2004

Analysis of e-mail attachment signatures for potential use by intrusion detection systems

Archis Vijay Raje
West Virginia University

Analysis of E-mail Attachment Signatures For Potential use by Intrusion Detection Systems

Archis Vijay Raje

Thesis submitted to the College of Engineering and Mineral Resources

at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Roy S. Nutter, Jr., Ph.D., Chairman
Bojan Cukic, Ph.D.
Katerina Goseva-Popstajanova, Ph.D.

Lane Department of Computer Science and Electrical Engineering
Morgantown, West Virginia
2004

Keywords: Network Security, Intrusion Detection, Packet data classification.

Abstract

Analysis of E-mail Attachment Signatures for Potential use by Intrusion Detection Systems

Archis Vijay Raje

Today, an Intrusion Detection System (IDS) is almost a necessity. An IDS can protect a network from external threats, hide system vulnerabilities, prevent misuse, act as a second layer of defense to the firewall, provide evidence in case of an attack. The effectiveness of an IDS depends on the number of parameters it can monitor to report malicious activity. Due to highly variable nature of network traffic, current Intrusion Detection Systems monitor packet headers only.

This thesis investigates the possibility of monitoring network packet data as one of the parameters for IDS. This is done by finding a pattern in each type of payload. This pattern might then be related to the application to which it belongs. Based on this pattern, an attempt is made to determine if there is a difference in packets generated by different applications.

This investigation limits the classification to packets generated by E-mail attachments. Frequency of characters in packet data is used to generate a pattern. This frequency is limited to Base64 alphabets since E-mail has to conform to RFC 822 standards. Based on these patterns, certain E-mail attachments can be related to the source type of the attached file.

This thesis is dedicated to my parents

Vijay and Vrushali Raje

and my brother Harshal, whose love and support made this possible.

Acknowledgment

I would like to thank my Advisor, Dr. Roy S. Nutter, Jr for his invaluable support and encouragement which helped shape this research. His guidance, especially at times when I was drifting away from the topic was precious.

I would also like to thank Dr. Bojan Cukic and Dr. Katerina Goseva-Popstajanova for taking time to aid in guiding me as committee members.

Finally, I'd like to thank all my friends for their support and understanding during the course of my studies at the University.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent developments in information technology have increased our day-to-day dependence on computers and the Internet. The increased transmission such as personal financial transactions and sensitive corporate information attracts malicious users to exploit shortfalls in a system's security. As people realized that a firewall was not sufficient to keep the intruders out, new techniques were developed. An Intrusion Detection System (IDS) is one such attempt.

1.1 Intrusion Detection

The main function of an IDS is to detect and isolate incorrect and anomalous events. Fast detection and isolation can minimize the damage in case of intrusion. Threats to a company's network may be due to malicious activity or vulnerabilities in computer systems. Because of the constantly evolving technology, it becomes very difficult to manage such threats. An Intrusion Detection System consists of a set of tools which help monitor a network's status and security. In addition to security policy, system auditing, router security, firewalls and incident response plans, IDS adds to the layered defense of a network. While a firewall merely allows or blocks traffic based on certain rules, an IDS tries to isolate specific intrusive traffic that is allowed by the firewall. Each layer is important for the overall security of a network. Failure of one layer may affect the overall

defense. Figure 1 shows the location of an Intrusion Detection System within a network. System vulnerability once past the firewall provides another method for an intruder to enter the network. Many of these vulnerabilities are either not yet identified and their impact on system security in unknown. Despite efforts to test a new product for bugs, there are always some bugs that escape scrutiny and may prove to be harmful. An IDS may detect an intruder and identify him. An IDS log can also help support an investigation of how the intrusion took place. Assessment can lead to steps being taken to prevent future events. Such investigations can also expose vulnerabilities of the network leading to improved security. An IDS is no longer a luxury but a necessity.

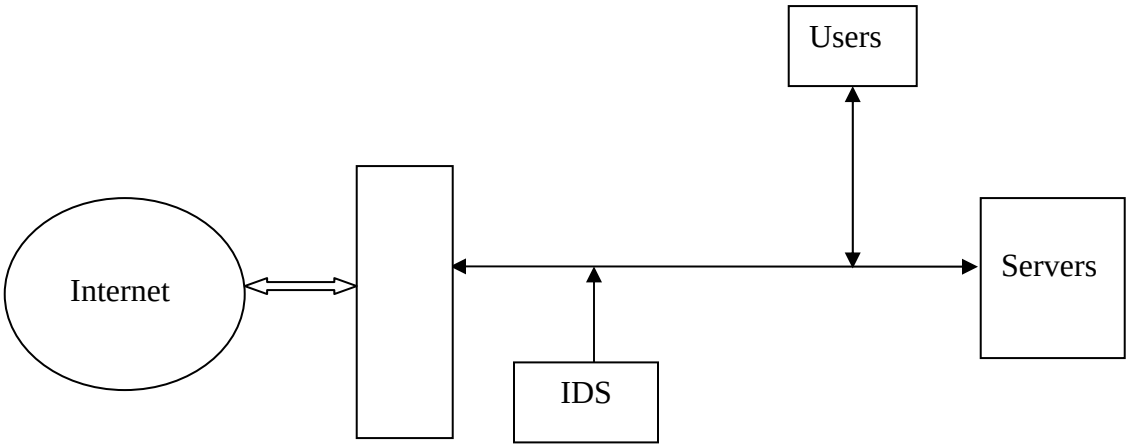


Figure 1. Position of IDS in a Network

1.2 Current IDS techniques

Considerable effort has been expended by the industry to develop a perfect Intrusion Detection System. There have been many techniques employed to detect intrusions. Each technique has its own pros and cons. Classification of ID systems is done [1] in the following ways.

Based on data source:

1. Host based
2. Network based

Based on model of intrusion:

1. Anomaly detection model
2. Misuse detection model

Each system has its own method for data collection and classification [2]. A host based system collects data from a single source to detect intrusions. This type of a system is useful in monitoring trusted insider attacks. It can also monitor system calls and can thus be more efficient, since it can relate the network activity to processes running on the machine. A network based IDS can monitor an attack which spans multiple hosts. Thus each system has its advantages and disadvantages.

A network based anomaly IDS uses observed activity on the network to create a baseline for standard behavior of a network. Given a certain margin for error, the system triggers an alarm if there is significant deviation from this set activity saying an anomaly exists in the behavior of the network. The system needs to be *trained* on the network before it can

perform the required task. This training provides a baseline for the systems on that network. Anomalies can then be detected using this baseline as a reference. These systems in addition can be adaptive as they can adapt to changes in normal behavior over a period of time.

Network misuse detection checks for activity, which is defined as bad. A description of undesired usage is made and the network traffic is compared against it. The signatures or rules correspond to a specific attack or known system vulnerability. A possible attack is detected if this known signature is found in the payload of a packet [3]. As the name suggests, this type of IDS checks mainly for internal misuse. A majority of commonly occurring intrusions can be successfully detected using pattern matching techniques. Although effective to a certain extent, misuse detection is not suitable for the remainder of intrusions [4]. Misuse detection for example will not identify newer types of attacks. Only when a known signature is available, can such systems detect an attack. The advantage of this technique is that it is fully operational the moment it is deployed.

Anomaly based systems take some time after deployment to learn the network traffic before they can detect any intrusions. This is one step, which cannot be ignored, as the system is specific to a given network. Even for two networks having identical configurations, traffic can have large variations. Hence the system requires a lot of tuning before it can be deployed effectively.

Anomalies in network traffic are due to typically four reasons [5]. The first is; exploiting bugs in system software, since bugs relating to normal activity will be easily detected and fixed. The second anomaly is due to carelessness of an attacker. A visible probe by a careless attacker who is seeking information which normal users would already know is the third reason. The fourth reason is deliberate insertion of unusual data to confuse the IDS.

Effectiveness of an Intrusion Detection System also depends on the method of collecting data [2]. Delayed detection and incorrect and incomplete data can all hamper the performance of IDS.

1.3 Need for research

Intrusion Detection techniques are constantly evolving to keep pace with advances made by attackers. Discoveries of new vulnerabilities also pressure this research. Although various methods of detecting intrusion are employed each has its advantages and disadvantages.

There are many challenges faced by today's ID systems. Apart from technology specific drawbacks, the common ones are

1. Variants: an IDS must detect variations in attacks. While anomaly based systems may detect such variations, misuse detection systems need a signature before it can pinpoint certain activity as aberrant.
2. False Positives: this is a case when legal traffic on a network is deemed as an intrusion attempt. While an IDS plays a role of a sentry, it must not affect legitimate users of the network.

3. False Negatives are obtained when an IDS fails to detect an attack. A misuse detection system can give false negatives for new attacks while an anomaly-based system might miss detection if the attack is well disguised. False Negatives defeat the purpose of IDS to a certain extent as it fails to prevent the damage caused by an attack.
4. Data Overload: IDS might also suffer from data overload. A key requirement of an IDS is analyzing data in real time. If it does not process data fast enough, there is every chance that it will allow some malicious traffic to pass undetected.

1.4 False Alarm Rate

False Alarm Rate forms a part of false positives for an Intrusion Detection System. It is inherent in every IDS. This is the case when an IDS triggers an alert when there is no malicious activity. In environments where system security is a key issue, network administrators monitor the systems round the clock. They have to investigate each alert and make sure there is no breach of perimeters. Presence of large amount of false alarms makes administration difficult and boring. The credibility of an IDS is thus compromised [6]. The system administrator may develop a tendency to overlook certain alerts thinking they are false. Although the false alarm rate can be reduced to an extent by tuning the system, there is still a chance that such alarms will be generated. Developing new techniques for detection can be one step towards improving the false alarm rate. Implementing Intrusion Detection Systems can pose many problems, but False Alarms are considered the greatest threat [7].

1.5 Solution approach

In this thesis an attempt is made to explore the possibility of an additional parameter to help detect Intrusions. This research will determine if there exists enough information in this parameter so that it can be incorporated in current IDS solutions. Certain assumptions are made before discussing the solution. The solution discussed here is devised based on Anomaly Intrusion Detection as implemented in Snort [8]. For detecting anomalous packets, snort uses a plugin called SPADE (Statistical Packet Anomaly Detection Engine). This plugin determines if a packet is anomalous based on joint probabilities of various header fields in a packet. Based on these probabilities, it calculates an anomaly score, which triggers an alert if it exceeds a certain threshold. Clearly, the number of parameters analyzed affects the decision making in an IDS. As a solution, network packet data content can be classified, which could then be incorporated while calculating the anomaly score. The payload content of a network packet varies by a large amount. There are cases when packets have no payload (TCP syn packets) and cases when the payload has malicious data (in an attack).

This thesis will try to classify the data within an E-mail attachment and try to relate it to a specific type of file and identify patterns generated by different files. These patterns can then hopefully be incorporated as one of the parameters used in Intrusion Detection Systems.

1.6 Design

The scope of this thesis will be limited to E-mail. Specific types of files having extensions .exe, .dll, .doc, .jpg, .mp3, .pdf, .txt., will be classified based on the frequency pattern they generate. Since E-mail is considered as the source of malicious activity, this project will capture traffic on port 110 (POP). E-mail with known payloads will be sent to a mail account. A standard POP mail account (Yahoo) will be used to receive the mail. This mail will be downloaded to a mail client, Outlook Express. A packet sniffer (ethereal) will be employed to capture data on the network as it is loaded to port 110.

The first step will be to isolate traffic on port 110. The response from a POP server can be one of the following

1. TCP [SYN, ACK]
2. TCP [ACK]
3. Response: +OK hello
4. Response: +OK password required
5. Response: +OK maildrop ready
6. Response: +OK (Number) messages
7. New mail
8. Response: +OK server signing off
9. TCP [FIN, ACK]

Packets captured on port 110 will contain traffic corresponding all of the above responses. Packets corresponding to the transfer of new mail to the client are isolated. In this process, packets corresponding to SYN, ACK, authentication, message list are ignored.

Once packets corresponding to new mail which contain the attachment are isolated, headers are stripped from the packet to obtain the payload. A frequency pattern is then generated for this payload. Once patterns from different types of payload are obtained, they are classified into different file types.

1.7 Conducting Experiments

The setup for the experiment is shown in the Figure 2. An e-mail client will be used to send E-mail with an attachment to another client through a server. The receiving client will download the E-mail using Outlook Express. This requires the use of POP protocol. A packet sniffer will be deployed on the network in which the receiving client is operating. The actual data payload of the attached file is then isolated and a pattern is extracted from it.

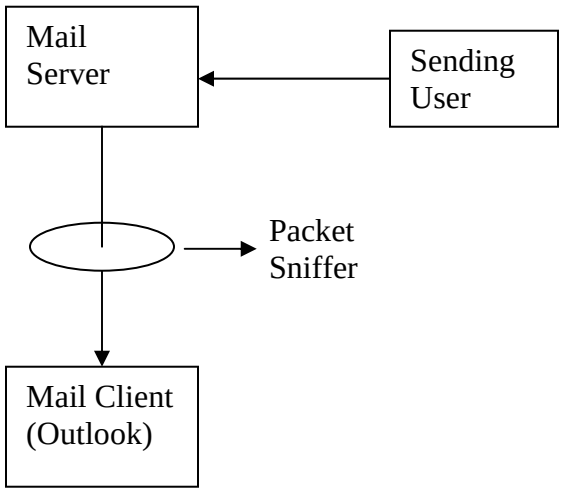


Figure 2. Experiment Setup

1.8 Analyzing Results

Once the packets corresponding to port 110 are captured, the payload of desired packets (packets corresponding to the file attached to the mail) is isolated. An algorithm will be employed to extract a pattern from the captured data. The analysis of these patterns will determine whether these patterns can be used to differentiate traffic. If the patterns do differ, then these patterns can be added as one more of the parameters that can be used for Anomaly Intrusion Detection.

1.9 Reporting the Results

The analysis of patterns obtained can be used to recommend the use of payload of a network packet in distinguishing good and bad network traffic. This will assist in improving the overall performance of IDS.

1.10 Research objective

The objective of this research is to identify one such key parameter for use in Intrusion Detection System. This then may generate a greater chance of detecting network abuses.. This research is to:

1. Determine whether the payload of a network E-mail attachment packet can be represented in a manner such that Intrusion Detection Systems can effectively use it.
2. Obtain data corresponding to specific attached file types as it is transferred over a computer network.
3. Identify patterns using this data.

4. Find a method to relate these patterns to the original data.

Chapter 2

Previous Work

Existing Intrusion Detection Systems perform data collection and analysis using monolithic architectures. A single module collects and analyzes data. This can be located at a prime location just after the firewall [9] [10]. Several other networks which enable distributed computing implement a different approach to network data collection and analysis. These networks collect data from several modules distributed throughout the network. Data collected from these modules is then transmitted to a central location where it is processed [11] [12].

Every Intrusion Detection System must expect unknown attacks at some point of time. In addition, they must also deal with finite failover resources. These resources which include processors, memory can withstand an attack for a small time until the attack can be detected. An IDS cannot survive any meaningful time unless the system can handle an unknown attack in the initial stages [13].

Detecting system vulnerabilities requires a lot of computation and manpower. This costs time and money. Extensive testing must be employed to find these vulnerabilities. But once these vulnerabilities are found, they are exploitable. Successive attacks can be launched in a small amount of time. Unfortunately, stealthy attacks do not execute

unauthorized processes, write unauthorized files or consume a lot of resources including CPU or memory [14]. Detecting such new or stealthy attacks is a very difficult task. Hopefully, E-mail attachment monitoring as proposed here can have a beneficial effect.

Considerable effort has been expended to determine network traffic based on the data content. Emphasis to date has been on built-in inter-sample behavior, where an attempt is made to relate different packets in order to determine their intent. This is done using frequency domain analysis of sampled data [15].

Another problem faced by an Intrusion Detection System is the amount of data which is analyzed. Most Intrusion Detection Systems detect intrusions based on headers of network packets. Many times this information is insufficient to reconstruct network activity which in turn affects the accuracy if the IDS [16].

Data mining techniques have also been used to gather audit data. Data mining models the behavior of intrusions and of normal activities [17]. The data thus obtained can also be combined to construct a picture of the total impact on a system.

Typically, an IDS uses packet headers from the network and transport layers to detect intrusions. This technique might improve IDS performance by filtering out malformed packets at the router level [18].

Systems that analyze application payloads are still evolving. Variations in payload content of a network packet are very high. Hence, processing of the packet payload

requires some information about the application [19]. In such a process, network traffic is partitioned depending on the application and then analyzed. Thus, setting a baseline for normal traffic specific to a certain service.

Efforts have also been made to create a graph based Intrusion Detection System [20]. In such a system, the network activity is depicted graphically to help recognize large-scale attacks more efficiently. A large scale attack may not otherwise provide sufficient information to trigger an alert. Collecting and analyzing data from several thousand hosts is very difficult. Graphs which represent network traffic activity can ordinarily be useful in fast detection of large scale attacks.

Gathering information to supply an Intrusion Detection System has always been a challenge. Information is generally gathered from the data stream by sampling, filtering and approximation. A novel approach uses a summary of data obtained as a “sketch” [21]. These “sketches”, which are obtained at distributed locations, are used to find features in network data. The advantages described include, small size, distributed collection and quick analysis.

Chapter 3

Anatomy of E-mail

Simple Mail Transfer Protocol (SMTP) is used to communicate between two users using plain text ASCII characters. It is independent of any transmission subsystem and requires a reliable data channel [22]. The SMTP communication model is shown in Figure 3. Communication in SMTP takes place in the following manner. Sender-SMTP and receiver-SMTP establish a communication channel. The sender then sends a MAIL command for which the receiver replies with an OK if it accepts mail. The sender then sends a RCPT command identifying the recipient of the mail. If the receiver can accept mail for this recipient, it replies with an OK. Once the recipients are negotiated, the sender sends the mail data. On receiving the data successfully, the receiver will send an OK reply.

Depending on systems, the format in which these messages are stored might differ. However process of communication of messages between two hosts follows a fixed format [23]. The message format is made up of an envelope and contents. The envelope is responsible for transmission and delivery while the contents consist of the object to be delivered. The message consists of lines of ASCII text with no provisions for drawings, facsimile, speech or structured text.

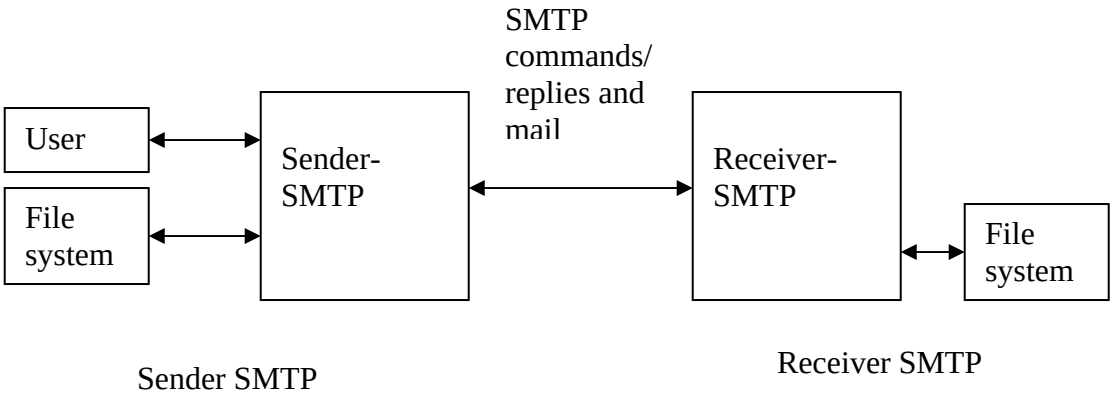


Figure 3. SMTP Communication Model

3.1 Sending Attachments

This technique for sending E-mail caters to simple ASCII text communication only [23] and leaves no provision for multi-part textual and non-textual message bodies. This information must still be communicated within the message body. Limitations of RFC 822 mail are also evident while communicating between RFC 822 hosts and X.400 hosts [24], as X.400 includes non-textual body parts within mail. Non-textual body parts within RFC 822 mail must be discarded which results in loss of information. Additional specifications for representing and exchanging non-ASCII data without loss of information are described in RFC 1521 [24]. RFC 1521 suggests methods by which messages in non-ASCII character sets, formatted multi-font text and non textual material like images, audio fragments, can be represented as textual messages which can be sent via E-mail. The process does not extend E-mail headers to be non-ASCII text data. The mechanisms described in [24] for sending non-textual messages include –

1. MIME-version header: declares conformance with specifications.
2. Content-type header field: specifies type and sub-type of data in message body.
3. Content-transfer-encoding field: specifies encoding that was applied to the data.

4. Two additional header fields to describe data. Content-ID and Content-Description header field.

3.2 MIME

MIME stands for Multipurpose Internet Mail Extensions. MIME provides extensible solution to send various types of data as 7-bit ASCII [25]. MIME creates a header using low-order ASCII characters. It then wraps this header around the attachment and encodes the attachment. The format of a MIME encoded attachment is shown in Figure 4. The message is delimited by the boundary ID at the beginning and end. The contents of the attachment are defined by a series of headers. The headers identify attachment type, original filename, encoding mechanism.

Figure 4. MIME format

```
--Boundary_(ID_g1Gepa0UrzY13XI9WaOapg)
Content-type: application/pdf; name="UG-Research Comp-WV-EPSCoR.pdf"
Content-transfer-encoding: base64
Content-disposition: attachment; filename="UG-Research Comp-WV-EPSCoR.pdf"

JVBERi0xLjIKJSDi48/TCiAKMTggMCBvYmogPDwg……

--Boundary_(ID_g1Gepa0UrzY13XI9WaOapg)--
```

3.2.1 Content-Transfer-Encoding header field

RFC 821 [22] restricts mail messages to 7 bit US ACII data. Content types like 8 bit character or binary data cannot be transmitted over SMTP. To send such files as part of E-mail, they must be encoded as plain text ASCII. Content-Transfer-Encoding describes invertible mapping of data. This mapping could then be exchanged with mail transfer protocol. The mechanisms described in [24] are

1. 7bit
2. quoted-printable
3. base-64
4. 8bit
5. binary
6. x-token

Content-Transfer-Encoding appearing in the message header applies to the entire message body, while Content-Transfer-Encoding applies to a part of the body if it appears in the body of the message. Since E-mails are character oriented, the above mechanisms will convert arbitrary octet streams and not bit streams. To represent bit streams, they must first be converted to 8-bit byte streams. RFC 1521 [24] also describes two standard Content-Transfer-Encoding types as

1. quoted-printable: represents data corresponding to printable characters in ASCII code. The encoding is such that mail transport systems are unlikely to modify the data and hence integrity is maintained.
2. base 64 encoding: represents arbitrary sequence of octets in human readable form. (see section 3.2.2)

Converting data to ASCII so that it can be sent via mail can also be done using UUencode. UUencode uses low-order ASCII characters to convert complex data to be sent over the communication channel. The receiver then decodes this message to obtain the original file.

3.2.2 Base64 Encoding

BASE 64 encoding is used to convert non-ASCII files into 7-bit ASCII. It uses a 65-character subset of US-ASCII in which 6 bits represent a printable character. The algorithm for converting any arbitrary sequence to base64 involves grouping 24 bits from the input to represent four encoded characters. The group of 24 bits is obtained by concatenating three 8-bit groups. This group in turn is assumed as a concatenation of four 6-bit groups. Each 6-bit group is then translated to a character as identified in Table 1. The decoder ignores characters, which do not belong to the table. The encoded output stream is represented in a line of 76 characters or less. Special characters (like ".", CR, LF), which are specified for SMTP, are omitted from this table.

In case, the input data stream is not a multiple of 24 bits, zero bits are added on the right of the data stream to form an integral 6-bit group. A 65^{th} character given as '=' is used for the purpose of padding. The input for base64 encoder is always an integral multiple of an octet. Hence, three cases are possible:

1. Final quantum is integral multiple of 24 bits – no padding necessary.
2. Final quantum is 8-bits long – two padding characters '=' are added.
3. Final quantum is 16-bits long – one padding character '=' is added.

Padding characters also indicate the end of data. More than two padding characters at the end of message are considered illegal.

Table 1: The Base64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	I	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Chapter 4

Configuration of the Experiment

For the purpose of determining a pattern from network data, a small subset of traffic present on a computer network is considered (E-mail attachments). Two known clients are used to generate required traffic. Both the clients communicate with a mail server. The principle function of client one is to send mails to client two with a known payload. Client two will receive these E-mails which provide necessary data for analysis. Client two belongs to a network where a packet sniffer is placed in promiscuous node to capture the network traffic. For the purpose of capturing packets, an open source Network Protocol Analyzer, Ethereal was used. Filters are set on the sniffer such that only packets corresponding to port 110 are captured. The captured packets are saved as raw data (libpcap format) for further analysis.

The format of these captured packets is given in Figure 5. Since filters are used on the packet sniffer, only those packets, which have a source port of 110 (POP) are captured.

4.1 Algorithm

The following steps were used to obtain a pattern from the captured packets.

1. Capture packets using a network sniffer (ethereal).
1. Set filters on the sniffer so that it will capture traffic with source port 110 (POP).
2. Isolate traffic corresponding to e-mail attachment from captured packets. (Captured packets will also contain other POP communication like SYN, SYN-ACK, authentication, message list, FIN, etc)
3. For every packet that corresponds to the attachment, determine the frequency of the Base64 alphabets.

4.2 Test Data

Seven types of files will be sent as attachments with E-mail. These types are binary executable (.exe), dynamically linked library (.dll), audio (.mp3), MS word document (.doc), image file (.jpg), Adobe Acrobat PDF file (.pdf) and plaintext file (.txt). These files are selected from random locations and vary in size.

A single captured packet contains the following fields.

1. Frame header – has a length of 12 bytes. This field indicates frame number, arrival time, packet length and captured length.
2. Ethernet header – has a length of 14 bytes. Contains source and destination Ethernet address as well as a type field, which indicated underlying network layer protocol.
3. IP header – as we are capturing packets on port 110, all the packets correspond to IP. Hence the Ethernet header is followed by the IP header, which indicates source, destination IP address, various flags, underlying protocol, packet length, etc.
4. TCP header – this field indicates source and destination port, sequence number, acknowledgement number.
5. Payload – this field contains the payload of the network packet. There is no payload in case the packet is a SYN or ACK. The payload also varies depending upon data exchange. Thus, it may contain authentication information, list of messages, new messages, etc.
6. CRC checksum – this field is part of the Ethernet header and indicates whether data is corrupted during transport.

Figure 5. Captured Packet Data Structure

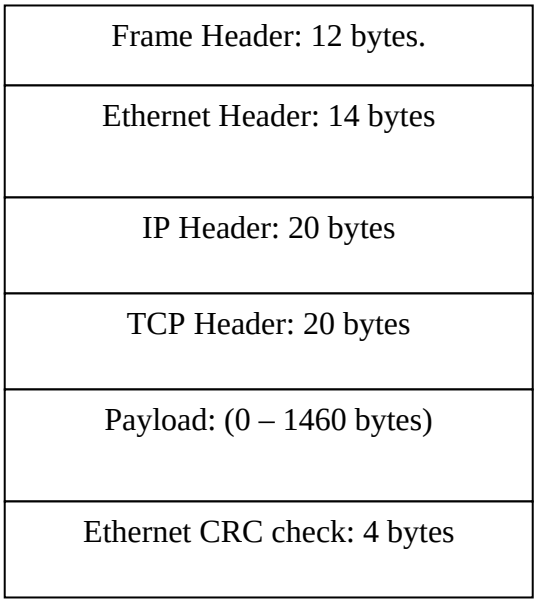


Table 2 shows the details of the data which is used for the experiment. The seven types of files used for the experiment varied in size as shown in table 2. Ten files of each type were used. Since the Maximum Transfer Unit is 1500 bytes, the number of packets generated by each file attachment varied from 2 packets for the smallest text file (2KB) to 3339 packets for the largest audio file (4874KB). Table 3 shows this number of packets which were generated by the files selected for the experiment. I believe the number of packets analyzed to obtain a frequency pattern is statistically sufficient.

Table 2: Range of File Size

File Type	File size (KB)	
	Minimum	**Maximum**
.exe	17	533
.dll	15	569
.pdf	56	571
.doc	23	315
.txt	2	552
.mp3	16	4874
.jpg	38	438

Table 3: Number of Packets Generated

File Type	Total File Size	Number of Packets
exe	1502	1029
dll	1003	687
pdf	2230	1528
doc	987	677
txt	1091	748
mp3	21126	14470
jpg	1502	1029

4.3 Isolating the E-mail attachment

The next step is to isolate the e-mail attachment. For this purpose, we determine all types of packets that were captured. The following types of packets were identified.

1. TCP [SYN, ACK]
2. TCP [ACK]
3. Response: +OK hello
4. Response: +OK password required
5. Response: +OK maildrop ready
6. Response: +OK (Number) messages
7. New mail
8. Response: +OK server signing off
9. TCP [FIN, ACK]

The packets corresponding to New Mail are the ones of interest. These packets can be identified from the response in the payload.

Once these packets are isolated, the frequency of the data will be found. For this purpose, samples of 8 bits are considered. These samples are 7 bit ASCII characters with the MSB set to 0. The length of a single packet is limited to 1460 bytes due to limitations on Maximum Transfer Unit (MTU). The frequency of a particular character is found as the number of occurrences of the character in the given packet.

Chapter 5

Observations

From the experiments, it can be confirmed that distribution of data is limited to US-ASCII character range. Within this range, there are three distinct groups. These three groups can be identified as

1. Group 1 - Sample value 48 – 57: corresponding to 0 – 9 in ASCII table
2. Group 2 - Sample value 65 – 90: corresponding to characters A – Z in ASCII table and
3. Group 3 - Sample value 97 – 122: corresponding to characters a – z in ASCII table.

Apart from these three groups, two characters CR (ASCII 0x0a, Sample value 10) and LF (ASCII 0x0d, Sample value 13) are found to occur at a regular frequency. This is due to the fact that every mail server has a line length limit of 76 characters. While transmitting an encoded file, a new line character is introduced so that the encoded message conforms to standards.

For a packet of size 1500 bytes, the frequency of CR and LF was observed to be 23 to 24.

Figures 6 through 19 show the frequency distributions for the seven types of files that were considered. A distinct difference in the frequency distributions of all the seven types can be seen.

Compared to the standard occurrence of CR/LF character (23 to 24 times per packet), the variation in frequencies of characters is very high for an application or a dll file (above 200 for Application and above 150 for dll at sample value 65). The peak frequency was highest for a word document at 1070. For other file formats the peak frequency was observed in the range 30 – 80 for the same frequency of CR/LF (23 or 24).

Figure 6 shows the frequency distribution for 20 packets of an E-mail attachment of an executable (.exe). It can be seen that the variations in peaks for the three groups are similar. Sample value 65 has the peak frequency. Sample value 47 has the peak frequency in group 2 while group 3 has no specific sample with the consistent peak frequency. Variation in frequency is more in groups 1 and 2 while it is less in group 3. Table 3 confirms this observation as the standard deviation for the three groups is 25.6, 37 and 8.53 respectively. Figure 7 shows a profile of a single packet.

Figure 6. Frequency Distribution of Attachment - Application

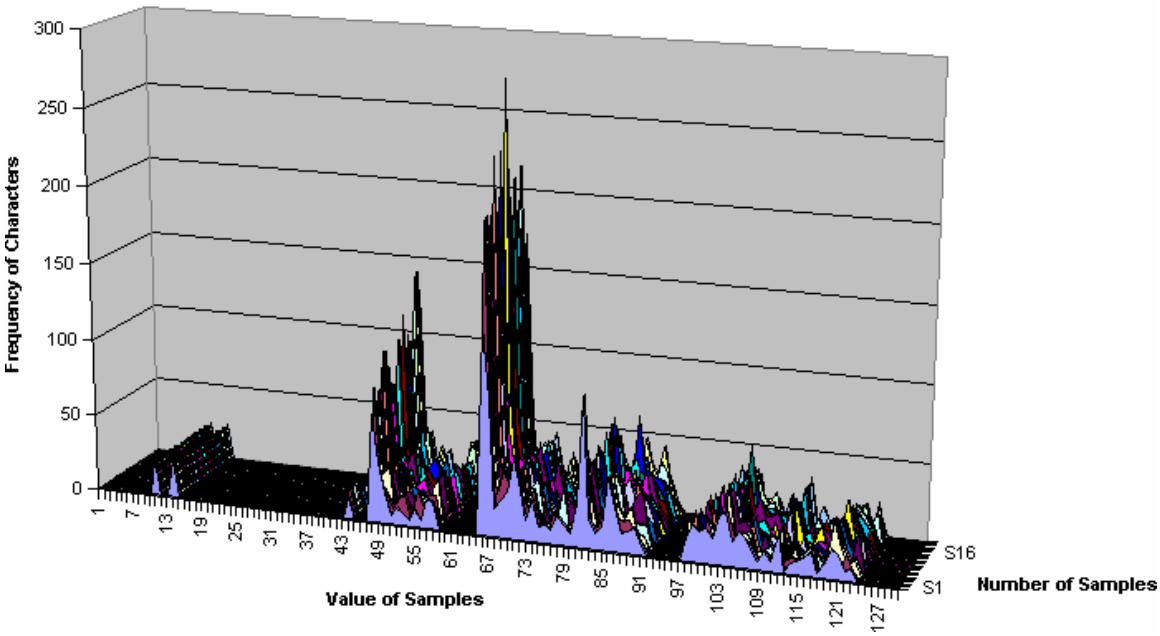


Figure 7. Frequency Distribution of Attachment - Application - Sample Single Packet

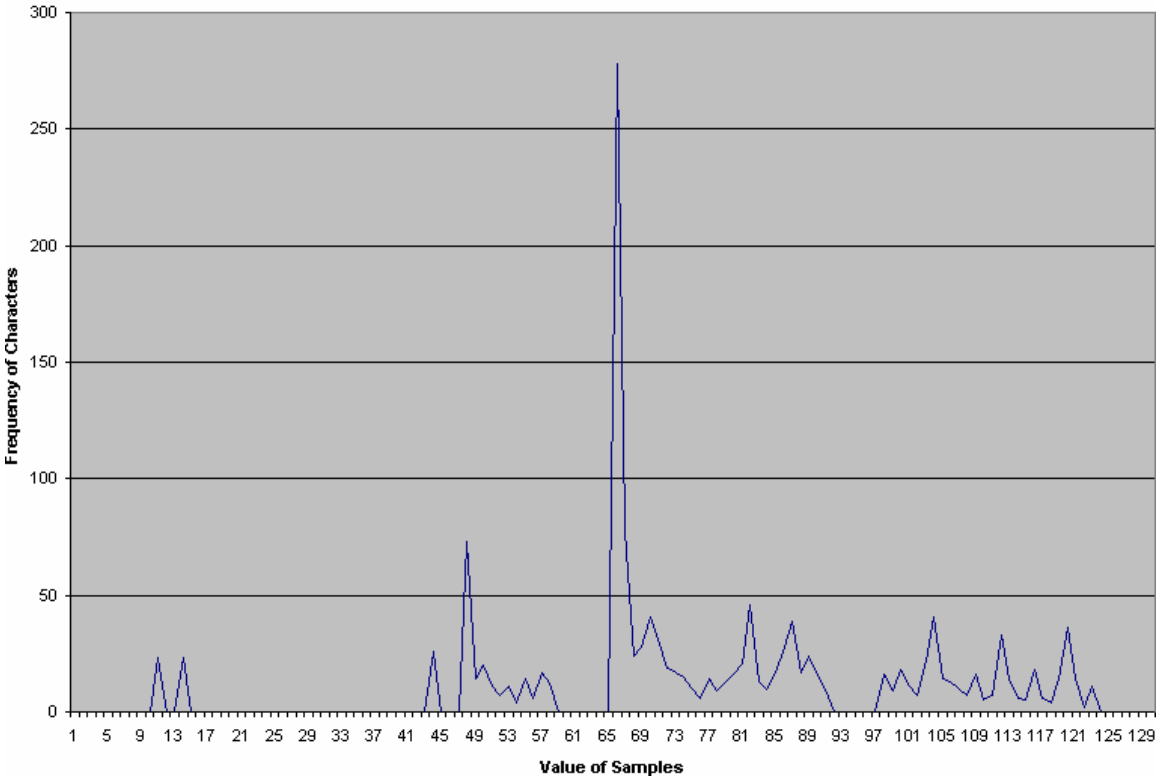


Figure 8 shows a sample frequency distribution for a packet corresponding to a .dll file. Here we observe that the frequency response is similar to that of an application but the peak values of frequencies are lower. Figure 9 shows a single packet.

Figure 8. Frequency Response of Attachment - dll

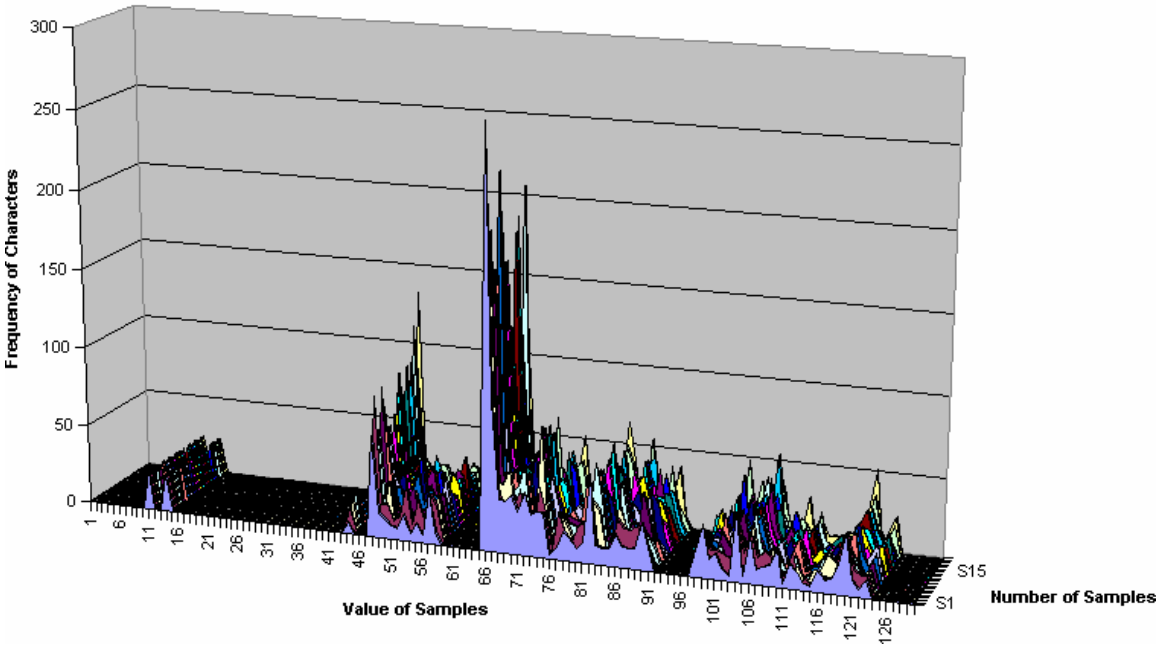


Figure 9. Frequency Response of Attachment - dll - Sample Single Packet

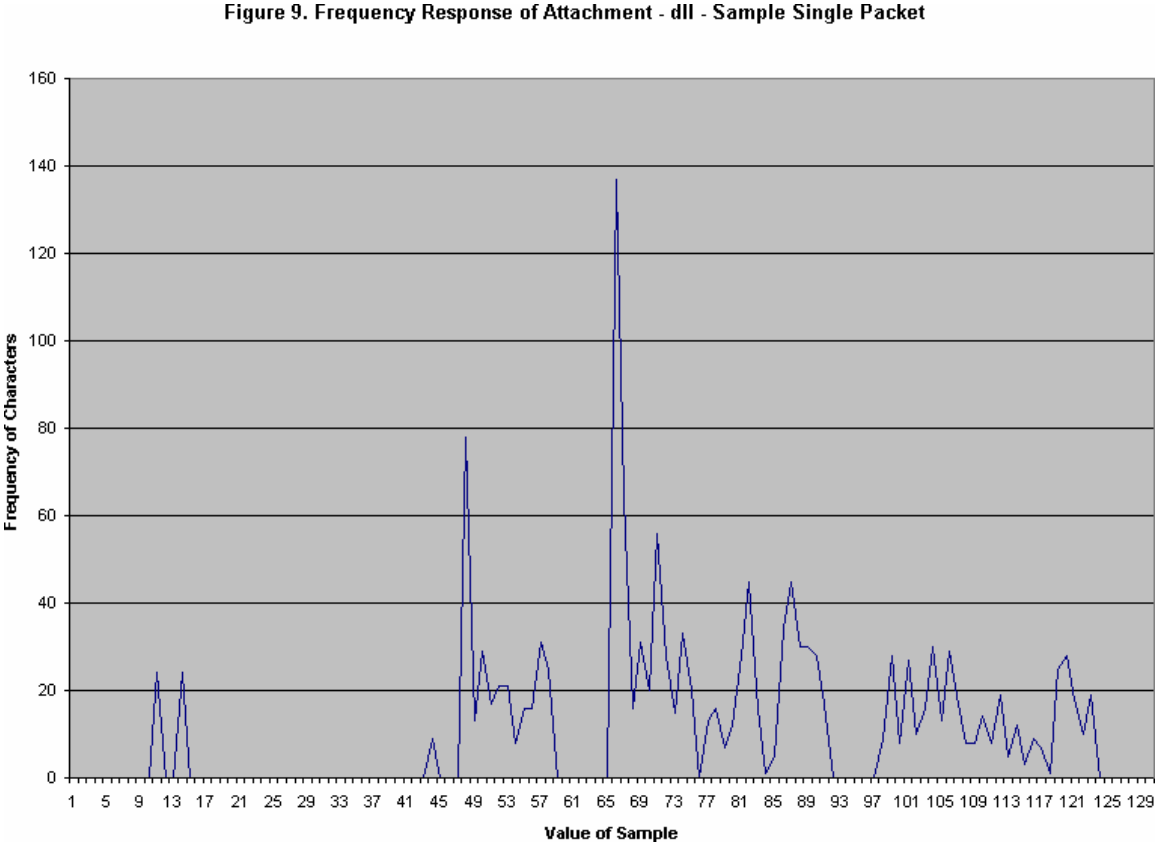
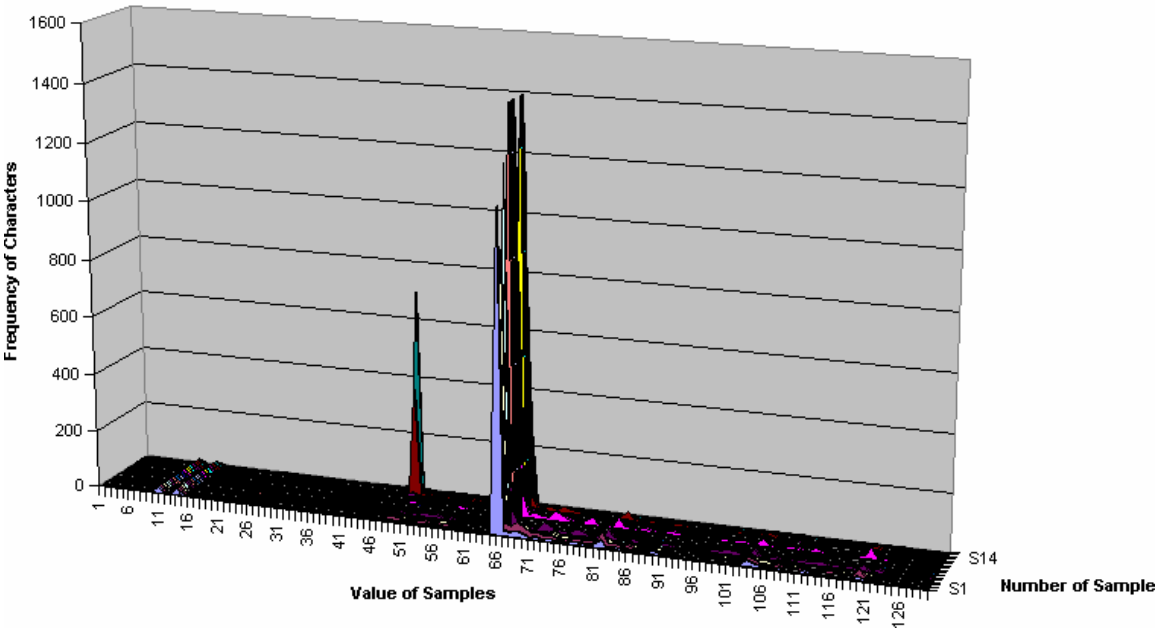


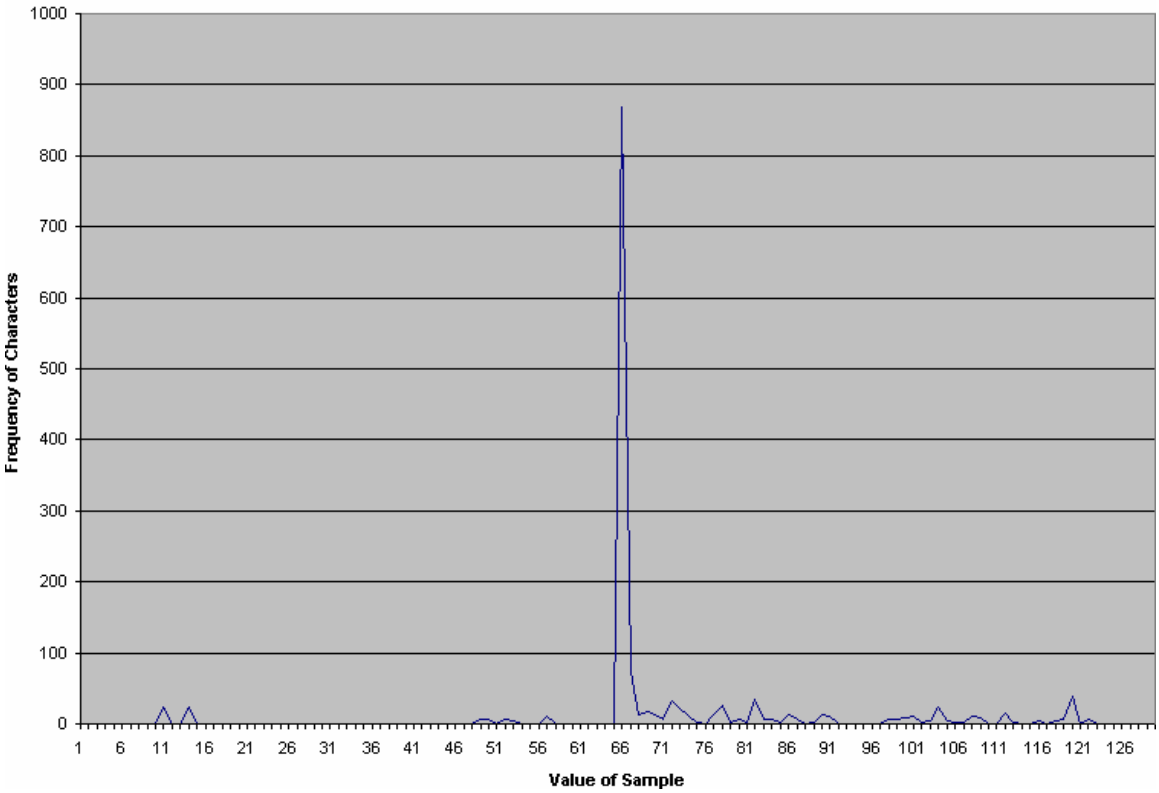
Figure 10 and 11 show the frequency distribution for a word document (.doc). it can be seen that the frequency corresponding to sample value 65 is much higher. Table 3 confirms that the standard deviation for group 3 is 209. There are very few samples in group 3.

Figure 10. Frequency Response of Attachment - doc



.

Figure 11. Frequency Response of Attachment - doc - Sample Single Packet



Frequency distribution for a pdf file is shown in figures 12 and 13. The peak frequency for pdf file is closer to CR/LF frequency. It is also observed that variation in frequencies for the three groups in case of a pdf file is minimum. This can be confirmed from Table 3.

FIgure 12. Frequency Distribution of Attachment - pdf

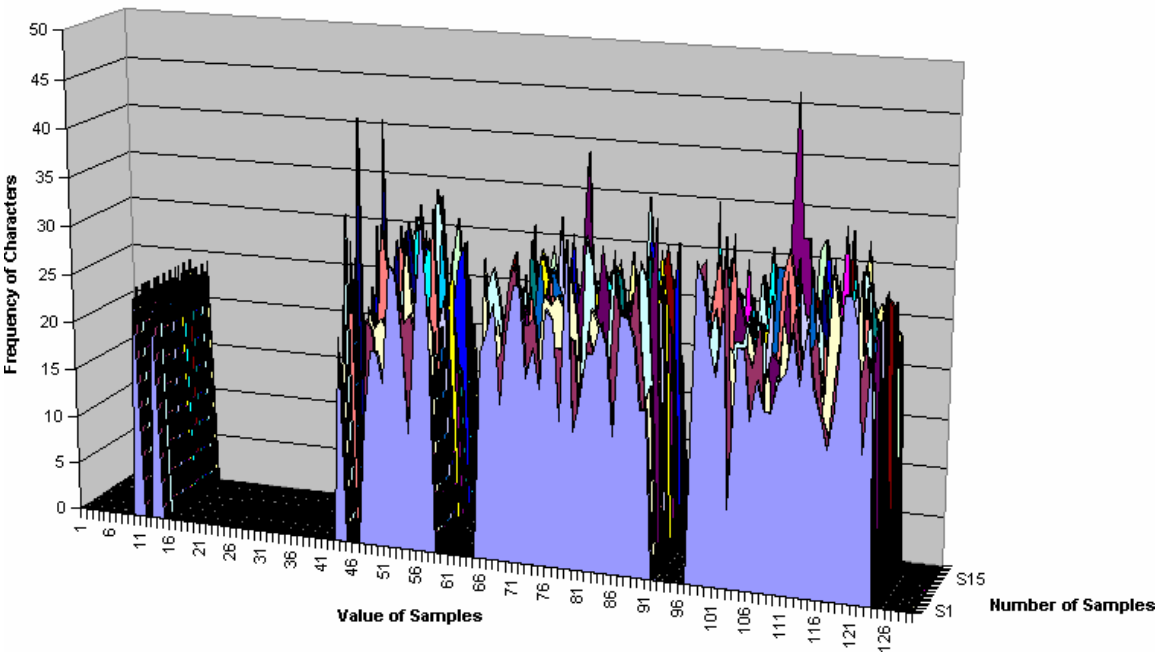
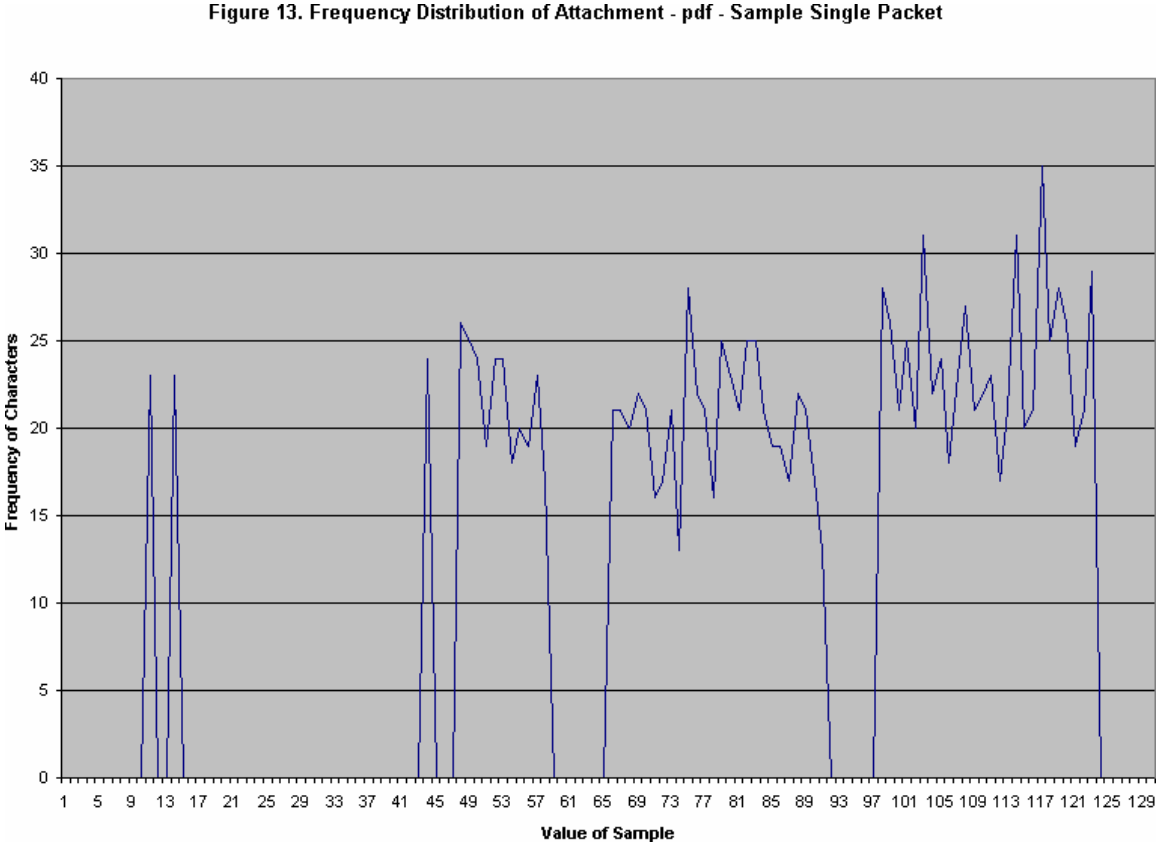


Figure 13. Frequency Distribution of Attachment - pdf - Sample Single Packet



For a text file, the peak value of frequency ranges between 50 and 80. However variation in frequency within a group is more. The values of standard deviation of frequency for a text file for the three groups are 11.3, 10.7 and 12.5.

Figure 14. Frequency Distribution of Attachment - text

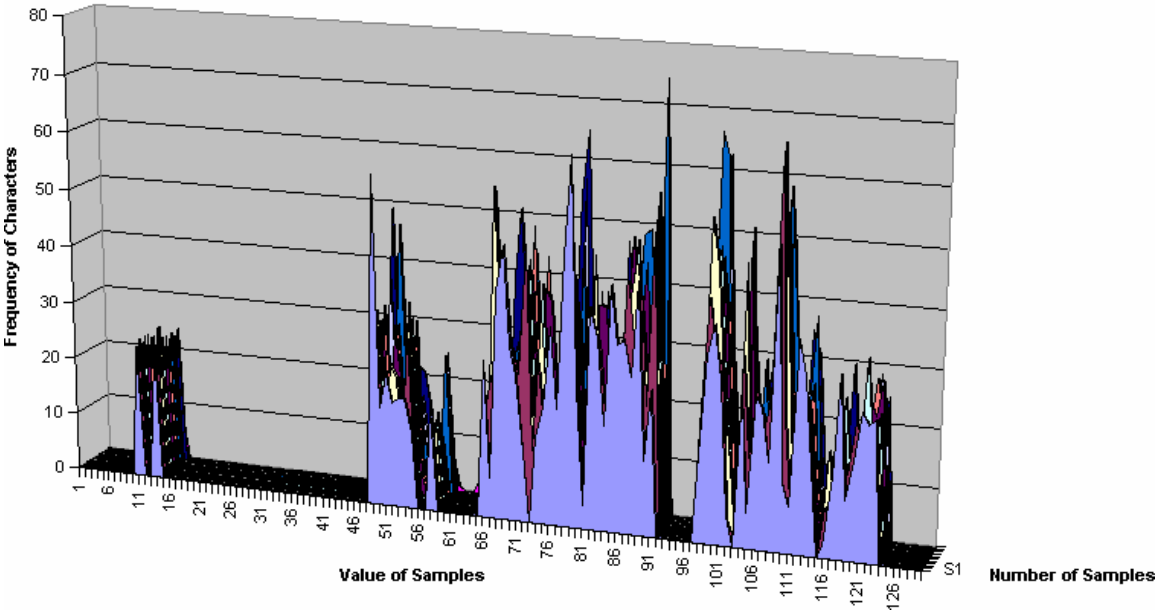
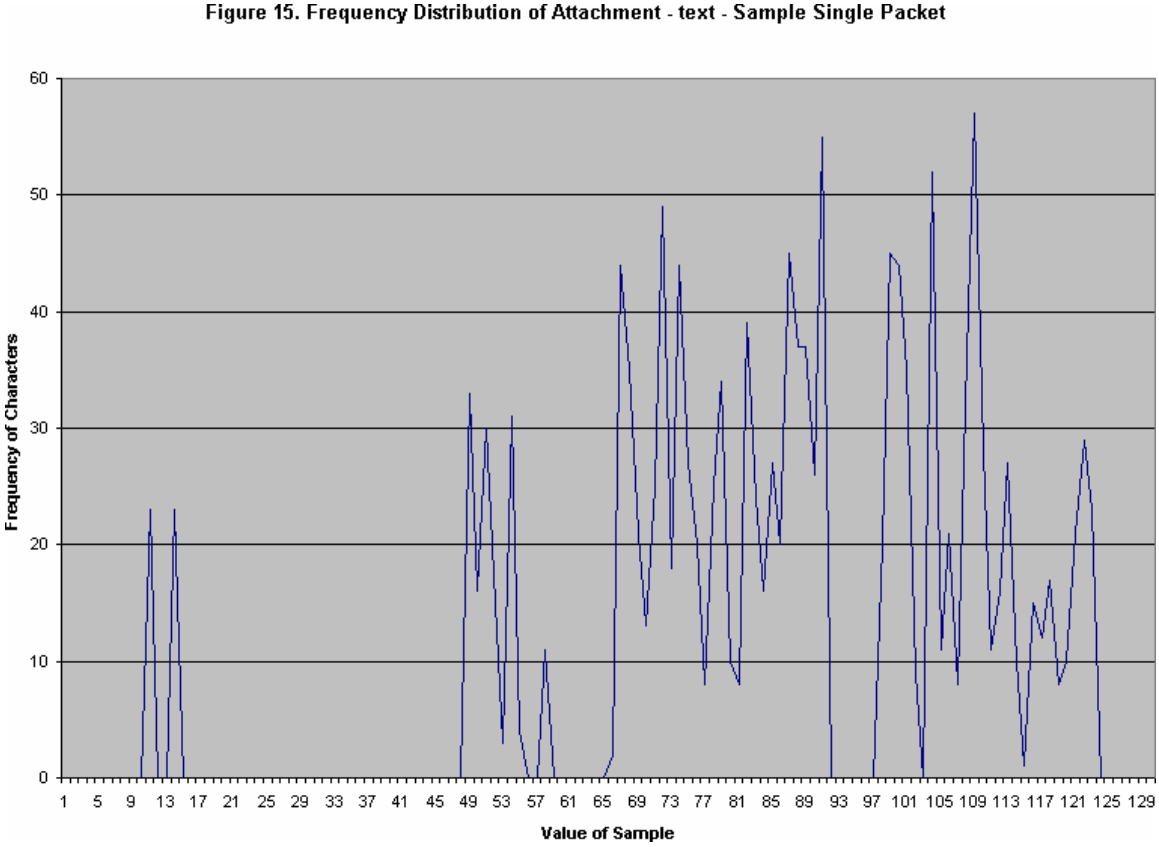
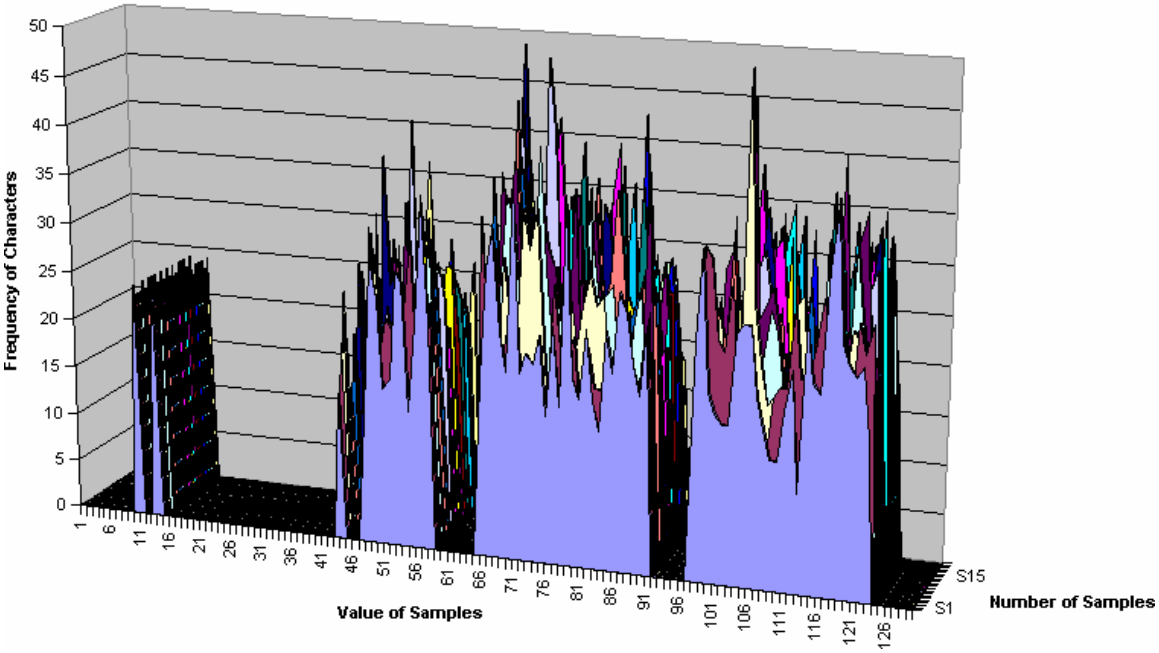


Figure 15. Frequency Distribution of Attachment - text - Sample Single Packet



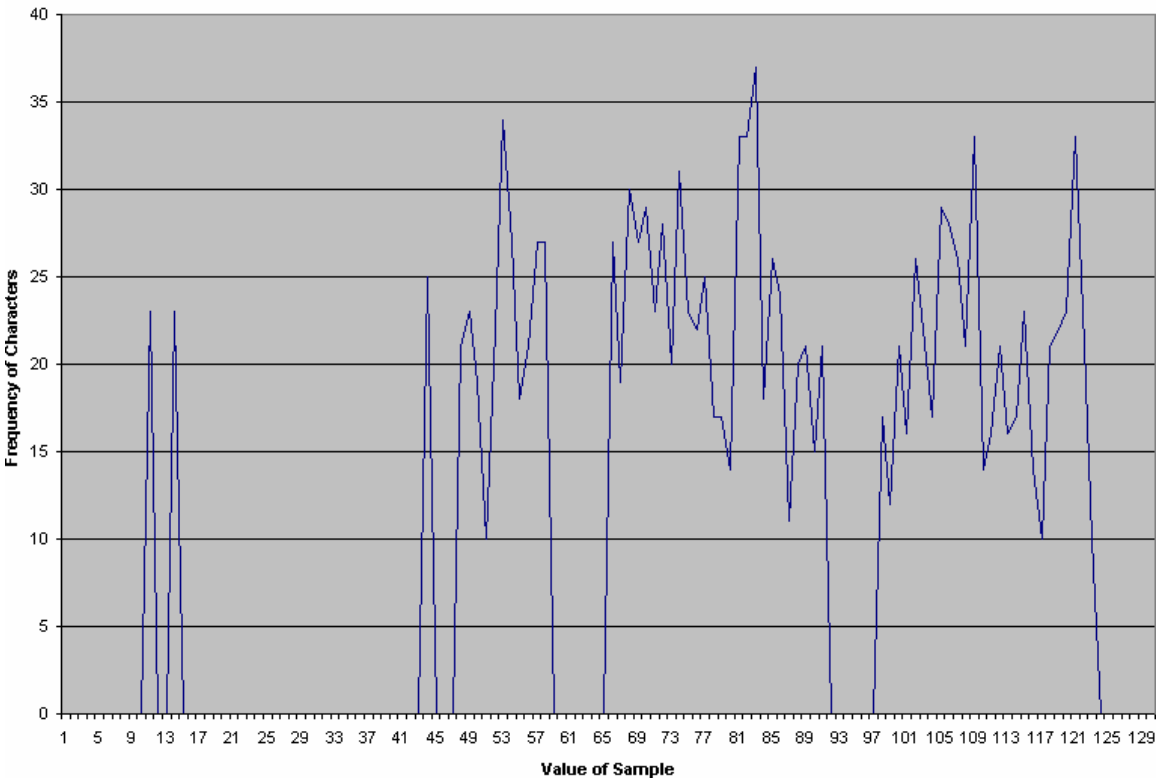
Figures 16 and 17 show frequency distributions for an image (jpg) file. The values of standard deviation are uniform for all the three groups.

Figure 16. Frequency Distribution of Attachment - jpg



s.

Figure 17. Frequency Distribution of Attachment - jpg - Sample Single Packet



Figures 18 and 19 show frequency distributions for audio (mp3) files. In this case, group 2 has slightly higher values for frequency compared to groups 1 and 3. Standard deviation for group 2 is higher than standard deviation for groups 1 and 3.

Figure 18. Frequency Distribution of Attachment - Audio

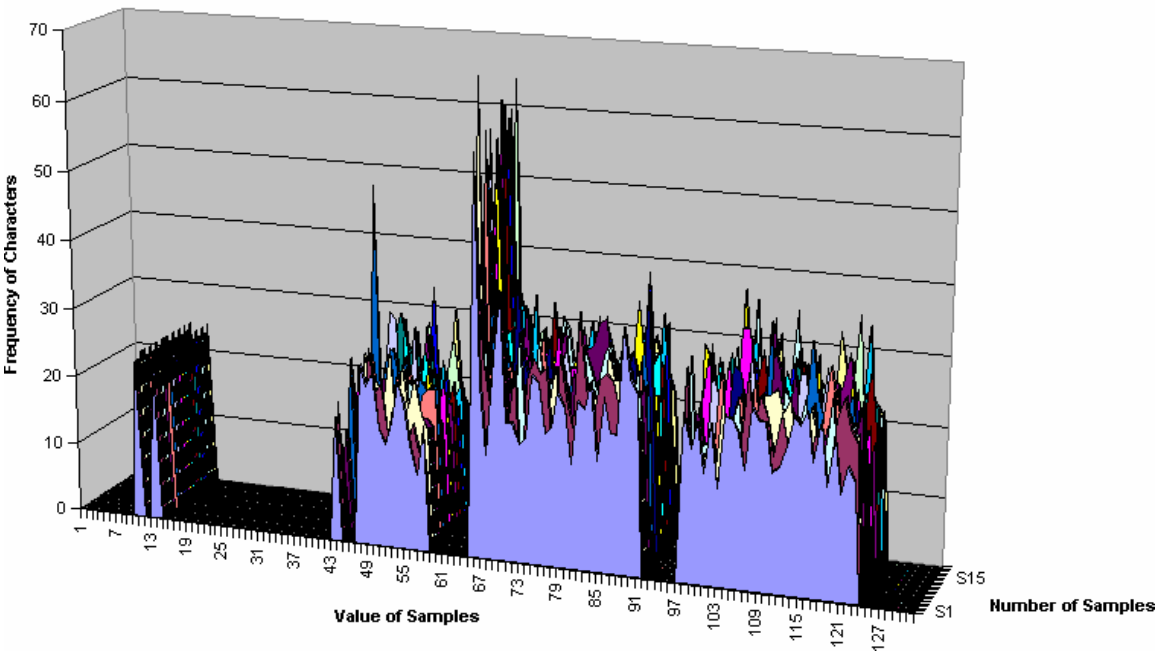
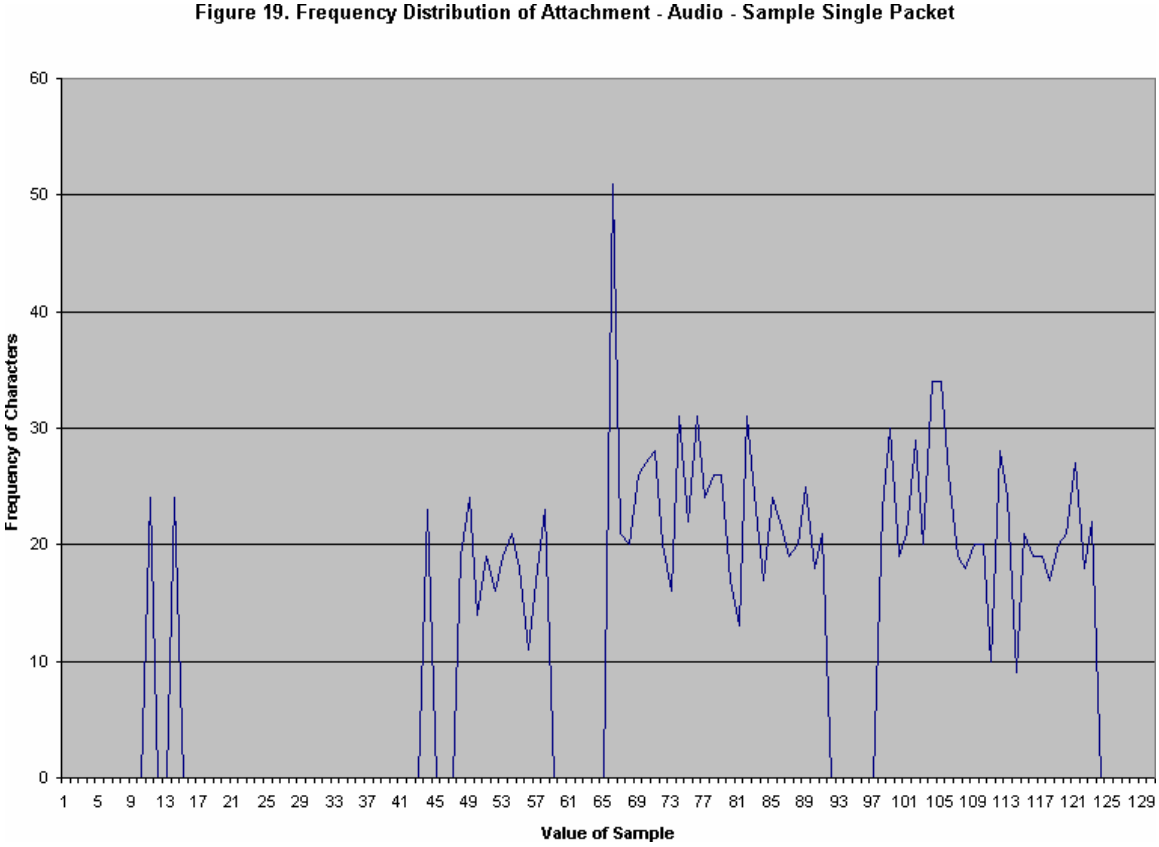


Figure 19. Frequency Distribution of Attachment - Audio - Sample Single Packet



Figures 20 and 21 show frequency distributions for a pdf file in Russian language while it is observed that compared to the frequency distribution of other (exe, dll, mp3, jpg, etc) file types, the frequency distribution of the pdf file in Russian language is much closer to the pdf file in English language (Figure 12). The current format of E-mail communication requires all the attachments to be encoded using base64 or similar algorithm. This means that pure ASCII text files are also encoded before attached to an E-mail. This technique ensures that the data sent through an E-mail depends on file type and not on the language. Hence, it is observed that packets corresponding to English text files do not follow the frequency of the English text.

Figure 20. Frequency Distribution of Attachment - pdf (Russian)

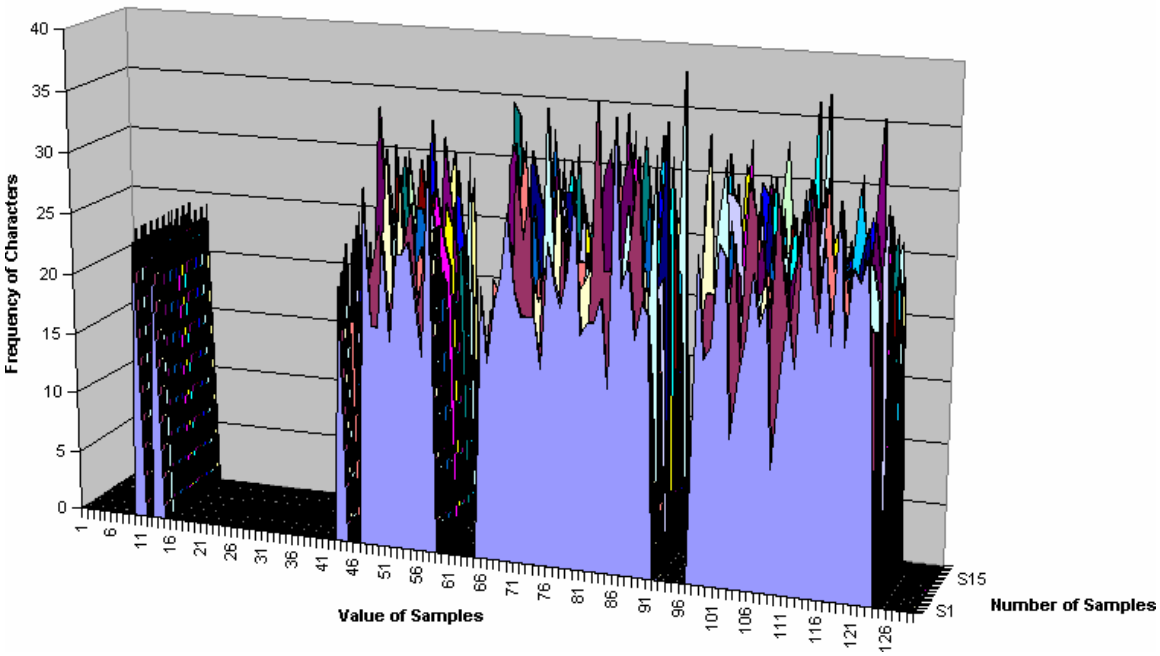


Figure 21. Frequency Distribution of Attachment - pdf (Russian) - Sample Single packet

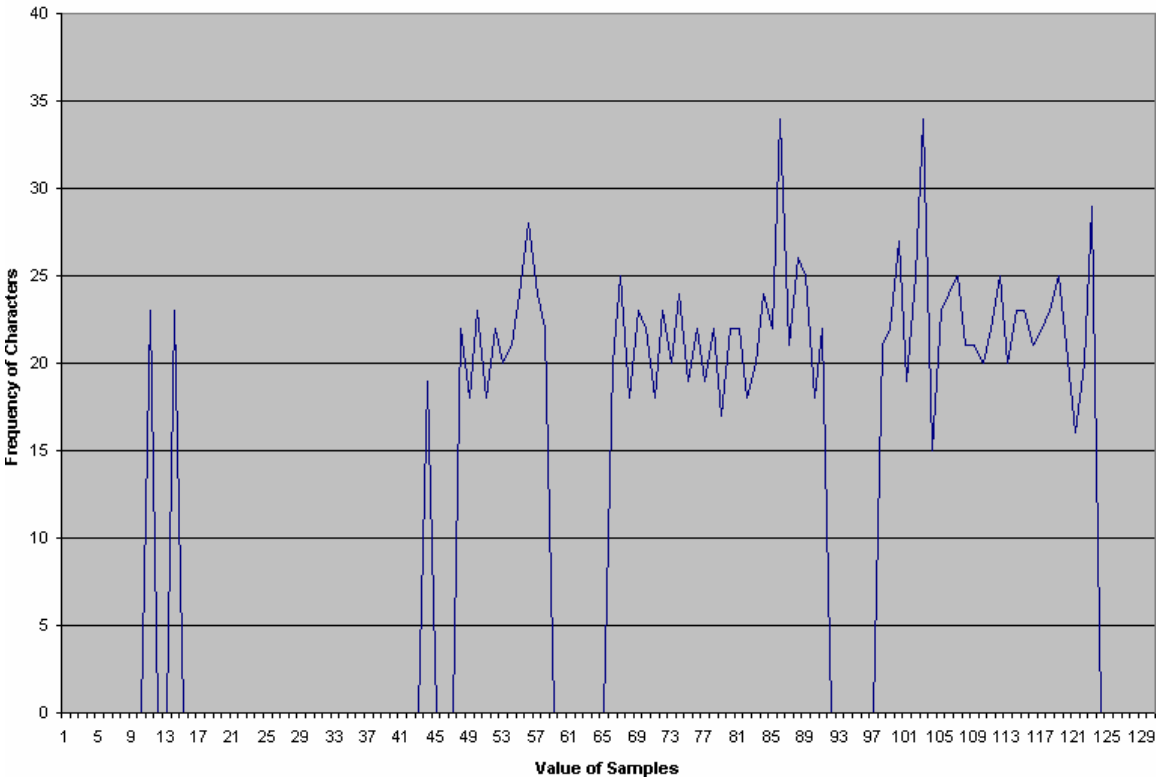


Table 4 shows the comparison of average frequencies of groups 1, 2 and 3 for all seven types of files. Figure 22 represents observations of table 2 graphically. It can be observed that the average frequencies for file types .exe and .dll are similar. Average frequencies for all three groups are almost same for a .pdf file while the frequencies for image and audio file types are similar. Average frequency of group 2 for a .doc file is much higher than frequency of group 1 or 3.

Table 4: Average frequencies of the three groups

File extensions	Average frequency		
	Group 1	Group 2	Group 3
Exe	23.4	30.7	13.2
Dll	23	29.4	14.5
Pdf	21.9	21.6	22.5
Doc	8.04	45.7	2.46
Txt	13.1	24.6	19.3
mp3	19.5	24.1	21.3
Jpg	20.2	23.6	21.4

Figure 22. Average Frequency of Groups

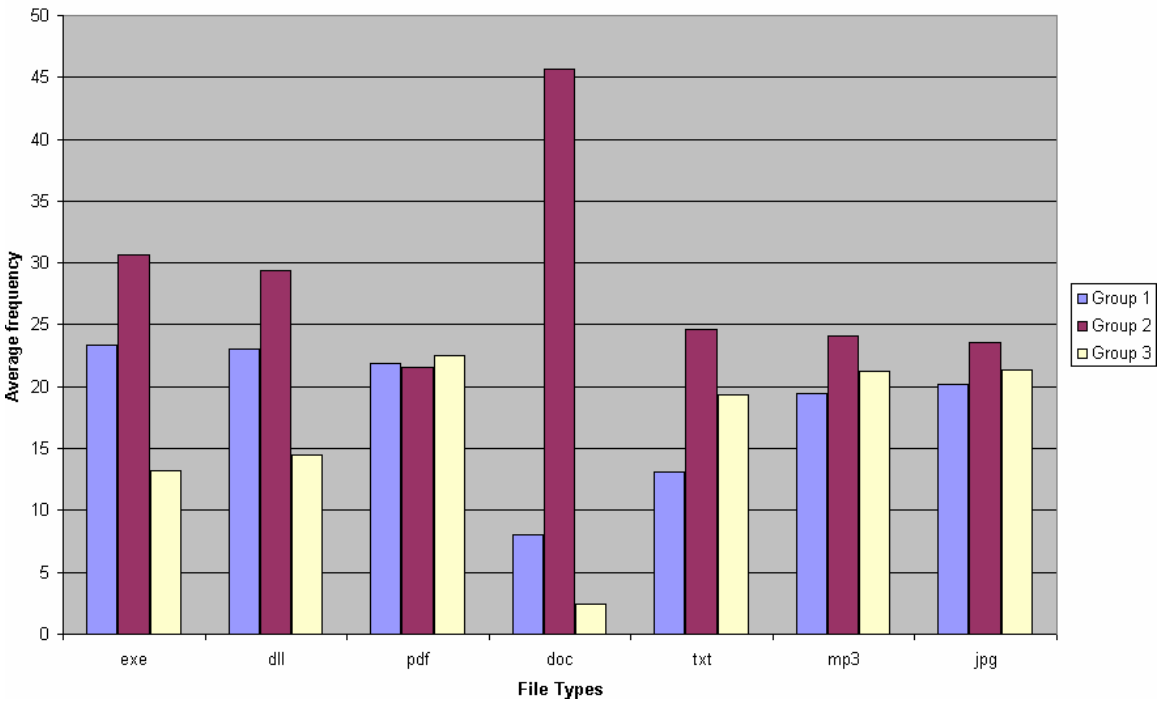


Table 5 shows standard deviation of frequencies of the three groups for all types of files. Figure 23 represents the results graphically. Standard deviation of group 2 frequency of .doc file is the highest. Standard deviation for all three groups of file types .exe and .dll is similar to corresponding group while standard deviation for all three groups is same for .pdf, .txt and .jpg. Standard deviation is the lowest for all three groups of .pdf file type.

Table 5: Standard deviation of frequencies of the three groups

File extensions	Standard deviation		
	Group 1	**Group 2**	**Group 3**
Exe	25.6	37	8.53
Dll	19.3	29.5	8.22
Pdf	1.24	0.93	1.29
Doc	20	209	2.29
Txt	11.3	10.7	12.5
mp3	1.6	6.64	1.82
Jpg	2.68	3.14	3.38

Figure 23. Standard Deviation for Frequencies of the three Groups

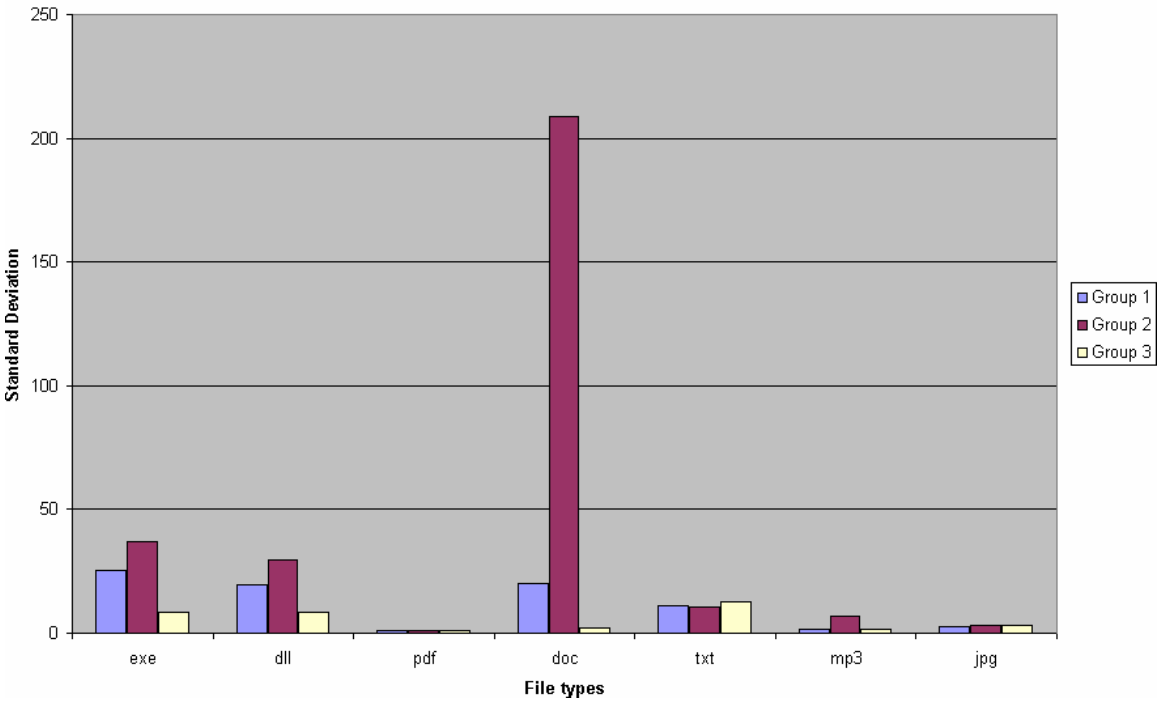
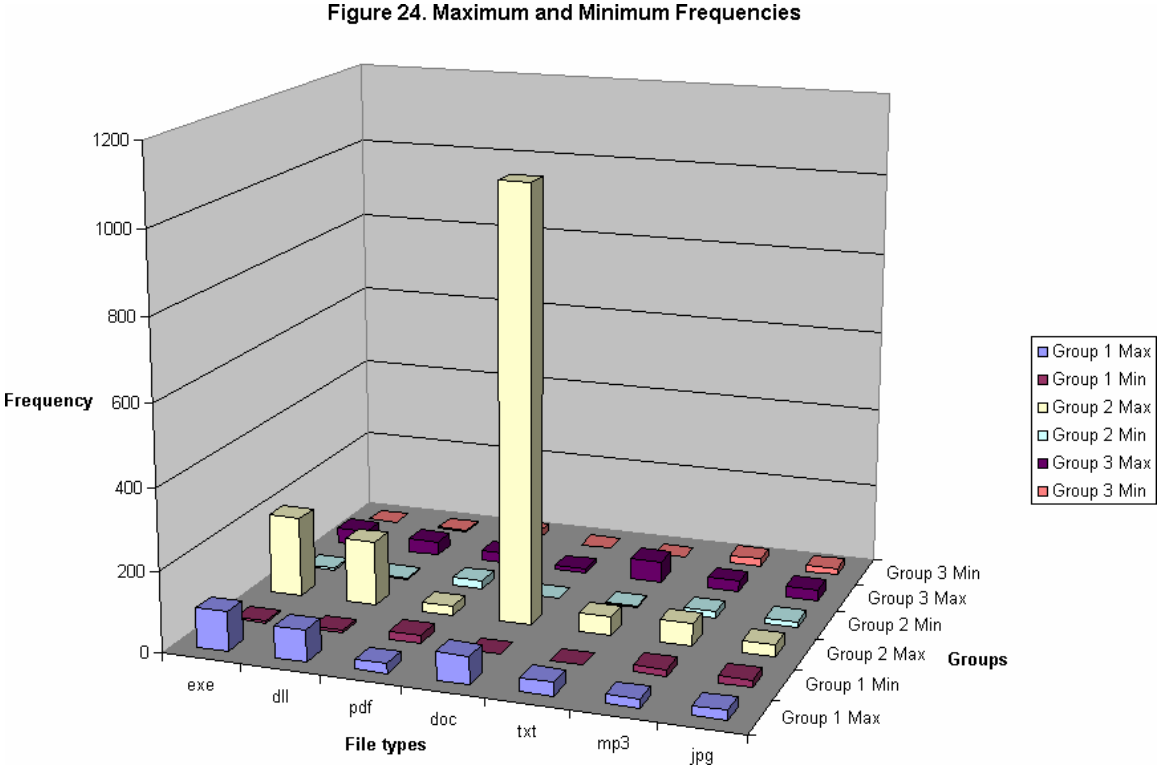


Table 6 shows the maximum and minimum value of frequency for each group of the seven file types. Figure 24 shows the same results graphically.

Table 6: Maximum and minimum frequencies

File extensions	Group 1		Group 2		Group 3	
	Max	**Min**	**Max**	**Min**	**Max**	**Min**
exe	97.3	4.76	199	5.05	39.1	2.48
dll	77.9	6.7	162	4.25	33.3	3
pdf	24.4	20.2	23.7	20	24.4	18.8
doc	68.2	0.81	1070	0.69	9.88	0.69
txt	33.4	0	45.6	3.9	52.1	0.1
mp3	21.8	16.9	54.7	17.1	25.5	18.5
jpg	24.3	17.6	30.3	16.2	27.6	15.6

Figure 24. Maximum and Minimum Frequencies



Chapter 6

Conclusion and Future Work

6.1 Conclusion

There exist sufficient cases to distinguish packet data based on the frequency pattern generated by it. The pattern obtained from the E-mail attachment is restricted to three principle groups since E-mail attachments use Base64 alphabets. The three zones correspond to the ranges 0-9, A-Z, a-z in the ASCII table. The frequency distribution of data for these three zones corresponds to the file type that is transmitted. This frequency can be compared with the occurrence of CR, LF characters.

A file type can be identified by the frequency pattern it generates. Parameters like peak frequencies, relative distribution of frequencies and standard deviations of a network packet can be used to classify file types. Classification of files in this manner requires no information from the packet headers, which may serve as an advantage if a packet is spoofed.

For the seven types of files that were considered, difference in pattern for .pdf, .doc, .txt, .mp3 and .jpg was clearly found. The pattern for .exe and .dll was similar to a large extent. The values of standard deviation and minimum and maximum frequencies were also similar. Based on the parameters considered for finding the difference, these two file types could not be clearly differentiated.

6.2 Future Work

Pattern recognition can be applied to network packet data without having to know the header contents. The work described here identifies files attached to E-mails. The same principle could be extended to traffic on other ports. This could be applied to identify normal and anomalous traffic on such ports.

E-mails restrict network traffic to US-ASCII characters. Communication on other ports could contain more complex data types. It would be interesting to know what kind of pattern will be generated if the data is not limited to ASCII character set.

The current implementation is an attempt to classify traffic based on data content. The next step would be to apply this principle to detect and classify traffic in real time. Because capturing and analyzing network traffic data in real time is a time consuming task, it would be interesting to see if it can applied in real time.

Efforts can be made to explore classification methods other than the one described in this thesis. For this purpose, advanced statistical methods or neural networks can also be employed.

An assumption was made regarding the statistical significance of the number of packets analyzed in making a decision about pattern related to an application. Exploration in this direction needs to be done to establish the minimum number of packets required for such analysis.

Bibliography

[1] Mukherjee, B., Heberlein, T. L., Levitt, K. N., Network Intrusion Detection, IEEE Network, 8(3):26-41, May/June 1994.

[2] Spafford, E., Zamboni, D., Data Collection Mechanisms for Intrusion Detection Systems, CERIAS Technical Report 2000 – 08, June 2000.

[3] Sommer, R., Paxson, V., Enhancing Byte-Level Network Intrusion Detection Signatures with Context,

[4] Kumar, S., Classification and Detection of Computer Intrusions, Dissertation, Purdue University, August 1995.

[5] Mahoney, M. V., Chan, P. K., Detecting Novel Attacks by Identifying Anomalous Network Packet Headers, Florida Institute of Technology.

[6] Axelsson S., The Base-Rate Fallacy and the Difficulty of Intrusion Detection. ACM Transactions on Information and System Security, 3(3), 186-205.

[7] Lundin, E., Jonsson, E., Some Practical and Fundamental Problems with Anomaly Detection, October 1999.

[8] Staniford, S., Hoagland, J. A., McAlerney, J. M., Practical Automated Detection of Stealthy Portscan.

[9] Heady, R., Luger, G., Maccabe, A., Servilla, A., The Architecture of a Network Level Intrusion Detection System, Technical Report, University of New Mexico, Department of Computer Science, August 1990.

[10] Heberlein, L., Dias, G., Levitt, K., Mukherjee, B., Wood, J., Wolber, D., A Network Security Monitor. Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1990.

[11] Hockberg, J., Jackson, K., Stallings, C., McClary, J. F., DuBois, D., Ford, J., NADIR, An Automated System for Detecting Network Intrusion and Misuse. Computers and Security, 12(3): 235-248, May 1993.

[12] Snapp, S. R., Brentano, J., Dias, G. V., Goan, T. L., Heberlein, L. T., Ho, C., Levitt, K. N., Mukherjee, B., Smaha, S. E., Grance, T., Teal, D. M., Mansur, D., DIDS (Distributed Intrusion Detection System) – Motivation, Architecture and An Early Prototype. Proceedings of the 14th National Computer Security Conference, Pages 167-176, October 1991.

[13] Giorgio Giacinto, Fabio Roli, and Luca Didaci, A Modular Multiple Classifier System for the Detection of Intrusions in Computer Networks.

[14] Just, J. E., Reynolds, J. C., Clough, L., Danforth, M., Levitt, K. N., Maglich, R., Rowe, J., Learning Unknown Attacks – A Start, Recent Advances in Intrusion Detection – 5th International Symposium, RAID 2002, Zurich, Switzerland, Oct 2002 Proceedings.

[15] Yamamoto, Y., Khargonekar, P. P., Frequency Response of Sampled-Data Systems,

[16] Ptacek, T. H., Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection, Technical Report, 1998.

[17] Lee, W., Stolfo, S. J., Mok, K. W., A Data Mining Framework for Building Intrusion Detection Models.

[18] Bykova, M., Ostermann, S., Tjaden, B., Detecting Network Intrusions via Statistical Analysis of Network Packet Characteristics, Ohio University.

[19] Krugel, C., Toth, T., Kirda, E., Service Specific Anomaly Detection for Intrusion

Detection.

[20] Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Wee, C., Yip, R., Zerkle, D., GrIDS: A Graph based Intrusion Detection System for Large Networks. Proceedings of the 19th National Information Systems Security Conference. Vol-1, pages 361-370. National Institute of Standards and Technology, October 1996.

[21] Gilbert, A. C., Kotidis, Y., Muthukrishnan, S., Strauss, M. J., QuickSAND: Quick Summary and Analysis of Network Data, DIMACS Technical Report 2001-43, November 2001.

[22] Postel, J., Simple Mail Transfer Protocol, RFC 821, Information Sciences Institute, University of Southern California, CA, USA, August 1982.

[23] Crocker, D. H., Standard for the Format of ARPA Internet Text Messages, RFC 822, Department of Electrical Engineering, University of Delaware, Newark, DE, USA, August 1982.

[24] Borenstein, N., Bellcore, Freed, N. Innosoft, MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies, RFC 1521, Network Working Group, September 1993.

[25] Moore, K., MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text, RFC 1522, University of Tennessee, September 1993.

Appendix 1: Base64 Encoding

Suppose a binary sequence is to be transmitted through e-mail. A 24 bit sample of this sequence is given as

1 0 0 1 1 0 1 1 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0 1

In byte format (decimal values) this sequence represents

155 162 233

On dividing this sample into four characters of 6 bits, we get the sequence

1 0 0 1 1 0 1 1 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0 1

which correspond to decimal values

38 58 11 41

using the base64 alphabet from Table 1, these numbers are represented as

m 6 L P

the final message will be sent as 7-bit ASCII where the above characters are sent using the ASCII table of Appendix 2.

Appendix 2: US ASCII Chart

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

Appendix 3: Program for generating pattern from packet data

```
/*
Title: Program for generating pattern from Email attachment

Filename: classification.cpp
Date: 3/15/04
Author: Archis Raje
Language: C++
Compiler: Microsoft Visual C++
Version: 6.0

Program Description: The following program is used to find the frequency of data
        in a network packet which corresponds to an E-mail. The program reads from a
        file where captured network packet data is stored. This captured data
        corresponds to traffic on port 110 (POP). The following program isolates
        packets which represent an E-mail attachment and proceeds to find frequency
        of data in these packets.
Execution Command:
        C:\>classification <sp> filename
        where filename is the name of the file containing captured data.
        the captured data must be in libpcap format (.cap)
Output expected: The program generates an output file by the name Frequency.csv
        This output file has Microsoft Excel Comma Separated Value (.csv) format
        The data stored in this file is arranged in rows and columns
        A single row corresponds to a single network packet while columns correspond
        to values from 0 to 130 (sufficient to cover the base64 alphabets)
*/

#include<iostream.h>
#include<stdio.h>
#include<stdlib.h>
#include<fstream.h>

void main(int argc, char *argv[])
{
        FILE *output_file;

        unsigned int frequency[1000][130] = {0};
        unsigned char * buff;
        const char * filename = argv[1];
```

```
	int i, j = 0, k;
	int next_packet = 40;		// beginning of ethernet header of next packet
	int start_packet = 40;		// point where ethernet header starts
	int packet_length = 0;		// length of TCP + IP headers and payload
	int tcp_hdr_length = 0;			// for cases when TCP header size > 20 bytes
	int start_tcp_data = 0;		// point where payload starts
	long size;
	int packet = 1;

	bool DATA;

	// beginning of paylaod
	bool start_data(int start_tcp_data, unsigned char * buff);

	// following block does initial file operations. the file specified in program
	// argument is opened for reading. the total file size is found and the entire
	// contents are read into a buffer.
	ifstream file (filename, ios::in|ios::binary|ios::ate);
	size = file.tellg();
	file.seekg (0, ios::beg);
	buff = new unsigned char [size];
	file.read (buff, size);
	file.close();

	// this while loop covers one single packet.
	while(next_packet < size)
	{
		packet_length = (int)buff[start_packet + 16]*256 + (int)buff[start_packet
+ 17];  // first the total length of current packet is found
		next_packet += (packet_length + 30);		// position of next packet is
noted.
		tcp_hdr_length = ((int)buff[start_packet + 46] >> 4) * 4;	// finds the
exact TCP header lenght as it may vary
		start_tcp_data = start_packet + 34 + tcp_hdr_length;// determine the
beginning of the packet payload
		if((next_packet - start_tcp_data) > 16)		// proceed only if the packet
is not empty. If the packet is empty, we proceed to next packet.
		{
			DATA = start_data(start_tcp_data, buff);	// determine if the
current packet belongs to the attachment of Email
			if(DATA)	// continue if the packet corresponds to Email
attachment
			{
				while(1)
				{
```

```
                                        start_packet = next_packet;    // initialize next
packet. as this will be the packet where data corresponding to the attachment will begin.
                                        packet++;

                                        // find packet length, position of nect packet, TCP
header length and beginning of next packet.
                                        packet_length = (int)buff[start_packet + 16]*256 +
(int)buff[start_packet + 17];
                                        next_packet += (packet_length + 30);
                                        tcp_hdr_length = ((int)buff[start_packet + 46] >> 4)
* 4;
                                        start_tcp_data = start_packet + 34 + tcp_hdr_length;

                                        // exit from the loop if the packets corresponding to
the attachment are finished
                                        if(buff[start_tcp_data + 1] == 0x4f &&
buff[start_tcp_data + 2] == 0x4b)
                                                break;

                                        // traverse through payload of a single network
packet and determine the frequency of sample values corresponding to base64 alphabet.
                                        for(i=start_tcp_data;i<(next_packet-16);i++)
                                                for(i=0;i<size;i++)
                                                        frequency[j][buff[i]]++;
                                        j++;    // increment packet counter
                                }
                        }
                }
                start_packet = next_packet;    // set start of next packet in the file
                packet++;                                       // increment packet counter.
        }

        output_file = fopen("frequency.csv","w");    // open output file to store data

        // the first 'for' loop corresponds to a single packet sample.
        for(i=0;i<j;i++)
        {
                // the second 'for' loop corresponds to all the base64 alphabets in a single
packet
                for(k=0;k<130;k++)
                {
                        fprintf(output_file, "%d", frequency[i][k]);
                        fputc(44,output_file); // a single character corresponding to "," is
inserted to separate each sample frequency.
                }
```

```
                fputc(13,output_file); // a single character corresponding to "line feed" is
inserted to separate data for each single packet.
        }
        fclose(output_file);    // close the output file
}

// following function determines whether the current packet is the beginning of
// an attachment or part of other POP communication. the function returns 'true'
// if the packet corresponds to attachment. else it returns a false.
bool start_data(int start_tcp_data, unsigned char * buff)
{
        if(buff[start_tcp_data + 1] == 0x4f && buff[start_tcp_data + 2] == 0x4b)
        {
                if((buff[start_tcp_data + 16] == 0x0d && buff[start_tcp_data + 17] ==
0x0a) ||
                        (buff[start_tcp_data + 17] == 0x0d && buff[start_tcp_data + 18]
== 0x0a) ||
                        (buff[start_tcp_data + 18] == 0x0d && buff[start_tcp_data + 19]
== 0x0a))
                        return 1;
                else
                        return 0;
        }
        else
                return 0;
}
```