shows this reconstruction, in which highlighted frame is previously labeled as DoS from `128.235.34.132` to `128.248.170.138`.

Under the same database structure used for the training set, the DARPA 98 Datasets, the captured network traffic data are logged into the database called

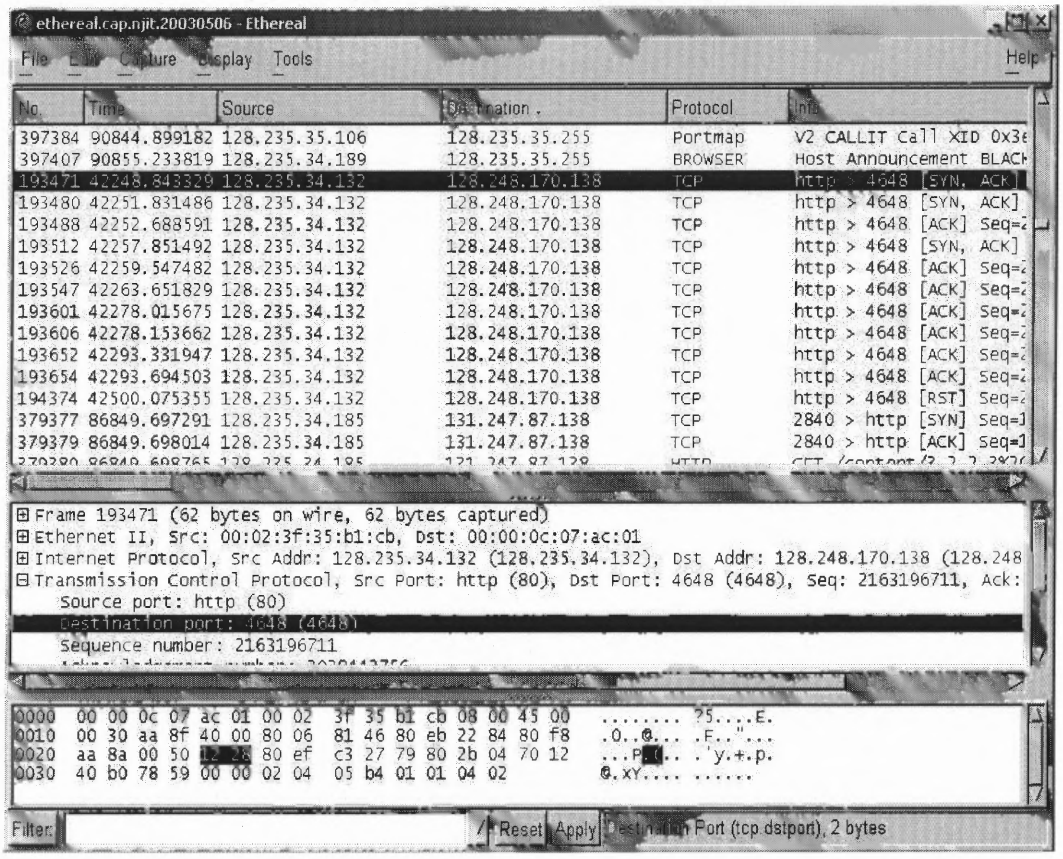

Figure 3.5 Reconstructed network traffic by Ethereal[20] showing real DoS attacks.

"realdata". In this database, the targeted traffic information can be retrieved in many ways. For example, by carefully comparing the TCP ports and time stamps, all the traffic between host `128.235.34.132` and host `128.248.170.138` can be found, weather it is normal or malicious, as shown in Figure 3.6. These databases now contain valuable information to feed data mining algorithms in further research works.

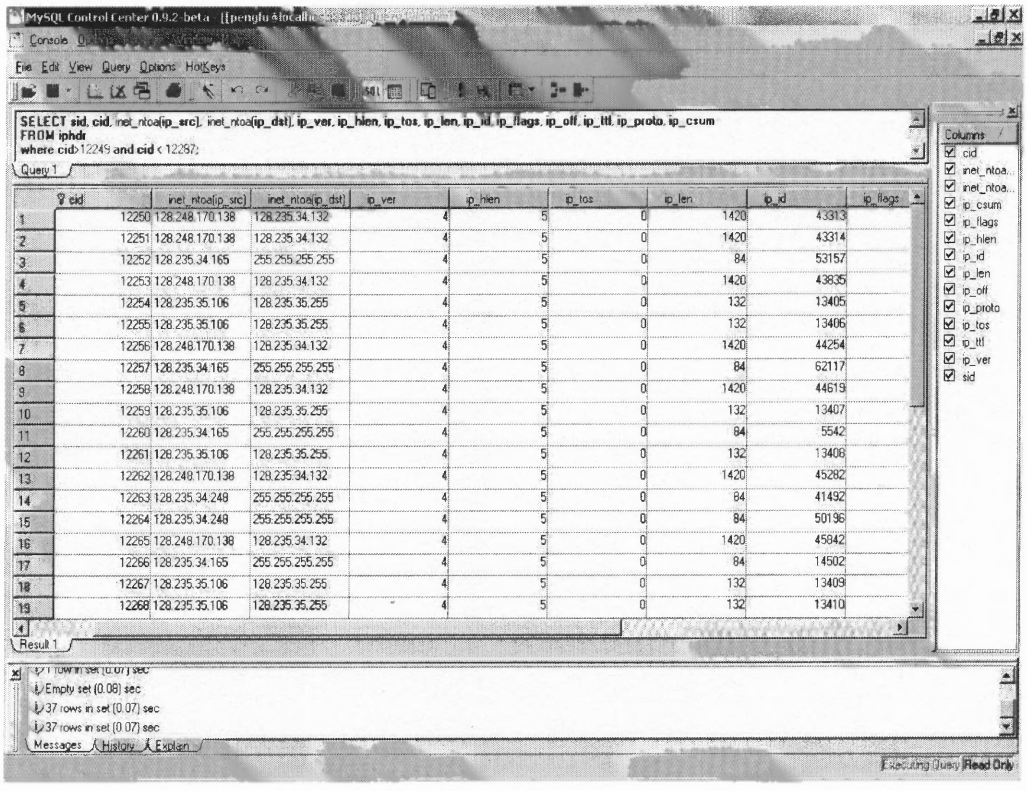

Figure 3.6 Real DoS attack packets between host `128.235.34.132` and host `128.248.170.138` in the NJIT Network Traffic Database.

CHAPTER 4

CONCLUSIONS AND FUTURE WORK

In this research, a workable preliminary procedure has been established to build up the network traffic data warehouse, which thus forms the basis of the automated intrusion detection signature generator.

It has to be pointed out that the data analyzer and logger used in this research is still very crude. It only dissects IP, TCP, UDP and ICMP data streams. More protocol analysis modules need to be added. The database schema also needs improvement for this matter. It will be great news for researchers in the field of Intrusion Detection if Ethereal is rendered database connectivity, given the fact that Ethereal can dissect more 200 protocols as of the time of writing this thesis [20].

Fighting against network intrusion is such a charming yet extremely challenging job that it demands prolonged enthusiasm and continuous hard working. It is obviously true that the more people pay attention to security issues, the safer the Internet would be.

APPENDIX A

MAJOR NETWORK PROTOCOLS

A.1 Internet Protocol

Figure A.1 shows an Internet Protocol version 4 packet is embedded in an Ethernet Frame, which is indicated in the Frame Type field with a hexadecimal value 0080. In the IP packet, IP header occupies the first 20 bytes of space (top five rows in the figure). There have been some changes in the meaning of data fields since the Internet Protocol was established. For example, Time to Live now means "how many hops (routers) the packet can pass before it expires (dropped by the system that receive it). Protocol field is a very informative; it implies what kind of data is embedded after IP header. A decimal value of 17 means it is UDP; 6, TCP and 1, ICMP.

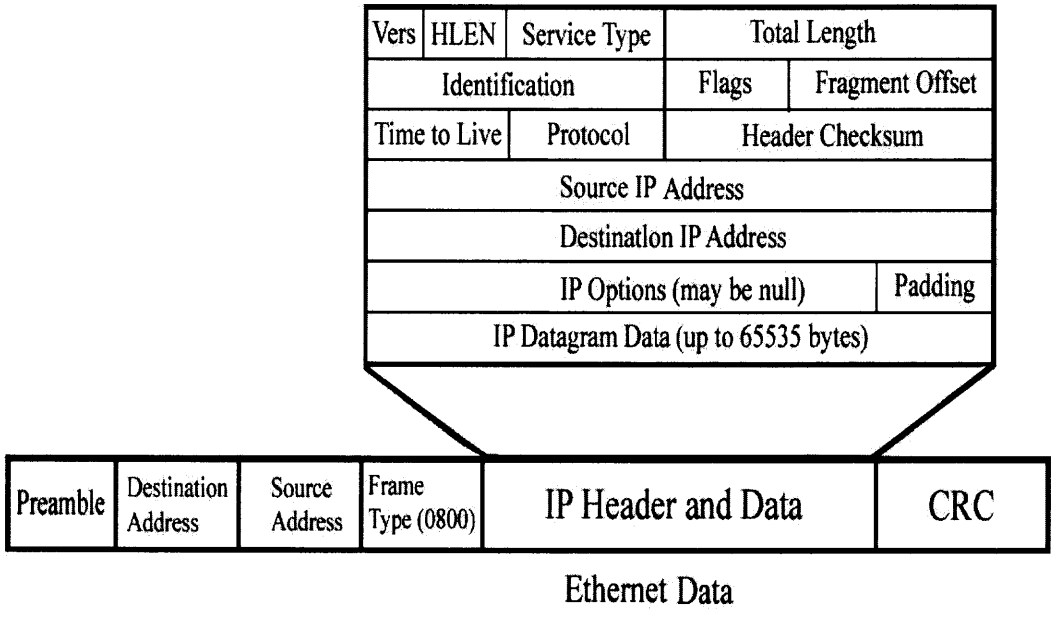

Figure A.1 IP packet format and ethernet packet format, IP packet is encapsulated in an ethernet packet.

A.2 User Datagram Protocol

User Datagram Protocol (UDP) is a connectionless, unreliable transport service. It does not provide mechanism to ensure connection. It does not acknowledge. It does not re-order the received packets. It may lose or duplicate data packets during transmission. Further more, it has no error messaging functions to inform the upper layer applications or to request a re-send from the sender. It is the duty of the application program, which employs UPD at transport-layer, to ensure the quality of service.

Even though having these unreliable features, UDP is still widely used by many applications such as IP phone applications and online multimedia services that demand speed more than stability. UDP is faster than TCP because it has much less overhead and simpler procedures to transfer data. An embedded UDP packet is shown in Figure A.2.

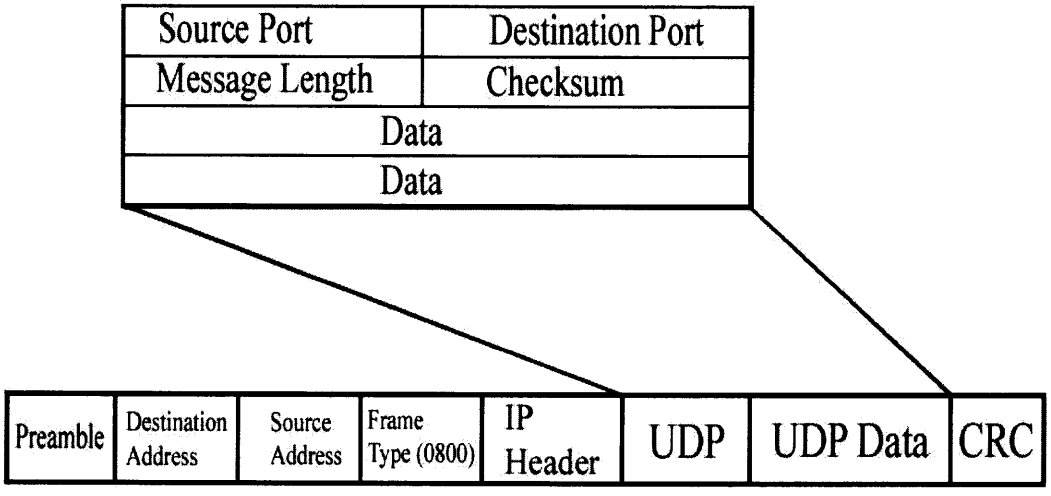

Figure A.2 UDP packet format, UDP packet is encapsulated in an IP packet, which is encapsulated in an ethernet packet.

A.3 Transmission Control Protocol

Transmission Control Protocol (TCP) is a reliable connection-oriented protocol employed by many applications that require stable connections between two hosts to ensure correct data transfer. TCP guarantees the reliability by the following mechanisms:

- TCP uses unique sequence numbers to identify the data sent to destinations.
- TCP uses acknowledgement numbers to inform the sender which exact data has been received. Acknowledgement numbers are generated by adding 1 to the corresponding sequence number.
- TCP uses three-way handshake procedure to establish a exclusive unicast connection between two hosts.

Figure A.3 shows the organization of a TCP packet. *SYN* flag is used at startup to establish the connection. *FIN* flag is used to terminate the existing connection.

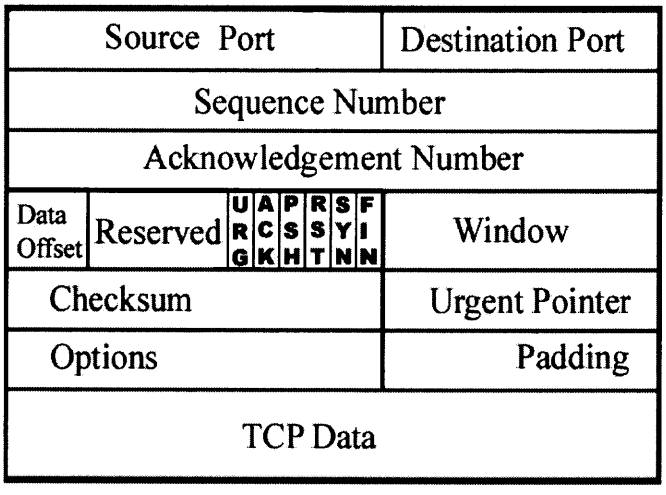

Figure A.3 TCP packet format, which is encapsulated in the same way us UDP is shown in Figure A.2.

APPENDIX B

SITE SPECIFIC CONFIGURATIONS

B.1 MySQL Configuration

MySQL 4.0 configuration used in this thesis is shown in Figure B.1. In order to avoid system configuration conflicts with Windows 2000, "`localhost`" is used as the bind-address, instead of the real IP address of the server.

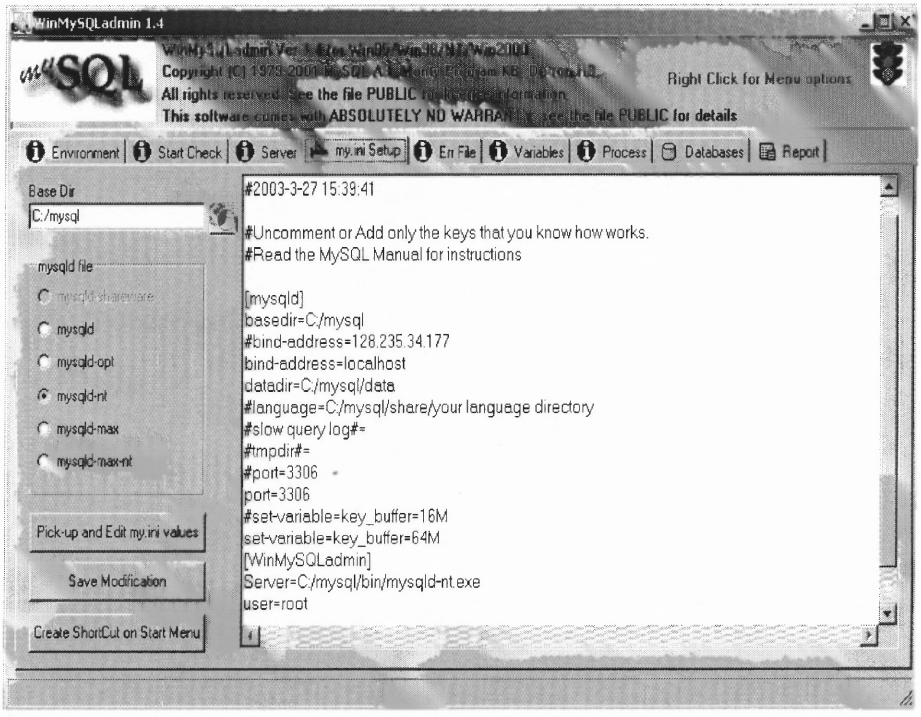

Figure B.1 Local host MySQL server configuration for MySQL 4.0 running on Windows 2000 Professional.

B.2 Snort Configuration

The snort configuration file is included in below. It is modified according to the default setting of Snort 2.0. This configuration has been tested on Windows 2000 with MySQL 4.0 in the given environment.

```
##################################
# The Snort Configuration File
##################################


var HOME_NET any

var EXTERNAL_NET any

# List of DNS servers
var DNS_SERVERS $HOME_NET

# List of SMTP servers
var SMTP_SERVERS $HOME_NET

# List of web servers
var HTTP_SERVERS $HOME_NET

# List of sql servers
var SQL_SERVERS $HOME_NET

# List of telnet servers
var TELNET_SERVERS $HOME_NET

# Ports you run web servers on
var HTTP_PORTS 80

# Ports you want to look for SHELLCODE on.
var SHELLCODE_PORTS !80

# Ports you do oracle attacks on
var ORACLE_PORTS 1521


# Path to the rules files
```

```
var RULE_PATH c:/snort/rules


# Configuration for preprocessors

preprocessor frag2

preprocessor stream4: detect_scans, disable_evasion_alerts

preprocessor stream4_reassemble

preprocessor http_decode: 80 unicode iis_alt_unicode double_encode
                          iis_flip_slash full_whitespace

preprocessor rpc_decode: 111 32771

preprocessor bo: -nobrute

preprocessor telnet_decode

preprocessor conversation: allowed_ip_protocols all, timeout 60,
                           max_conversations 32000

# The output plugin used for MySQL database
output database: log, mysql, user=pengfu password=12345678
                             dbname=realdata host=localhost
                             port=3306 sensor_name=<nobody>


# Include classification & priority settings
include c:\snort\etc\classification.config


# Include reference systems
include c:\snort\etc\reference.config


#
# Include rule sets
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
```

```
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules

include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-php.rules

include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/snmp.rules

include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/pop3.rules
include $RULE_PATH/pop2.rules

include $RULE_PATH/nntp.rules
include $RULE_PATH/other-ids.rules
include $RULE_PATH/experimental.rules
include $RULE_PATH/local.rules
```

APPENDIX C

CODE SAMPLES

The code listed here contains two parts. Part one is a basic network packet dumper, and part two is a basic data logger engine. Both parts are written in C Language. Based on the assumption that users of the code have basic knowledge about C, network, and MySQL C API, only the core routines are shown for better understanding, all the other unnecessary details are omitted. If any problem is encountered, refer to Reference [15], [18] and [19] for a more comprehensive description.

C.1 A Basic Network Traffic Dumper

```
/* Copyright information coming with LibPcap library
 *
 * Copyright (c) 1993, 1994, 1995, 1996, 1997
 *    The Regents of the University of California.  All rights
 *    reserved.
 *
 * Redistribution and use in source and binary forms, with or
 * without modification, are permitted provided that the following
 * conditions are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above
 *    copyright notice, this list of conditions and the following
 *    disclaimer in the documentation and/or other materials
 *    provided with the distribution.
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgement:
 *    This product includes software developed by the Computer
 *    Systems Engineering Group at Lawrence Berkeley Laboratory.
 * 4. Neither the name of the University nor of the Laboratory
 *    may be used to endorse or promote products derived from
 *    this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS
```

```
 * ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
 * BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO
 * EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY
 * DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
 * AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
 * THE POSSIBILITY OF SUCH DAMAGE.
 *
 */

........ /*necessary "include"s*/

#include <pcap.h> /*the libpcap library header*/
#include <sys/socket.h>
#include <netinet/in.h>

......../*necessary declarations*/


int main(int argc, char* argv){


    pcap_t *packet;  /*the packet capture descriptor*/
    pcap_dumper_t * _datafile; /*pointer to he datafile that
                                 the data will be written into*/
    char interface_name[16]; /*name of the network interface
                               on the host*/
    char errbuf[PCAP_ERRBUF_SIZE]; /*buffer for error messages*/
    int snaplen; /*maximum number of bytes to capture in a packet*/
    int promisc; /*flag to determine if the interface is
                   in promiscuous mode or not*/
    int to_ms; /*read timeout in milliseconds*/
    char datafile[32]; /*the datafile that captured packets
                         will be written into*/
    int count; /*number of packets that are designated to capture*/
    int packetnumber; /*number of packets read
                        during a live capture*/
```

```
    /*First, pcap looks up the network interfaces
      on which it will capture packets*/
    if(!(interface_name=pcap_lookupdev(errbuf))){
        fprintf(stderr, "Error Message, %s\n",
                interface_name, errbuf);
        exit(1);
    }


    /*Open the network device interface to capture packets*/
    if(!(packet=pcap_open_live(interface_name, snaplen, promisc,
                               to_ms, errbuf))){
        fprintf(stderr, "Cannot open network interface %s: %s\n",
                interface_name, errbuf);
        exit(1);
    }


    /*Open the dump device to write captured packet
      data into a datafile*/
    if((_datafile=pcap_dump_open(packet, datafile))==NULL){
        fprintf(stderr, "cannot open datafile!\n")
        exit(1);
    }

    /*pcap_dispatch() is used to collect and process packets
      on the openned interface. Then it passes the collected
      packets to pcap_dump(), which will out the packets to
      the saved datafile. pcap_dispatch() will return when
      the number of count packets have been collected. instead
      of using pcap_dispactch(), we can also use pcap_loop()
      in the similar way.*/
    if((packetnumber=pcap_dispatch(packet, count, &pcap_dump,
                                   (u_char *)_datafile))<0){
        fprintf(stderr, "cannot capture packets on interface %s:",
                interface_name);
        exit(0);
    }

    pcap_dump_close(_datafile); /*saved data file is closed*/
    pcap_close(packet); /*packet descriptor openned by
                          pcap_open_live() is closed*/

}
```

C.2 A Basic Network Traffic Data Logger Engine

```
/*=============================================================
* The same copyright information as listed above
* ============================================================
* The following is a sample data logger with connectivity to
* MySQL. It reads the saved data file previously dumpped by the
* data capture engine, it then logs into the MySQL database
* server running on localhost, and then inserts the data items
* into relavent tables.
* ============================================================*/

........ /*necessary "include"s*/

#include <pcap.h> /*the libpcap library header*/
#include <sys/socket.h>
#include <netinet/in.h>

#include <mysql.h> /*MySQL C API headers*/
#include <my_global.h> /* MySQL C API header*/

......../*necessary declarations*/


int main(int argc, char *argv ){


    pcap_t *packet;  /*the packet capture descriptor*/
    pcap_dumper_t * _datafile; /*pointer to he datafile that
                                 the data will be written into*/
    char interface_name[16]; /*name of the network
                               interface on the host*/
    char errbuf[PCAP_ERRBUF_SIZE]; /*buffer for error messages*/
    int snaplen; /*maximum number of bytes to capture in a packet*/
    int promisc; /*flag to determine if the interface
                   is in promiscuous mode or not*/
    int to_ms; /*read timeout in milliseconds*/
    char datafile[32]; /*the datafile that captured
                         packets will be written into*/
    int count; /*number of packets that are designated to capture*/
    int packetnumber; /*number of packets read*/
    struct pcap_pkthdr *packet_header; /*packet header*/
    u_char* packet_data;  /*pointer to packet data*/
```

```
char* host_name = "localhost"; /*server host (default==localhost)*/
char* user_name = "pengfu"; /*user name (default == login name)*/
char* password = "12345678"; /*password(default == none)*/
unsigned int port_num = 3306; /*default mqsql port number*/
char* socket_name = NULL; /*socket name (use build-in value)*/
char* db_name = "realdata"; /*database name (default == none)*/
unsign int flags = 0; /*connection flags
                        (0 when no connection option chosen)*/

MYSQL *connection; /*pointer to connection handler*/

char* query;


/* prototype function that handles the operations on MySQL
   database. mysql_db() also serves as the callback function
   called by pcap_loop()*/
void mysql_db(MYSQL *, u_char *);


/*prototype function that handles the insertion on MySQL
  database. it returns 0 when succeeds, otherwise fails*/
int myInsert(char *, u_char *);


/*pcap_open_offline() opens a saved file, presents it
  for reading if pcap_open_offline()fails, it returns NULL*/
if((packet=pcap_open_offline(argv[1], errbuf))==NULL){
    fprintf("cannot open file %s\n", argv[1]);
    exit(1);
}

/*when pcap_loop() finds packets (returns positive integer)
  or there is no error occuring (returns zero), packet data is
  inserted into the relational database*/
while((packetnumber=pcap_loop(packet, count, &mysql_db, NULL))>=0){


    /*pcap_next() reads the next packet
      and returns a pointer to data in that packet*/
    if((packet_data=pcap_next(packert, packet_header))==NULL){
        fprintf("Error: %s\n", pcap_geterr(packet));
        exit(1);
```

```
    }

    mysql_db(connection, packet_data){

        /*initialize the connection handler*/
        if((connection = mysql_init())==NULL){
            fprintf(stderr, "fail to initialize the connection\n");
        };


        /*this function connects to the mysql server.*/
        if((mysql_real_connect(connection, host_name, user_name,
                               password, db_name, port_name,
                               socket_name, flags))==NULL){
            fprintf(stderr, "fail to connect to mysql server:\n%s",
                    mysql_error(connection));
            mysql_close(connection);
            exit(1);

        }


        /*This is a customly defined insert function
          that inserts packet data into database*/
        if(!myInsert(query, packet_data)){
            fprintf(stderr, "insertion failed\n%s",
                    mysql_error(connection));

            exit(1);
        }
    };
}


/*if other query are demanded, the following function
  can be used to conduct the query.
  however, mysql_real_query() must be used here because
  captured data is in binary format, which contains '\0'
  character.*/
if(!mysql_real_query(connection, query, strlen(query))){

    fprintf(stderr, "query failed \n%s", mysql_error(connection));
}
```

```
    /*connection to mysql server is closed*/
      mysql_close(connection);

    /*packet descriptor openned by pcap_open_offline() is closed*/
    pcap_close(packet);

    return 0;
}
```

REFERENCES

1. CERT/CC Statistics 1988-2003
   http://www.cert.org/stats/cert_stats.html
   (January - July, 2003).

2. Intrusion Detection FAQ, version 1.80, updated June 12, 2003
   http://www.sans.org/resources/idfaq/
   (January - July, 2003).

3. Brief History of IDS
   http://www.chinaitlab.com/www/news/article_show.asp?id=2507
   (January - July, 2003).

4. Intrusion detection systems: Defining Protocol Anomaly Detection
   http://www.zdnetindia.com/biztech/ebusiness/whitepapers/stories/79205.html
   (January - July, 2003).

5. Stephen Northcutt, Lenny Zeltser, Scott Winters, Karen Kent Frederick and Ronald W. Ritchey,
   Inside Network Perimeter Security (2003)
   New Riders Publishing, ISBN 0-73571-232-8.

6. ICSALabs,
   An Introduction to Intrusion Detection and Assessment
   http://www.icsalabs.com/html/communities/ids/whitepaper/index.shtml
   (January - July, 2003).

7. B. Feinstein, G. Matthews and J. White,
   Intrusion Detection Exchange Format - Internet Draft, October 22, 2002
   http://www.ietf.org/internet-drafts/draft-ietf-idwg-beep-idxp-07.txt
   (January - July, 2003).

8. Phil Porras, Dan Schnackenberg, Stuart Staniford-Chen, Maureen Stillman and Felix Wu,
   The Common Intrusion Detection Framework Architecture
   http://www.isi.edu/gost/cidf/
   (January - July, 2003).

9. Stephen Northcutt and Judy Novak,
   Network Intrusion Detection 3rd ed.
   New Riders ISBN 0-73571-265-4.

10. Wenke Lee,
    Applying Data Mining to Intrusion Detection: the Quest for Automation, Efficiency, and Credibility.
    SIGKDD Explorations December 2002. vol. 4, Issue 2.

11. Jiawei Han and Micheline Kamber,
Data Mining: Concepts and Techniques
Academic Press and Morgan Kaufmann Publishers ISBN 1-55860-489-8.

12. Defense Advanced Research Projects Agency (DARPA)
http://www.darpa.mil/
(January - July, 2003).

13. Air Force Research Labs
http://www.rl.af.mil/
(January - July, 2003).

14. The Information Systems Technology Group (IST) of MIT Lincoln Laboratory
http://www.ll.mit.edu/IST/ideval/index.html
(January - July, 2003).

15. *tcpdump* and *libpcap* Public Repository
http://www.tcpdump.org
(January - July, 2003).

16. *Snort*^{TM}, The Open Source Network Intrusion Detection System
http://www.snort.org
(January - July, 2003).

17. Jay Beale, James C. Foster and Jeffrey Posluns,
Snort 2.0 Intrusion Detection
Syngress ISBN 1-931836-74-4.

18. WinPcap: The Free Packet Capture Architecture for Windows
http://winpcap.polito.it
(January - July, 2003).

19. *MySQL AB*, The World's Most Popular Open Source Database
http://www.mysql.com/
(January - July, 2003).

20. Ethereal, Sniffing the Glue That Holds the Internet Together
http://www.ethereal.com
(January - July, 2003).

21. SUN Microsystems Inc.
http://www.sun.com
(January - July, 2003).

22. *System Administration Guide: Security Services*, Solaris 9, April 2003
http://docs.sun.com/db/doc/816-4883?q=bsm
(January - July, 2003).

23. Cisco Sytems,
http://www.cisco.com
(January - July, 2003).

24. Internet Security Systems^{TM},
http://www.iss.net
(January - July, 2003).

25. AXENT Technologies (merged with Symantec on July 27, 2000),
http://www.symantec.com
(January - July, 2003).

26. NFR Security,
http://www.nfr.com
(January - July, 2003).