

國立交通大學

資訊科學與工程研究所

碩士論文

合 作 式 垃 圾 信 件 過 濾 系 統



A Collaborative Anti-Spam Email Filtering

研 究 生：黃威力

指 導 教 授：蔡文能 教授

中 華 民 國 九 十 五 年 七 月

合作式垃圾信件過濾系統
A Collaborative Anti-Spam Email Filtering


研究生：黃威力

Student : Wei-Li Huang

指導教授：蔡文能

Advisor : Wen-Nung Tsai

國立交通大學
資訊科學與工程研究所
碩士論文



A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science

July 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年七月

合作式垃圾信件過濾系統

國立交通大學資訊工程學系（研究所）碩士班

摘 要

在網際網路上，電子郵件已經成為了最重要的應用軟體之一，最近這幾年，網路的管理者及使用者花了很大的心力在對付惱人的問題，那就是垃圾信件，垃圾信問題的嚴重性在於發送垃圾信的人只要花很低得成本就能夠從垃圾信上獲得相當大的利益。

在本論文，我們設計且建立了 CASEF 系統在路由的階段來過濾垃圾信件，我們的方法主要是利用垃圾信會在短時間內發送大量且相同信件的特性，除此之外，我們也建立了垃圾信誘捕系統來幫助 CASEF 系統，我們的 CASEF 系統不僅能在路由的階段來阻擋相同的垃圾信也比 SpamAssassin 有較高的準確率。

English Abstract

A Collaborative Anti-Spam Email Filtering

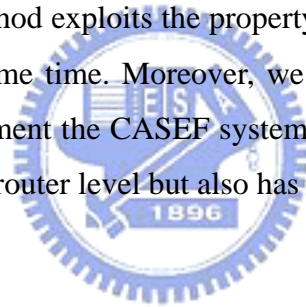
Institute of Computer Science and Information Engineering

National Chiao Tung University

Abstract

Email has become one of the most important applications on the Internet. In recent years, system administrators and Internet users had tried their best effort to solve the annoying email spam problem. The problem is that spammers can send huge amount of junk emails with a low cost and then get a large reward from the spam.

In this thesis, we design and implement the CASEF system to filter the spam at the router level. Our proposed method exploits the property of the spam that is distributing lots of identical spam mails at the same time. Moreover, we also implement an email honeypot to capture the spam mails to augment the CASEF system. Our CASEF system not only defends the repeated spam mails at the router level but also has a higher accuracy than SpamAssassin.



誌 謝

經過兩年的努力，終於完成了我的畢業論文，中間經過了許多的困難與問題，也受到許多人的幫助，在此一一感謝。首先要感謝的是我的指導教授—蔡文能教授，在碩士班兩年的期間，他給我許多方面的指導，讓我受益良多，也成長了不少。除此之外，也感謝家人給我一個平穩的環境及在背後支持著我，讓我能夠專心學習。最後要感謝的是實驗室的學長姐、同學們，大家一起工作、學習，互相幫忙，讓我有個難忘的碩士回憶。



Table of Contents

Chinese Abstract.....	iii
English Abstract.....	iv
Acknowledgment.....	v
Table of Contents.....	vi
List of Figures.....	viii
List of Tables.....	x
Chapter 1 Introduction.....	1
1.1 Motivation.....	1
1.2 Objective.....	2
1.3 Synopsis of the Thesis.....	3
Chapter 2 Background Knowledge.....	4
2.1 Internet Email Standards.....	4
2.1.1 Mail Messages Format and Encoding.....	7
2.1.2 Simple Mail Transport Protocol (SMTP).....	12
2.2 Spam.....	18
2.3 Spamming techniques.....	19
Chapter 3 Related Work.....	25
3.1 Controlling Spam Emails at the Routers.....	25
3.2 Email Honeypot.....	27
3.3 SpamAssassin.....	29
Chapter 4 Our CASEF System.....	31
4.1 System Overview.....	31
4.2 System Architecture.....	33
4.2.1 Email Honeypot Module.....	33
4.2.2 Collaborative Anti-Spam Email Filtering.....	35

4.2.3	Policy Filtering Module.....	37
4.2.4	Digest Matching Module.....	39
4.2.5	Spamassassin Filtering Module.....	41
Chapter 5	Experimental Results.....	42
5.1	Experimental Environment.....	42
5.2	Performance and Overhead	43
5.2.1	Digest Matching Threshold	43
5.2.2	Four Caches Sizes.....	45
5.2.3	Spamassassin Accuracy	48
5.2.4	CASEF System Accuracy	49
Chapter 6	Conclusion.....	52
6.1	Conclusion and Discussion.....	52
6.2	Future Work.....	53
Reference	54



List of Figures

Figure 1 Average global ratio of spam in email scanned by MessageLabs.....	2
Figure 2 Spam by Type, March 2006	2
Figure 3 The structure of the header and body	5
Figure 4 A simple email message	7
Figure 5 The communication model of the SMTP design.....	13
Figure 6 Synchronous Dialoging in SMTP	13
Figure 7 Invisible text examples.....	21
Figure 8 Blank HTML example source.....	21
Figure 9 Invalid HTML tags source	21
Figure 10 Letter spacing example	22
Figure 11 HTML comments source.....	22
Figure 12 MIME segment example.....	22
Figure 13 Vertical slicing example	23
Figure 14 JavaScript example	24
Figure 15 URL encoding examples	24
Figure 16 Infrastructure of the e-mail honeypot environment	28
Figure 17 CASEF System overview.....	32
Figure 18 The process of email honeypot	34
Figure 19 The process of the mails from two folders to two caches.....	35
Figure 20 The architecture of the collaborative anti-spam email filtering	36
Figure 21 The detail mail process flow	37
Figure 22 The move of the four caches	40
Figure 23 The first experimental result of digest matching.....	44
Figure 24 The second experimental result of digest matching	44
Figure 25 The third experimental result of digest matching	44

Figure 26 The fourth experimental result of digest matching 45

Figure 27 The experimental result of first cache 46

Figure 28 The experimental result of second cache 47

Figure 29 The experimental result of third cache 47

Figure 30 The experimental result of fourth cache 48

Figure 31 The accuracy of the first experiment 50

Figure 32 The accuracy of the second experiment 51



List of Tables

Table 1 Internet Standards related to Electronic Mail	6
Table 2 The RFCs of MIME.....	9
Table 3 MIME RFCs	9
Table 4 Standard Header Fields.....	11
Table 5 ESMTP common commands	15
Table 6 First Digit of Reply Code	17
Table 7 Second Digit of Reply Code.....	17
Table 8 The scenario of digest matching module	40
Table 9 The scenario of spamassassin filtering module	41
Table 10 Experimental environment.....	42
Table 11 The first experiment of Spamassassin.....	49
Table 12 The second experiment of Spamassassin.....	49
Table 13 The first experiment of the CASEF system.....	50
Table 14 The second experiment of the CASEF system	51
Table 15 Compare the CASEF system to Spamassassin only	53

Chapter 1 Introduction

Today, email has become one of the most important applications on the Internet. The greatest characteristics of the email systems are free and convenience. Those are the main reasons why everyone continually uses email to communicate with each other. Even though Instant Message and Voice over IP are hot applications, emails are still used everyday around the world. Because email is free and convenient, it is often used to send junk mails. In recent years, system administrator and Internet users had tried their best effort trying to solve the annoying junk email problem.

1.1 Motivation



Every day, millions of people lose valuable time trying to find their much needed emails because of the unsolicited commercial emails. In addition, sending these unwanted emails also costs businesses billions of dollars in wasted bandwidth. Spam is defined by some researchers as unwanted junk mail from a stranger that is sent in bulk to large mailing lists, usually with some commercial objective. The problem is that spammers can send junk emails with a very low cost and gain a large reward from spamming. More than two-thirds of the emails in the world are considered as spam: Messagelabs [9] finds spam by scanning email from its global network of control towers. The average global ratio of spam in email scanned by messagelabs is 70% in 2005 [10](see Figure 1). The top three of spam type are financial, commercial products and health according to Symantec reports [11] in March 2006 (see Figure 2). Those three types combined sum up to almost 60% of all spam mails. Due to this, we have already designed and implemented an anti-spam email filtering system [1].

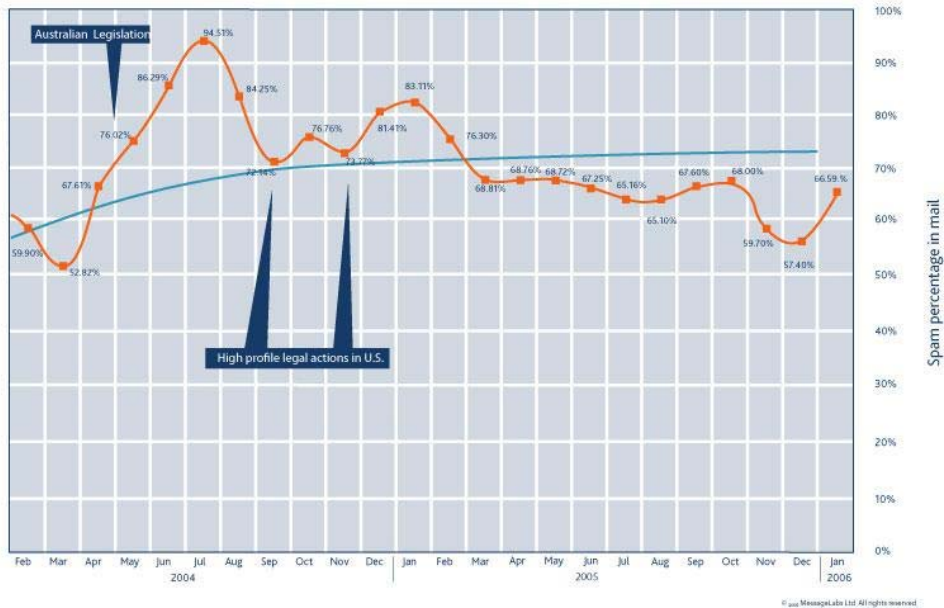


Figure 1 Average global ratio of spam in email scanned by MessageLabs

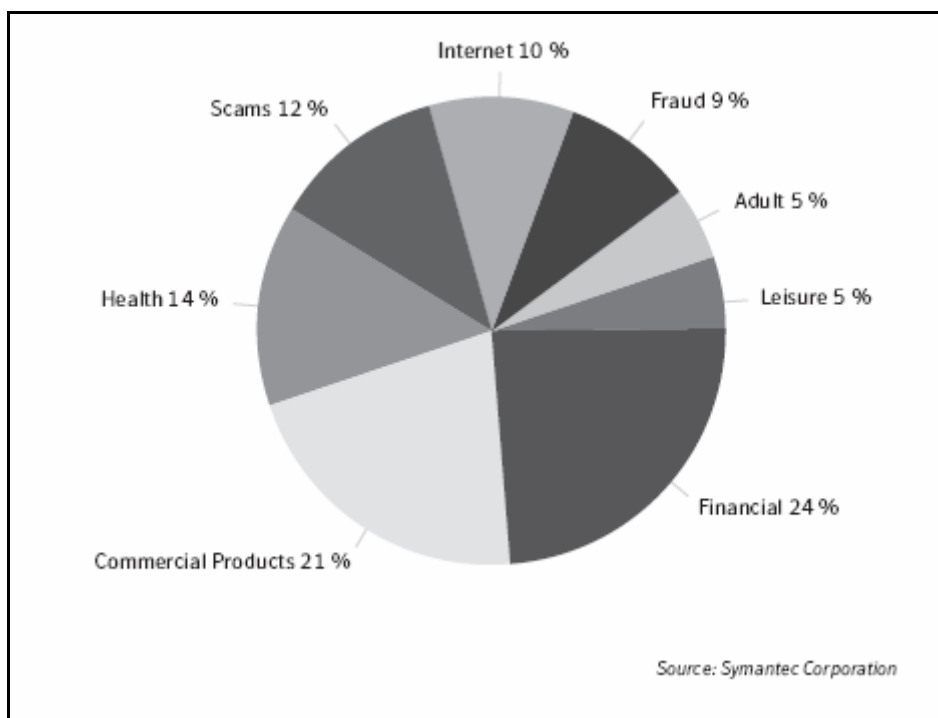


Figure 2 Spam by Type, March 2006

From the above reports, we can clearly point out that spammers like to focus on several kinds of topics so that identical spam emails always flood mailboxes at one time. Therefore, the challenge here is not only to save the resources of the network but also to protect the end users.

1.2 Objective

In this thesis, we designed and implemented a collaborative anti-spam filter with honeypot. At the router level, it can filter the emails before they arrive at mail servers of a company or any organization. Using the property that spam mails are sent to the mail box at the same time, our system has an email honeypot that can be used to collect the spams to assist the collaborative anti-spam filter. We use SpamAssassin, popular open-source software for detecting spam mail, to be the base of our anti-spam system. The system also has a flexible policy framework that the administrator can adjust through a user-friendly interface. Another advantage of the system is that it does not increase the ratio of false positive.

The goals of our system presented in this thesis are as follows:

- Defining Spam at Router level
- Email Honeypot
- Collaborative Anti-Spam Filtering
- Policy Filtering
- Administrator Management



1.3 Synopsis of the Thesis

The rest of this thesis is organized as follows: Chapter 1 introduces the motivation and objective of designing the collaborative anti-spam filtering system. In Chapter 2, we briefly describe the background knowledge of Internet Email Protocols [12], including SMTP and Internet Message Format. Chapter 2 also delivers an introduction of spam and spam technologies. The related works of our system are presented in Chapter 3. Chapter 4 gives details about the implementation and architecture of our collaborative anti-spam filtering system. Chapter 5 explains the experimental settings and results. Finally, chapter 6 presents our conclusions and outlines avenues for future work.

Chapter 2 Background Knowledge

In this chapter, we will briefly discuss the background knowledge of Internet email protocols, spam and spam technologies. In section 2.1, we introduce the standards and fundamentals of Internet email, including SMTP and Internet Message Format. In section 2.2, we describe the history and definition of unsolicited commercial email. In section 2.3, we briefly illustrate the technologies of spam.

2.1 Internet Email Standards

Email system is a set of mechanisms designed to allow messages to be sent from one computer to another. A user uses a program to write their email and send it to one or more addresses. When the message is delivered to its final destination, the recipient uses a program to receive the mail and view it. Although this process is purely simple, it contains a relatively sophisticated set of protocols, standards, and conventions under the mail transaction.

An email message consists of an envelope, a header, and a body. The envelope includes the source and destination addresses of the message. The header contains a lot of information about the message, i.e. the sender's address, the recipient's address, subject. The body contains the content that the sender wants to present or share. Of these three components, the header and body are the only ones that most users can use their mail reader to see; the envelope is generally used only internally. The overall structure of the header and body is conceptually simple, as illustrated in the following example (see figure 3).

```
From: test@test.example.com
To: hello@test.example.com
Subject: An email test
Date: Tue, 16 May 2006 14:02:57

This is a test.
Can you see that?

--
thx,
Test
```

Figure 3 The structure of the header and body

An email message by itself is just the content that the sender wishes to share. There needs to be a way to move it from the sender to the recipient. This process is divided into several tasks. An MTA (Mail Transport Agent) is designed to route mail, an MDA (Mail Delivery Agent) is designed to deliver the mail on behalf of an MTA, and an MUA (Mail User Agent) provides an interface for the user to read/write mail. These tasks are typically completed by different programs. Although some email programs can perform more than one function, it is a good practice to keep a mental distinction between them. The definitions of these elements are presented in the following paragraphs.

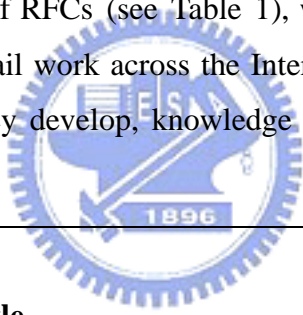
- (1) Mail Message: A mail message is consisted of two parts. The first part is the message body, which is the information a user wants to send to the recipient over the Internet. It may also include the files that are attached to the message. The second part is the administrative data for indicating the recipient's address and transport medium. The sender uses MUA to send a message out.
- (2) Mail Transport Agent (MTA): The Mail Transport Agent (MTA), also known as a mail server, or mail exchange server, is a server program or a software agent that receives message from an MUA or another MTA. The MTA does the task of mail routing. While MTA receives each message, the MTA decides where and how the message should be routed. If it is necessary, MTA will rewrite the address of mail.
- (3) Mail Deliver Agents (MDA): When an MTA has received a message, processed it, and decided where to route it, the MTA hands it off to an MDA. The MDA is designed to deliver a message to another location, which could be another MTA, a user's inbox, or a program

that performs some special task. If an MTA determines a message should be routed to another MTA, it hands the message over to an MDA that performs SMTP (Simple Mail Transfer Protocol), which defines a set of commands to transfer a message to a remote MTA. This MDA is often built into the MTA.

- (4) Mail User Agents (MUA): While MTAs and MDAs are designed for routing and transporting mail messages, Mail User Agents are designed to provide a management interface for users to manage their mail. This management interface generally includes viewing messages, managing mail folders, composing and sending new mails, as well as replying to messages and forwarding existing messages to other recipients.

MTA is the most important one among these components. MTA handles all the work of email. The major contribution of MTA is that it tells the other parts how to interact and what to do. It is the connection between all the components in the emailing process.

A set of Internet applications, protocols, and standards define how email operates. These standards are defined in a set of RFCs (see Table 1), which consist of several protocols and formats necessary to make email work across the Internet. In order to understand how these RFCs are created and how they develop, knowledge about the standard making process is needed.



RFC Number	Title
821	Simple Mail Transport Protocol(SMTP)
822	Format of Electronic Mail Messages
1049	Content-type Header Field
1869	SMTP Server Extensions
1870	SMTP Server Extension for Message Size

Table 1 Internet Standards related to Electronic Mail

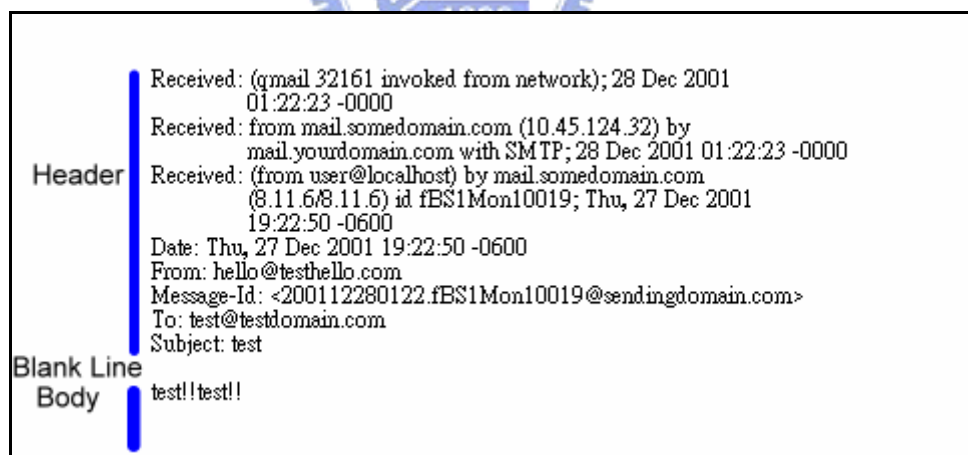
2.1.1 Mail Messages Format and Encoding

The base of email is the mail message, around which nearly every aspect of email revolves. Understanding email requires a thorough understanding of how messages are structured. The main structure of email is defined primarily in two RFCs, both of which are needed readings for anyone doing anything with the technical side of email.

RFC 822 [13][20], published in 1982, is the current standard for Internet email messages. It defines how mail messages are to be formatted when being transmitted from one location to another. The purpose is to provide a standardized format for mail messages so that different types of networks can transfer email from host to host. RFC 1123 [14] (Requirements for Internet Hosts—Application and Support), published in 1989, contains important updates to many Internet standards.

RFC 822 mainly defines how the message headers and body should be presented. It also defines the basic structure of a mail message. Although RFC 822 has been extended several times by later RFCs, the basic structure remains the same.

The sample message below (see Figure 4) demonstrates the format of an Internet email message that has been transmitted through many mail systems.



```
Received: (gmail 32161 invoked from network); 28 Dec 2001
01:22:23 -0000
Received: from mail.somedomain.com (10.45.124.32) by
mail.yourdomain.com with SMTP; 28 Dec 2001 01:22:23 -0000
Received: (from user@localhost) by mail.somedomain.com
(8.11.6/8.11.6) id fBS1Mon10019; Thu, 27 Dec 2001
19:22:50 -0600
Date: Thu, 27 Dec 2001 19:22:50 -0600
From: hello@testhello.com
Message-Id: <200112280122.fBS1Mon10019@sendingdomain.com>
To: test@testdomain.com
Subject: test

test!!test!!
```

Figure 4 A simple email message

The message begins with a set of lines called the header. The header contains information such as who sent the message, whom the message is being sent to, when the message was sent, and what the message is about. The format of the header is designed to allow programmed parsing of the header data. This enables MTAs, MDAs, and MUAs to analyze and perform on information provided in a mail message.

The header is followed by a blank line to divide the header from the remainder of the message. The remainder of the message is a collection of lines called the body. The body is the actual information to be relayed to recipients of the message.

The Body

The body will be presented first, since it is simpler than the header. The body conveys the actual message sent to a recipient. Pure RFC 822 message bodies are only a series of text lines, having no structure or meaning imposed on them. Because additional structure is sometimes necessary, several conventions have developed over the years, outside the scope of the message format standard.

One of the conventions that gained widespread use early was the use of encoding programs to convert binary data into a format safe for email transport. A common technique was to use a program such as *Uuencode* to encode the binary data into printable US-ASCII characters.

RFC 822 messages cannot handle these additional capabilities in their basic form. Because the characters available for RFC 822 messages are limited to US-ASCII, not every characters used in many non-English languages are supported. Recognizing this drawback of RFC 822 enables us to create extensions to cope with the limitation.

MIME [21][22] provides a way to overcome RFC 822 limitations. Some might claim that MIME extensions are one of the primary reasons for the popularity of email. In 1992, MIME was newly added to the world of Internet email standards with a set of RFCs supporting its first definition (see Table 2).

RFC Number	Title
1341	MIME: Mechanisms for Specifying and Describing the Format of Internet Message Bodies
1342	Representation of Non-ASCII Text in Internet Message Headers
1343	A User Agent Configuration Mechanism for Multimedia Mail Format Information

1344	Implications of MIME for Internet Mail Gateways
------	---

Table 2 The RFCs of MIME

RFC 1521 (MIME Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies) and RFC 1522 (MIME Part Two: Message Header Extensions for Non-ASCII Text) were published as updates to the previous RFCs in 1993. They provided minor clarifications and corrections. Finally, a series of RFCs was published in 1996 as another round of clarifications and corrections (see Table 3).

RFC Number	Title
2045	MIME Part One: Format of Internet Message Bodies
2046	MIME Part Two: Media Types
2047	MIME Part Three: Message Header Extensions for Non-ASCII Text
2048	MIME Part Four: Registration Procedures
2049	MIME Part Five: Conformance Criteria and Examples

Table 3 MIME RFCs

In addition to making data and binary files conversion, *Uuencode* and MIME can also be used to relay extended characters found in Asian and European dialects. A message may be shown in English, when it does not use any extended ASCII characters found in non-English languages. Yet, many countries use extended characters that are not included in the 7-bit ASCII character set. Due to *Uuencode* and MIME, it becomes possible to encapsulate these messages into a form that is suitable to be transmitted across the Internet.

The Header

The lines of a header are divided into fields, which provide information intended for both users and programs. Each field is composed of one or more lines of text. Some fields contain more than one line; the additional, continuation lines start with whitespaces.

Each header field is consisted of a field name, optional whitespace, a colon, optional commented folding whitespace, and an optional field body. The field body can also include leading whitespaces.

```
field = field-name<SP>”:”[SP][field-body]CRLF
```

To clear up syntax descriptions, future examples will not show the trailing CRLF, except where it is fundamental to an understanding of a particular item.

The following example illustrates a typical field.

```
Subject: Hello everyone!! Nice to meet you!!
```

The field-name includes a sequence of any printable US-ASCII characters, excluding the space character or ‘:’. Most field names compose of a series of alphanumeric characters, often interspersed with dash characters.

```
Form:  
Status:  
Subject:  
X-Mailer:
```

The standard headers are detailed in RFC 822. Most of these fields are common, and are present in most email systems. RFC 822 defines a standard set of fields for mail messages (see Table 4).

Field name	Description
From	The creator of the message
Sender	The sender of the message
Reply-To	The address to send replies to
To	Primary recipients of the message
Cc	Secondary recipients of the message

Bcc	Blind Carbon Copy recipients of the message
Message-ID	The message's unique identifier
In-Reply-To	The message being replied to
References	All messages ancestors
Date	The date the message was created
Received	MTA footprint
Return-Path	The address of the originator
Subject	The subject of the message
Comments	Miscellaneous comments regarding the message
Keywords	Topical keywords related to the message
Encrypted	Encryption information (obsolete)
Resent-*	Fields created when redistributing
X-*	Extension fields

Table 4 Standard Header Fields

Among the twenty fields defined in RFC 822, very few header fields are actually needed. For example, the message must indicate the date it was created on, using a Date or Resent-Date field. The message must also point out a mailbox for the person or the program creating the message, using the From field. A recipient field is also required, and can be To, Cc, or Bcc field, or its Resent -* equivalent.

Other than the required header fields, all MTAs that processed the message must add a Received line at the start of each header. A recipient can then know the delivery path of the message when looking at the message header.

With very little exceptions, the fields in a header are not required to be in any specific order. The Received, Return-Path, and Resent-* fields are the exceptions. When a message passes through a set of MTAs, each MTA adds a Received field to the message. The Return-Path is added by the final MTA prior to the final delivery. Because these fields provide tracing information for analyzing problems, their placement must not be altered. Any Resent-* fields should also be added at the beginning of the message. RFC 822 recommends the following order.

Date
Form
Subject
Sender
To
Cc
...

In practice, there is neither necessary nor required to follow this order. In fact, browsing through an arbitrary collection of mail messages shows that the order varies widely. While MTAs, mailing list processors, and MDAs insert fields that are typically appended, the rest of the fields typically do not have a fixed ordering

2.1.2 Simple Mail Transport Protocol (SMTP)

At the highest level, SMTP is designed as a communication protocol between two machines—a host and another host. In its simplest form, a user sends a message from a user on one machine to a mailbox on another machine. In the early 1980s, SMTP started becoming widely used, and the format that includes mail message format and SMTP protocol is defined in RFC 821 [15][19] and RFC 822 in 1982. When Internet technologies developed, the protocol has grown and changed rapidly. More protocol extensions [16] were developed, and were known as the Extended Simple Mail Transport Protocol (ESMTP). SMTP and ESMTP became the backbone of the Internet Mail System. We use SMTP to refer to the basic protocol without service extensions and use ESMTP to refer to SMTP with other service extensions.

When MUA wants to send mail messages to MTA, it needs to use SMTP protocol. In fact, SMTP protocol is built-in in most MTAs. MTAs can use it to transfer email to other MTAs. MTAs that provide TCP/IP-based SMTP service, such as through sendmail, listen on port 25 (SMTP).

SMTP is a store-and-forward protocol, meaning that it can send a message through a set of servers in order to deliver the message to the destination. Incoming messages are stored in a queue of the server. And the server attempts to transmit them to the next host.

Figure 5 shows the communication model of the SMTP design

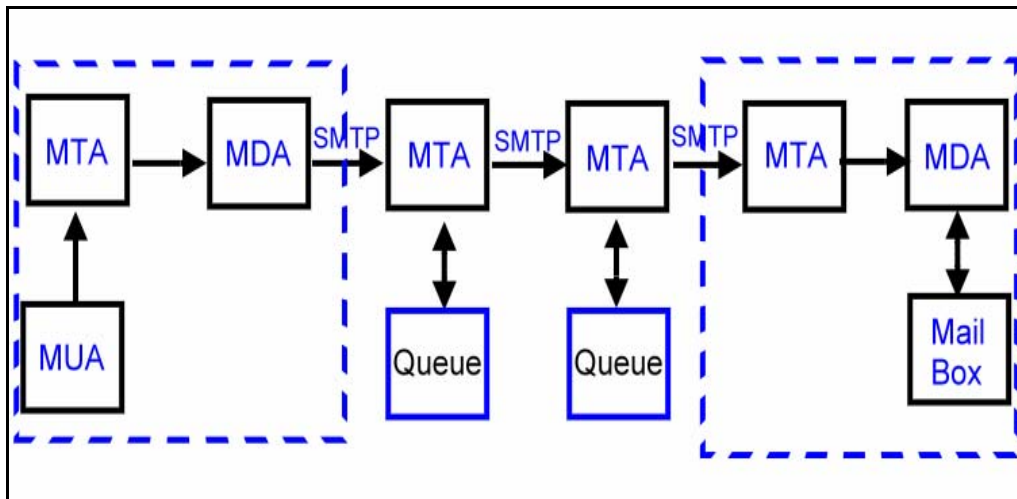


Figure 5 The communication model of the SMTP design

SMTP is a command-based, text-oriented protocol. The client makes a request to the server to execute a command. The server handles the request and sends the reply back to the client. The process continues until the session is terminated. The process is shown in the following figure (see Figure 6).

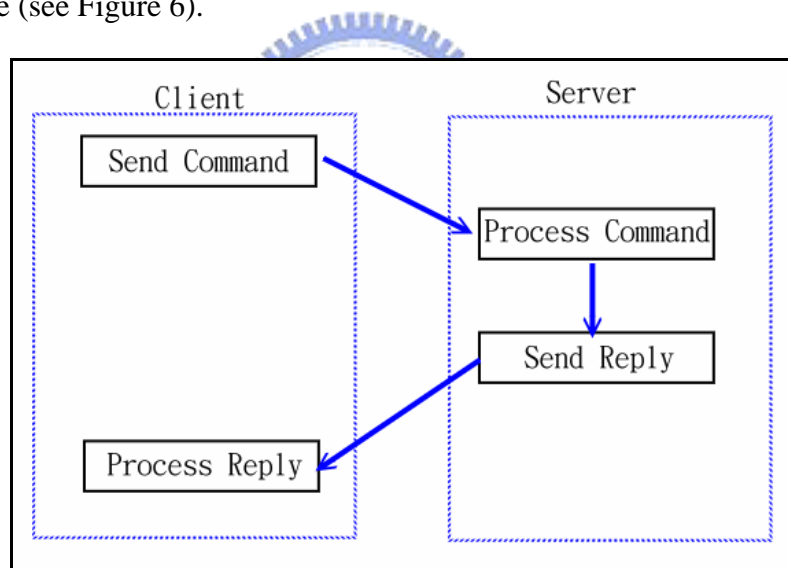


Figure 6 Synchronous Dialoging in SMTP

We will describe the SMTP commands and SMTP replies that are used in an SMTP session in the following section.

SMTP session

An SMTP session can be divided into several phases:

1. Determining which machine to connect to
2. Establishing a connection

3. Initializing the session
4. Performing various transaction dialogs
5. Shutting down a session

When a SMTP mail session begins, it will send a "HELO" command. When the SMTP mail session wants to terminate, it will send a "QUIT" command. We need to use the following three commands, "MAIL FROM:", "RCPT TO:", and "DATA", at least once when sending out emails. Although many other commands can be used in the SMTP session, these five commands are used much more frequently than any others.

SMTP Commands

A small SMTP server must support, at the minimum, the commands including "HELO", "MAIL", "RCPT", "DATA", "RSET", "NOOP", and "QUIT". Besides, all other commands are extensions anyway. The ESMTP server must include the EHLO command at least. The server uses the EHLO command to query the other server whether it supports ESMTP commands or not. The commands are listed in the following Table (see Table 5).

Command	Description
HELO	Format: HELO <SP> <domain> <CRLF> Used to identify the client, and stored it in the "Received:" message header.
EHLO	Format: EHLO <SP> <domain> <CRLF> ESMTP replacement for HELO. Used to identify the client and request a list of ESMTP service extensions supported by the server.
MAIL	Format: MAIL <SP> FROM:<sender-address> [DSN parameter] <CRLF> Used to identify the message sender, and stored in the message envelope. It is not necessary for the information in this field to be the same as that in the "From:" field in the message header. The data in the "From:" field is provided in the message headers, which are a part of the message proper.
RCPT	Format: RCPT <SP> TO:<receiver-address > [DSN parameter] <CRLF>

	<p>Used to specify the recipient of a message, and is also stored in the message envelope.</p> <p>This does not have to be the same information as that in the "To:" or "CC:" fields in the message header. Because the envelope is used by the local delivery agent, the information provided in "RCPT TO:" dictates the destination mailbox for the message, regardless of the information provided in the "To:" or "CC:" fields in the message header.</p>
DATA	<p>Format: DATA<CRLF></p> <p>Used to mark the beginning of the mail message being sent.</p> <p>Any data sent after the "DATA" command is assumed to be the message body. The message body will contain headers created by other mail servers that have processed the mail, and will also contain the original message body and any attachments. The end of the message is marked with a dot (.) surrounded by a pair of carriage-return/line-feeds. Once the end-of-message string has been acknowledged, the SMTP client can either create another mail message or can close the connection using the "QUIT" command.</p>
RSET	<p>Format: RSET<CRLF></p> <p>Used to reset the server and to nullify the entire message transaction.</p>
NOOP	<p>Format: NOOP<CRLF></p> <p>No operation.</p>
QUIT	<p>Format: QUIT <CRLF></p> <p>Used to terminate the SMTP session and closes the mail connection.</p> <p>Rather than the client simply closing the TCP connection, it uses the "QUIT" command to request that the server close the connection.</p>

Table 5 ESMTP common commands

SMTP Reply

After the SMTP server processes a command, it sends a reply back to the client. The primary purpose of the reply is to indicate the success or failure status of the request. The SMTP client needs to wait for a valid response code from the server before sending more

commands or data. It must ensure the integrity of the message transfer. The reply message is designed for both machine and human to understand. Replies may appear in a single-line format or in a multi-line format. The format and an example for a single-line reply are shown below.

```
Reply = reply-code<SP>human-readable-text
```

The first portion of reply code is a 3-digit numerical. A single space character follows the numbers, and the remainder of the line is the human-readable portion. Some commands that impose constraints on the text format are always intended to be readable by humans. An example SMTP reply is shown below.

```
250 Requested mail action is successful, completed
```

The human-readable portion generally includes descriptive text associated with the reply number. The purpose of the text is to support users with a readable description of problems encountered. In fact, it is not designed to be machine-parsed.

RFC 821 supports standard text strings to go with each of the reply codes. In general, the standard text is used. The following example is the standard text for the 250 reply code. The text portion of reply code is often reduced to a simple acknowledgement of success.

```
250 ok
```

This is a perfectly acceptable difference from the standard text. Besides, client should be prepared to handle reply lines that do not provide the human-readable text or even the space character after the reply code. Although the space character and human-readable text are required, some servers have been known not to issue them.

SMTP replies may also appear in a multi-line format. The format is very similar to the single-line reply, except that the space character is replaced with a '-' on all but the last line. The '-' serves as an indicator to the client that more reply lines are coming. A client sending an EHLO command and the server sending a multi-line response are presented in the following example.

```
Ehlo test.example.com  
250-test.example.com  
250-size 2035983
```

250 ok

The server's status is shown by an SMTP three digit reply code:

1. The first digit: This digit indicates whether the response is good (1, 2), intermediate (3), or incomplete (4, 5).
2. The second digit: This digit indicates what type of errors has happened.
3. The third digit: This digit provides more detailed information about the error.

A more detailed description is shown in the following tables (see Table 6 & 7).

First Digit	Description
1	Positive preliminary reply
2	Positive completion status
3	Positive intermediate reply
4	Transient negative reply
5	Permanent failure

Table 6 First Digit of Reply Code

Second Digit	Description
0	Syntax
1	Responses to requests for additional information
2	Communications Channel
5	Mail system

Table 7 Second Digit of Reply Code

Using detailed response codes can almost guarantee the success of email exchange between the SMTP client and server. The SMTP client would terminate the connection to stop

the operation and return an error message back to the original sender if a response code could not be understood.

SMTP sends a mail message consisted of an envelope and the content. However, SMTP only knows how to exchange mail messages between hosts. It does not concern itself with the content of the message body, or even the message envelope. This is the reason behind its name, the “Simple” Mail Transport Protocol.

2.2 Spam

Spam is a term used to describe a large number of unsolicited messages irrespective of the content, though they usually contain information about some products or services. If the unsolicited email message is sent in huge volumes or is cross-posted to more than 20 mailing lists, the message is definitely regarded as spam.

Spam is also defined as Unsolicited Commercial Email (UCE) and Unsolicited Bulk Email (UBE). It first came into existence in the mid 1970s with the following incident. When an overzealous salesman at a large computer manufacturer sent a pure advertisement to many recipients, the recipient list could be longer than the message itself. The company apologized soon afterwards and made sure that the salesman would be educated in proper Internet etiquette.

A chain letter refers to a form of spamming when the spammer sends a single email to thousands and thousands of users and to encourage them to forward the letter to other users still. Spammers often use network resources to send unsolicited commercial emails. Spammers also use some techniques, such as email spoofing, to protect their addresses, so that the recipients cannot trace the original source. Some email systems, including Microsoft Exchange, can block incoming mails with a specific address. However, this technique along cannot prevent every spam from reaching your email inbox because these individuals change their email addresses frequently.

The IETF explains why spam is a serious problem in RFC 2635 [17]. RFC 2505 [18] provides more information to mail administrators on how to tackle spam. Furthermore, Geoff Mulligan published a book about spam removal in March 1999 [2].

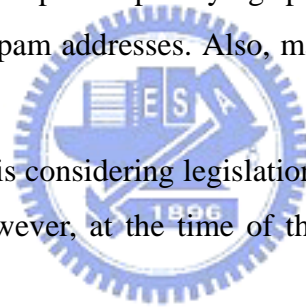
Spam differs in one major economic aspect from the junk mails of the postal system even if they are much alike. The receiver withstands more cost from spams than the sender. The cost for the sender to email one thousand, ten thousands, or a million email messages is little,

especially when mailing lists or relays are used. On the other hand, the cost of creating, printing, and distributing junk mails rest upon the sender and not the receiver. Typically, sending spam messages may cost only a few dollars. The content is usually an advertisement for some service or product. Even if only one person who received the advertisement decides to make a purchase, the revenue created will be more than enough to compensate the cost of sending the spam. Spam is very much like getting junk mails labeled with cash on delivery.

The low-cost e-mail spamming engines which provide salespeople with millions of e-mail addresses, together with the fact that sending large volume of emails does not incur much cost to the sender, has resulted in the current explosive growth of “Unsolicited email”. Nowadays, no legislations are powerful enough to stop spammers unless the spam mails contain illegal items.

Currently, there is no effective way for the consumer to “charge back” the spammers the cost coupled with reading, storing, transporting, and replaying their junk mails. In fact, a large number of systems are being set up to stop relaying spam messages and to block all messages originating from well-known spam addresses. Also, many states of America are now passing bills outlawing spam.

Presently, USA Congress is considering legislations to require the marking of unsolicited commercial email (UCE). However, at the time of this writing, the legislation is yet to be finished.



2.3 Spamming techniques

Before we can start implementing a good anti-spam filtering system to protect our mail systems and users, we must first understand the techniques used to send spam mails. Spamming techniques have evolved rapidly over the past few years. Increasingly complex techniques that cause damages to legitimate businesses make it tougher to prevent the spamming from happening.

In March 2004, Allister Cournane and Ray Hunt presented in their paper, titled “An analysis of the tools used for the generation and prevention of spam” [7], the problems created by spamming to e-mail and newsgroup users. Spamming is now believed to be a serious threat to the Internet in general and is especially a serious troublemaker to both ISP and network users. The motivation of, and the tools used to generate, spams are also presented in that paper. We describe in the following paragraphs the protection and prevention techniques mentioned

in Cournane and Hunt's paper.

The following list expresses some techniques used by most spammers.

(1) Obtaining E-mail lists

The methods used to acquire email lists are:

■ Common sources:

A spammer may extract email addresses from some public accessible sources.

- ✧ Usenet newsgroups.
- ✧ Web-based discussion bulletins.
- ✧ Internet chat rooms..

■ Renting lists:

Although some users may not provide real addresses on the common sources, spammers could rent email lists collected by someone else. For example, the following are some rental sites a spammer could take advantage of.

- ✧ <http://www.rent-a-list.com>
- ✧ <http://www.optinic.com>

■ Purchasing lists:

Instead of renting, some well-known websites have compiled cheap, large-volume mail lists available for purchase. Examples include:

- ✧ <http://www.e-mail2success.com>
- ✧ <http://www.75.jonrecommends>
- ✧ <http://www.horizon-place.com>

■ Email harvesters:

Applications exist that could scan HTML pages for email addresses.

(2) Spam obfuscation

As new techniques for detecting spam e-mails are developed, corresponding counter-techniques will soon be devised and integrated into the next version spam e-mail generation software. Some tricks that are used by spammers to obfuscate the spam mail filter include:

(a) Invisible text:

This technique attempts to hide legitimate texts within a spam message in order to give the appearance of a legitimate message. Figure 7 illustrates the example.

```
(a) 445992001 448223992 unanimously repetition 002348821 799340773 435893022
modem (noun) a device or program that enables a computer to transmit
data 344000232

(b) X-Mime-Key: search words: interface external telephone modem copper
modulate demodulate Internet e-mail 56k bandwidth binary signal voice
bps bits per second compression fax CCITT V.34 protocol

(c) <font color="white" size="-1">search words: interface external telephone
modem copper modulate demodulate Internet e-mail 56k </font>
```

Figure 7 Invisible text examples

(b) Blank HTML:

This technique uses HTML hyperlinks to point to the actual spam message. The email itself, however, contains no plain text. Figure 8 shows an example.

```
<html><div>
<a href="http://www.your-info-station.com/Sla/eb.php?x=52c">
</a></html>
```

Figure 8 Blank HTML example source

(c) Invalid HTML tags:

Since invalid HTML tags are ignored by web browsers, this is the main technique of hiding legitimate text within a spam message. An example is given in Figure 9.

```
< Despite statements last week from chief U.N. inspector Hans Blix that full
cooperation was expected from Iraq, Iraqi Foreign Minister Najji Sabri lashed
out at the United Nations in a 19-page letter to Secretary-General Kofi Annan
written in Arabic. In it, Sabri repeated previous claims that Iraq has no
weapons of mass destruction and that the inspections are just a false pretense
for the United States and Britain to attack his country. >
```

Figure 9 Invalid HTML tags source

(d) Letter spacing:

Separating each character by special delimiters, such as a whitespace, could fool some filtering systems that scan for certain keywords. Figure 10 shows an example.

```
(a)  U N I V E R S I T Y   D I P L O M A S
(b)  F * R * E * E   L - O - A - N - S   O ^ N ^ L ^ I ^ N ^ E
```

Figure 10 Letter spacing example

(e) HTML comments and blank tags:

Using HTML comments and blank tags to split possible key words is also an effective method that could prevent a filter from successfully identifying the key words. An example is given in Figure 11.

```
(a)  <HTML><BODY>milli<!-- xe64 -->onaire</BODY></HTML>
(b)  <HTML><BODY>milli<b></b>onaire</BODY></HTML>
```



Figure 11 HTML comments source

(f) MIME segment:

MIME segment technique refers to the placing of a legitimate message in the plain text segment (which is almost never displayed) and the spam message in the HTML segment. An example is shown in Figure 12.

```
-----=_NextPart_001_2D3DF_01C29D73.26716240
Content-Type: text/plain;

Short for modulator-demodulator. A modem is a device or program that enables a
computer to transmit data over, for example, telephone or cable lines. Computer
information is stored digitally, whereas information transmitted over telephone
lines is transmitted in the form of analog waves. A modem converts between
these two forms.

-----=_NextPart_001_2D3DF_01C29D73.26716240
Content-Type: text/html;
<p><b><font face=Arial>Earn thousands now from home, NO RISK INVOLVED! 1000%
return GUARANTEED!</font></b></td>
```

Figure 12 MIME segment example

(g) Vertical slicing:

Slicing a spam message into vertical strips, placing them within a HTML table cells, and formatting the table into human readable form are also a technique that could serve to confuse the key word scanner of an anti-spam filter, as illustrated in Figure 13.

```
(a) <table cellpadding=0 cellspacing=0 border=0>
<tr>
<td>
<table cellspacing=0 cellpadding=0 border=0><tr><td><font
face="Courier New, Courier, mono" size=2> H <br> T <br> S
</font></td></tr></table>
</td>
<td>
<table cellspacing=0 cellpadding=0 border=0><tr><td><font
face="Courier New, Courier, mono" size=2> E <br> H <br> P
</font></td></tr></table>
</td>
...
(b)
|H|E|L|L|O
|T|H|I|S|I|S|M|Y
|S|P|A|M|M|E|S|S|A|G|E
```

Figure 13 Vertical slicing example

(h) JavaScript:

Spam messages can be specially encoded in a JavaScript variable and then the variable is interpreted at runtime. An example of this method is shown in Figure 14.

```
<HTML><HEAD><SCRIPT LANGUAGE="Javascript"><!-- var Words = "
%3CHTML%3E%3CBODY%3E%3CHI%3EDo%20you%20w%62nt%20to%20e%62rn%20million%24%3F.%3C
/H1%3E%3CH2%3Eclick%20%3C%62%20href%3D%27http%3A//www.mywe%63sit%66.%64om%27%3E
here%3C/a%3E%20to%20find%20out%20more%21%3C/H2%3E%3C%63r%3E%3Cimg%20src%3D%27ht
tp%3A//www.myim%62gesour%64%66.%64om/im%62g%66.jpg%27%3E%3C/BODY%3E%3C/HTML%3E"
function SetNewWords() { var NewWords; NewWords = unescape(Words);
document.write(NewWords); } SetNewWords(); // --> </SCRIPT> </HEAD> <BODY>
</BODY> </HTML>
```

Figure 14 JavaScript example

(i) URL encoding:

The URL encoding technique is a method that encodes URL addresses in hexadecimal form to prevent anti-spam filters from identifying websites that were known to deliver spam mails. An example can be found in Figure 15.

```
(a) http://7763631671/obscure.htm
(b) http://0xCeBF9e37/obscure.htm
(c) http://0316.0277.0236.067/obscure.htm
(d) http://3468664375@3468664375/o%62s%63ur%65%2e%68t%6D
```

Figure 15 URL encoding examples

In order to implement an effective anti-spam filter, designers must read and understand the information on the tricks of email spamming, that include but are not limited to the use of complex HTML techniques and HTML encoding skills. Designers must also identify new spamming techniques as they appear.



Chapter 3 Related Work

In this chapter, we study several related works on anti-spam filtering techniques and products. Spam is a continually growing problem in the world and many solutions have been proposed. In section 3.1, we investigate techniques that detect spams at the router level. In section 3.2, we learn how to implement a honeypot to attract spam emails from spammers. In section 3.3, we give a brief description regarding SpamAssassin.

3.1 Controlling Spam Emails at the Routers

If we want to control spam mails, the best method of defense is to work directly at the recipient level. The entire SMTP session is visible between all routers used to transfer SMTP messages from one mail server to another. Any of these routers can close the SMTP session by transmitting a TCP reset segment to both hosts. Observing this, we have a chance to control spam mails at the router by monitoring all SMTP traffic and determining if the message stream contains a spam or a non-spam.

Banit Agrawal, Nintin Kumar, and Mart Molle, three professors from the Department of Computer Science & Engineering of University of California, have presented a paper at the IEEE International Conference on Communications (ICC 05) in Seoul Korea from 2005. Their paper [3] describes a mechanism for detecting and controlling spam mails at the router level. In general, spammers usually send spam mails to many people with essentially identical contents with only some minor alterations at the same time. The professors utilized this fact in their design and implemented their system at the router level. Their design is described briefly below.

When the router sees a SMTP's message, it will copy and redirect the message to another

system that does the spam classification. If the SMTP session is classified as to contain spam, the session will be blocked in constant time at the router. The router can also set a flag that is normally reserved in TCP header to the packet that contains the spam mail's header. The flag will then be checked by recipient's MUA when the recipient decides to check his/her mails. Following this process, the recipient can filter out the spam mails automatically.

In their system, they have implemented a two phase method to check spam at the router level. In the first phase, they used a pattern-matching approach to detect spam mails. If the system considers the mail as spam, it will not pass the mail to the second phase. Otherwise, the second phase uses a Bayesian classifier to try to categorize the same mail again as a spam or a non-spam. If either one of the two phases concludes that the mail is spam, the system will rate limit it at the router level.

The Boyer Moore algorithm was chosen to be their technique of pattern-matching. The Boyer Moore algorithm is much more efficient than other algorithms that used the method of brute force. In the first phase, a two-level cache was constructed that consists of a primary cache and a secondary cache that store all mails. The primary cache is implemented by LRU data structure and the secondary cache is implemented by FIFO data structure. When the mail arrives at the system the first time, the system will compare it with the mails in the secondary cache. If there is a cache hit, the mail will be promoted from the secondary cache to the primary cache and the mail will not be marked as spam. Otherwise, a copy of that message will be saved in the secondary cache. If the mail produces a cache hit in the primary cache, the mail will be flagged in its TCP header to indicate as spam mail. The mail's counterpart in the primary cache will be moved to the first position of the cache.

In the second phase, a Bayesian classification technique is used to classify mails. In general, a Bayesian classifier consists of two phases: Training and Testing. The classifier must be trained with a carefully constructed database prior to its commencement. Each training mail is reduced into a set of unique tokens. The tokens are then recorded and counted, and they will be given a score to evaluate the testing mail. In the testing phase, a score is computed for every mail, and a testing mail is judged as spam if its score is too high as determined by the limit chosen by the administrator.

Detecting spam at the router is an efficient approach to protect the user. Two-level cache is a good idea in detecting mail duplication which is one of the defining characteristics of spam. We will apply the concept of router level spam detection in our CASEF's design and

also incorporate multi-level caches to improve our system's performance.

3.2 Email Honeypot

Everyone knows valid email addresses are the most valuable resources of spammers. Nevertheless, little are known by the general public about the spammers' techniques in collecting and harvesting addresses and their capabilities. Some techniques such as generating email addresses by brute force and extracting valid email addresses from the Internet, organizations and websites are, however, widely known. Some examples are depicted in Section 2.3.

Guido Schryen, a doctor in the Institute of Business Information Systems, University of Technology Aachen (Germany), has published a paper at the Systems, Man and Cybernetics (SMC) Information Assurance Workshop from June 15-17, 2005, describing a spam honeypot project that includes a mechanism of placing email addresses on the Internet and analyzing received emails [4].

The purpose of the honeypot is to study spammers' behavior in harvesting email addresses from all kinds of services such as websites and newsgroups in the Internet. Thus, many email accounts must be created for different Internet services such as web pages, web chats, newsletters, ICQ, MSN, etc. Within each service also contains many kinds of topics that are grouped by types (administration, context, commerce and content). Finally, each type of topics is dissected into four addresses (de-, com-, net-, and org-address).

Schryen implemented a mail server for covering email addresses of the four top level domains. The mail server can be reached by the following URLs: *charlie.winform.rwth-aachen.de*, *wforasp.com*, *wforasp.net*, and *wforasp.org*. A mail generator is then used to create thousands of email addresses automatically. In order to prevent email addresses from being generated by a brute force application, it is necessary to add a little change to each mail. Giving an appropriate number of characters is the method such as *wasp10208@wforasp.com*. After that, thousands of emails are distributed to each service manually.

When an e-mail arrives at the honeypot's mail server (see figure 16), the mail parser, which is the first part of the system, is used to classify the mails into spam or non-spam mails. A white list is also included in the parser in order to filter out some known mails. After

parsing, the spam and non-spam mails are all stored in MySQL database. The database is later used by data mining tools and statistical analyzers.

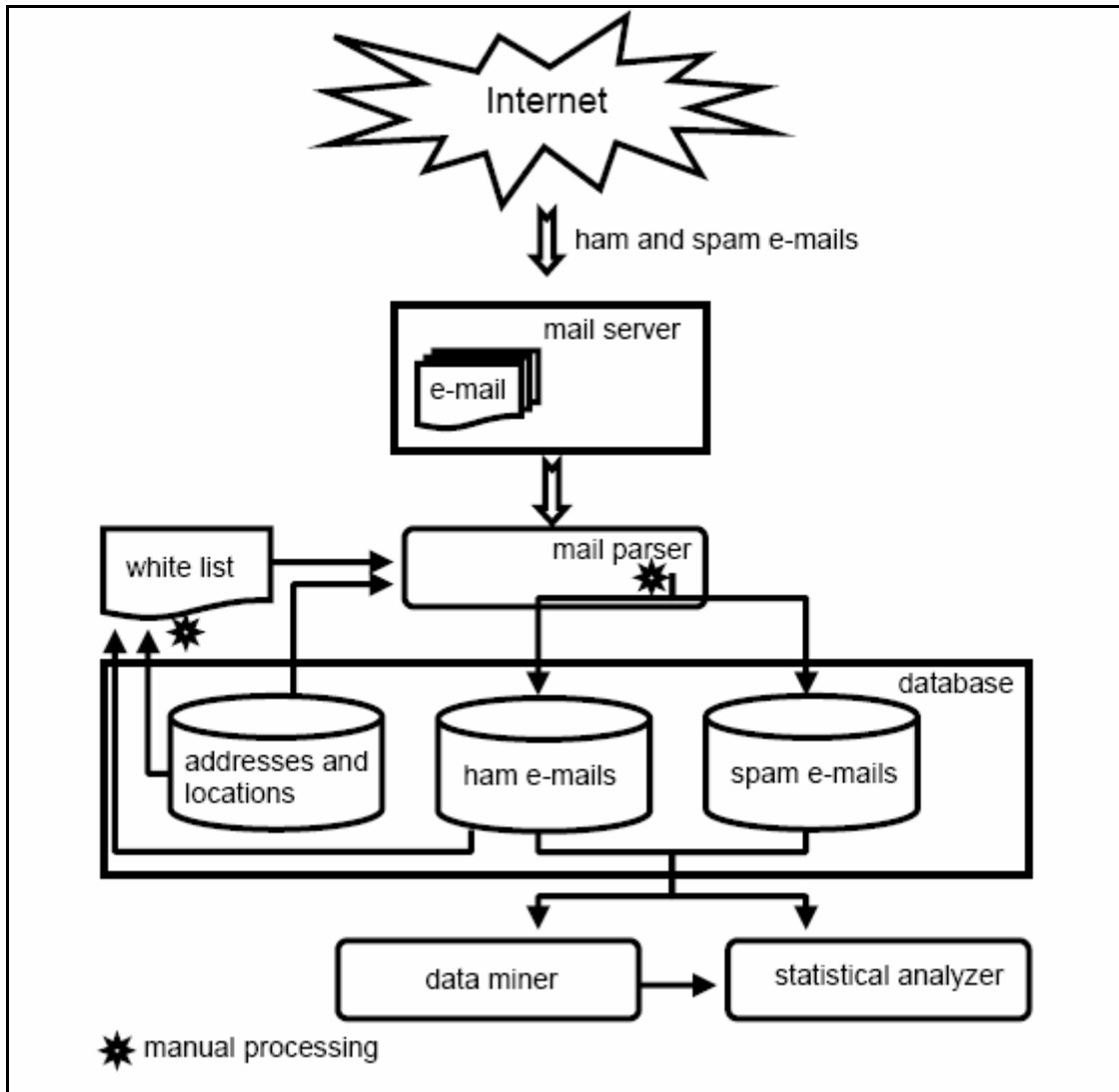


Figure 16 Infrastructure of the e-mail honeypot environment

A total of 7,968 non-spam emails and 2,482 spam emails had arrived at the mail server. Preliminary results of the honeypot project have shown that no spam have been sent to German newsletters/ mailing lists, and only a few spam emails arrive at the US newsletter/ mailing lists. Almost all spam mails originates from addresses posted on the web pages; spammers are especially interested in com-addresses. And spammers also focus a lot of their attention in sending spam mails with the types of administration, content and commerce.

Using honeypot to attract spam mails can help us understand spammers' behavior. Thus, we also want to use a honeypot to be a part of our system. Based on the result of Schryen's paper, our honeypot will focus on the areas of web pages, the mails with com-addresses and the types of administration, content and commerce.

3.3 SpamAssassin

The SpamAssassin [5] system is a software package that is designed to detect email messages, to categorize the mail messages into spam or non-spam, and to report the results back to administrators or users alike. The system is rule-based, meaning that spam identification is based on a set of rules. Each rule describes a relationship among the words of an email message that reduces to points to be added or deducted from the overall score. If an email message has a score exceeding the preset threshold, the message is reported as spam.

Numerous anti-spam systems are available today. Nevertheless, SpamAssassin has become popular for several reasons:

1. SpamAssassin derives its spam filtering ability from many different kinds of rules. The system can be improved by adding more effective rules and removing less effective ones. Rules shown to have better spam discriminating abilities are given higher scores.
2. Tuning the scores associated with each rule or adding new rules based on regular expressions is easy.
3. SpamAssassin can adapt to each system's unique email environment. The system can learn to recognize which senders to trust and to incorporate latest findings about spam.
4. SpamAssassin can report spams to several spam clearing houses. It also can be configured to generate spam traps—email addresses that are used only to forward spams to a clearing house.
5. SpamAssassin is free. It is distributed under either the GNU Public License or the Artistic License. The license permits users to openly modify the software and redistribute their modifications under the same terms.

Although a spam message could utilize several devices mentioned in Section 2.3 to deceive an anti-spam system, SpamAssassin can still recognize several suspicious characteristics and hence give a high score to the spam mail.

The SpamAssassin system core is composed of a collection of modules written in the Perl programming language. Moreover, SpamAssassin uses a Perl script to receive messages from the standard input and redirect them for the modules to check. To improve performance,

SpamAssassin also contains a daemonized version of the spam-checker. In addition, a client program written in C is included in the package that enables the daemon to process messages received from the standard input.

After SpamAssassin decides that an email is spam, that email is neither deleted nor filtered out. On the contrary, the email is marked with a tag on its subject line and is delivered to the recipient. The MUA on the receiving end can be configured to recognize emails with the tagged subject and to divert them into a folder. Some systems even permit their users to have personalized configurations. An example configuration may be to tag spam mails for a score of 2, to transfer spam mails to a special folder on the mail server for a score of 10, and to delete spam mails immediately for a score of 15.

SpamAssassin is quite robust and has been used in Unix world for many years. Thus, the system is chosen to be the foundation on which we build our anti-spam system.



Chapter 4 Our CASEF System

CASEF—Collaborative Anti-Spam Email Filtering— is a system that is designed to filter spam emails at the router level. The system contains several components such as email honeypot module, policy filtering module, digest matching module and SpamAssassin filtering module. We use several methods to defeat spamming collaboratively. In section 4.1, we give an overview of the CASEF system. In section 4.2, we describe the system architecture completely and illustrate all modules that are used to filter spam mails.

4.1 System Overview

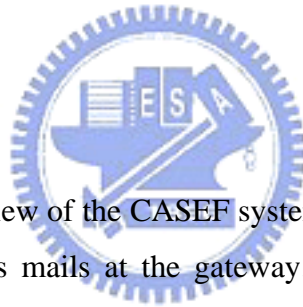


Figure 17 shows an overview of the CASEF system. Installed on Linux systems, CASEF system is designed to process mails at the gateway. An email honeypot is a mail server designed to harvest spam mails. When the honeypot receives a new spam mail, it will send the mail to the CASEF system. CASEF system then will calculate the mail's digest and store it into the system's internal caches.

The mails come from not only the honeypot but also the Internet. In general, everyday mails coming from the Internet are the main targets that need to be checked. When the mails pass through the firewall, CASEF will dichotomize each mail into spams or non-spams. If the system regards some mails to be spams, the mail will neither be deleted nor quarantined. Rather, a subject tag is placed on the mail to notify email users or MUAs of the situation. Whether or not the incoming mail is a spam, the mail will be transferred to the internal mail server. Eventually, the mail will arrive at the recipient by his/her use of an MUA. The email sending process is not altered; emails are directly sent out by an outgoing SMTP server.

Some email filtering systems integrate themselves into the mail server; however, the original mail server will need to be modified. Furthermore, the original mail server will have

an increased overhead. We take a totally different approach by implementing our CASEF system at the gateway. Due to this, the original mail server needs not be changed at all.

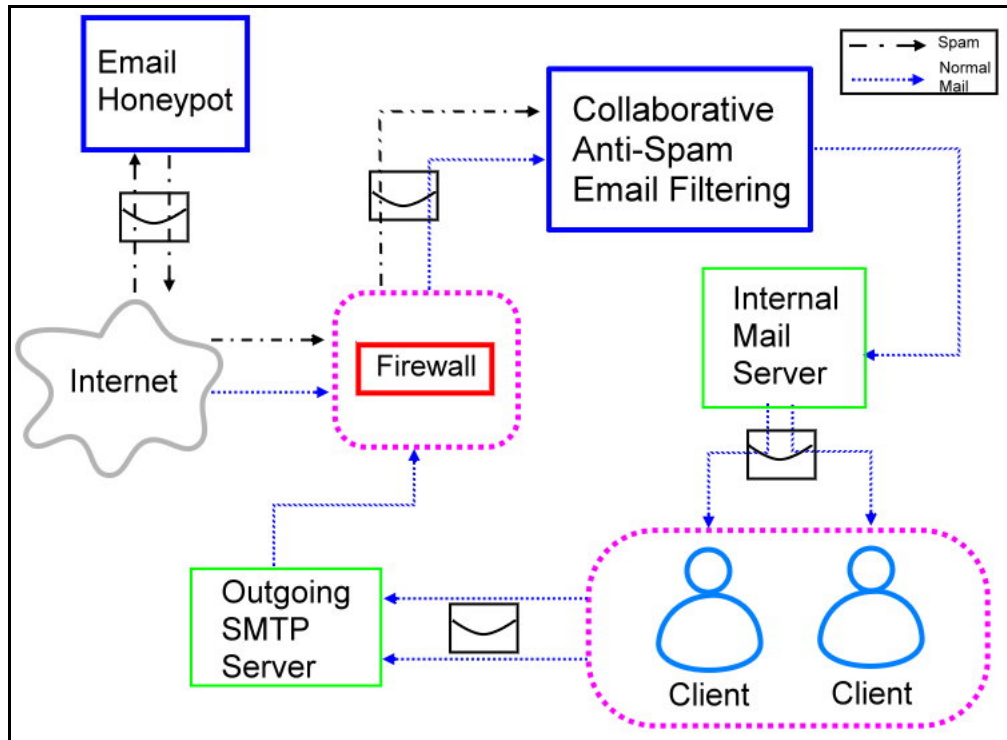


Figure 17 CASEF System overview



4.2 System Architecture

CASEF system is designed to act as an email filter on Linux systems. CASEF system receives emails from the email honeypot and the Internet, matches the mails against filtering rules, and then transfers the mails to the internal mail server. In our CASEF system, there are four kinds of filtering modules: email honeypot module, policy filtering module, digest matching module and SpamAssassin filtering module.

The email honeypot module has the objective of luring spammers into sending it spam mails. The sample of spam mails obtained can help the digest matching module to differentiate between normal emails and spam emails. Policy filtering module incorporates a black/white list to pass or delete incoming emails immediately, and also includes a mechanism to validate regular mails. Digest matching module have four caches that maintain digests of mails coming from email honeypot and SpamAssassin. If the digest of the newly received mail matches any digest stored in the four caches, a mail handling event will be triggered. SpamAssassin filtering module uses the popular SpamAssassin anti-spam engine to process each mail and to compute its score. If the score exceeds the limit set by the administrator, the mail will be tagged on the subject line. Detailed process of each module will be illustrated in the following sections.

4.2.1 Email Honeypot Module

Email honeypot is a part of the CASEF system. The purpose is to collect spam mails. Spam mails collected this way can assist the CASEF system in noticing any repetitions in the spam mails' content.

We use the program, sendmail, to serve as an email server to receive emails from spammers. On the Internet are many kinds of services and contents such as finance, entertainment, education, auctions, jobs and health care. Thanks to the research done by Dr. Guido Schryen, we now know what kinds of services are popular among the spammers. We thus create several dummy accounts to match all types of popular services. The next task is then to strategically disperse those accounts to web pages, forums and other Internet services according to their classification. For example, the home page of a bank will be classified as finance. These dummy accounts are merely created to trap spammers and are never used to communicate with other people. Therefore, non-spam mails sent by normal users should never find their way into one of these accounts.

In general, the mail server receives mails from the Internet and distributes them into mailboxes labeled by the destination addresses. Similarly but not exactly, our email honeypot only uses two folders: the “spam” folder and the “unknown” folder. Figure 18 illustrates the concept of the email honeypot. Initially, a portion of the dummy accounts are assigned to web pages that feature adult-oriented, financial and gambling information. The mails generated from these types of services are classified as spam mails without any judgment by people or by machines. Therefore, these mails are put into the spam folder directly. Other mails that need to be classified by people or machines are placed into unknown folder. In the end, the spam folder should contain only spam mails while the unknown folder could contain both spam and non-spam mails.

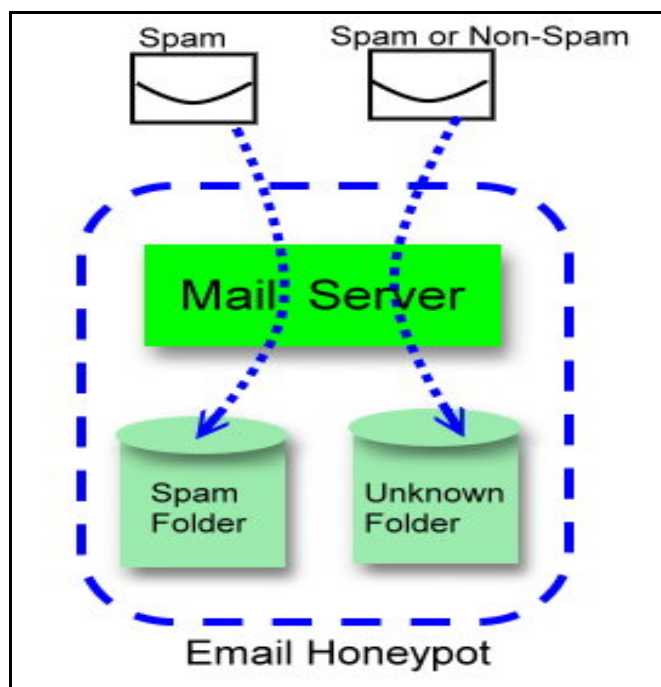


Figure 18 The process of email honeypot

In the email honeypot, there are two special email accounts named “honeypot.spam” and “honeypot.unknown”. These two accounts are used to send the mails in the spam folder and the unknown folder to CASEF for further processing. If the mails are in the spam folder, the system will use “honeypot.spam” as the email account. Otherwise, “honeypot.unknown” will be used. When CASEF receives its emails, the policy filtering module will check the source mail address. If the mail address is “honeypot.spam” or “honeypot.unknown”, the mail will be moved to the first or second cache according to the mail address. The mail is moved to the first cache if the mail address is “honeypot.spam”. And the mail is moved to the second cache if the mail address is “honeypot.unknown”.

Figure 19 shows the path the mails have to travel from the two folders of the email honeypot to the two caches of CASEF.

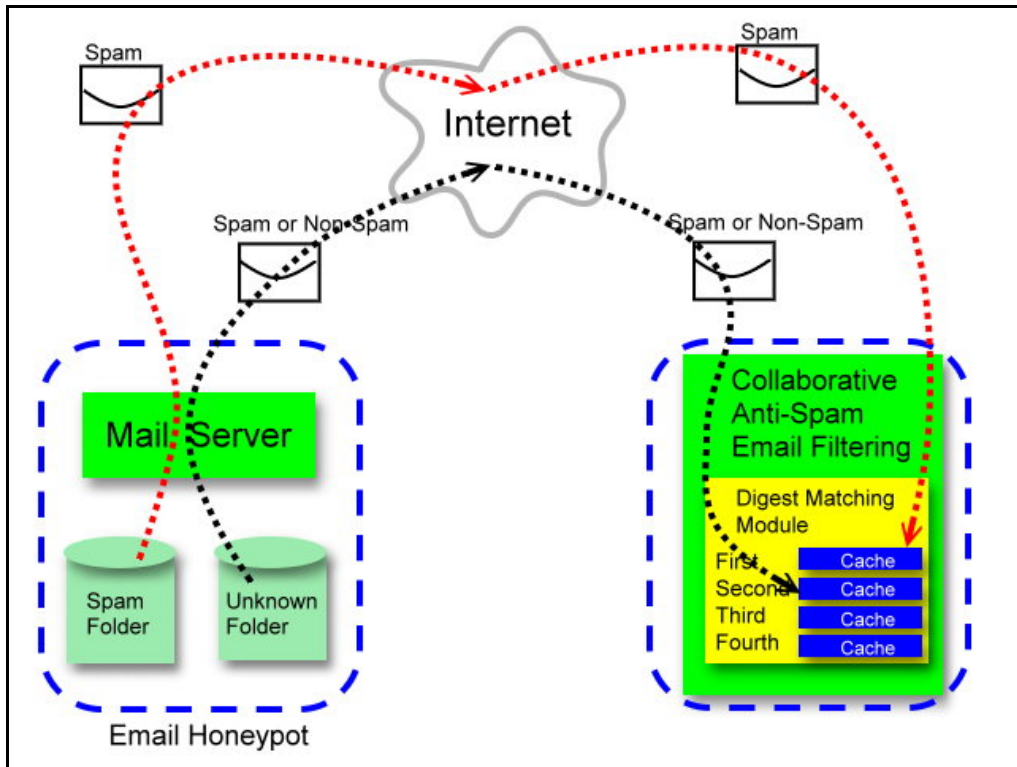


Figure 19 The process of the mails from two folders to two caches

4.2.2 Collaborative Anti-Spam Email Filtering

Collaborative anti-spam email filtering retrieves emails from the Internet and the honeypot and detects spam mail within the mails retrieved. The mails from the honeypot are directly moved into the first cache and second cache according to the mail addresses. Other Internet mails require further testing. After a series of tests, the mails are sent to the internal mail server. If the mail is a spam mail, the system will set a spam tag on the subject to remind the user or MTA.

There are three important modules in the CASEF system. They are the policy filtering module, the digest matching module and the SpamAssassin filtering module. Figure 20 shows the architecture of the CASEF system. The three modules form the core of CASEF system to deal with the spam mails. When a mail first arrives at the CASEF system, it will be processed by the policy filtering module, the digest matching module and the SpamAssassin filtering module in turn. If the mail is on the white list, it just passes through to the internal mail server

immediately. Other mails must be processed by the three modules. If a module determines that a mail is spam that mail will be tagged on its subject line and will not be processed by the remaining modules.

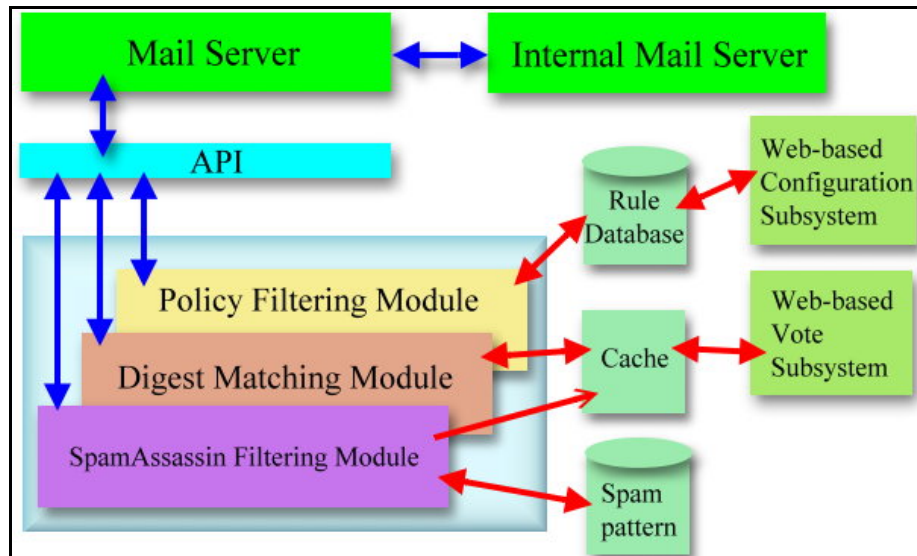


Figure 20 The architecture of the collaborative anti-spam email filtering

After a mail from the Internet is processed by the CASEF system, the mail may be delayed by 0 min, 10 min, 30 min or 1 hour to be delivered to the internal mail server. The delayed time can be set by the administrator according the requirement of his/her system. If the user wants to receive the mails immediately, he/she can ask the administrator to add his/her mail address to the white list. The detailed process of the three modules is explained in the rest of this chapter. Figure 21 shows the detailed mail process flow.

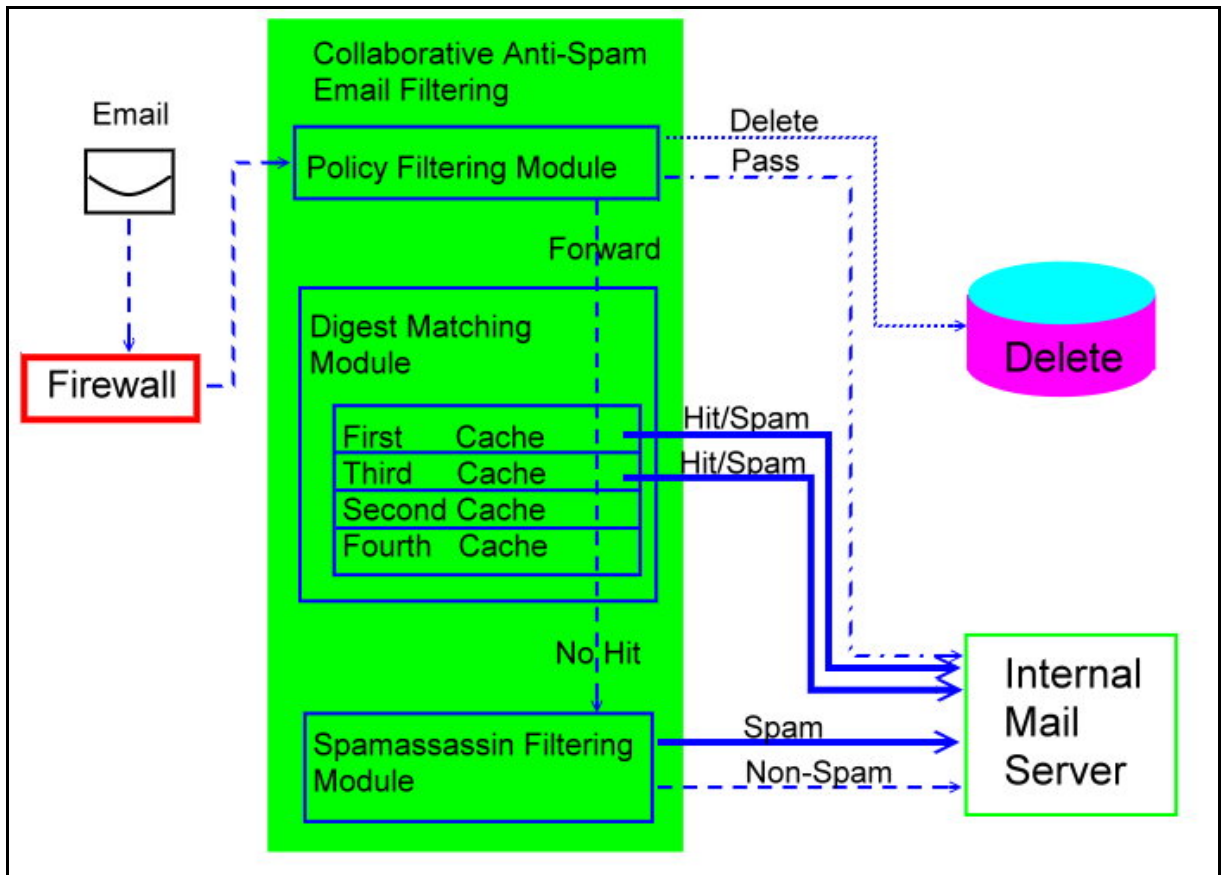


Figure 21 The detail mail process flow

4.2.3 Policy Filtering Module

Policy filtering module is the first line of defense in our CASEF system. All mails need to be checked in this module no matter what the mail is categorized into—spam or non-spam. The policy filtering module works with the information provided in the SMTP dialog. The dialog includes the interplay of the SMTP protocol and also the messages of the mail transaction itself. An example dialog may contain such items as helo test.com.tw, mail from: hello@test.com.tw, and rcpt to: example@try.com.tw. Interested readers may want to look at Chapter 2 for a detailed explanation on those items. Policy filtering module performs the tasks of matching mail addresses against the white/black list, validating the IP address, validating the HELO/EHLO parameters, and validating the envelope sender.

White/Black list

First of all, the policy filtering module will look at the IP address of the sender and the

mail addresses of the sender and the receiver. Because our CASEF system also receives its mails from an email honeypot module, it is important to determine the source of the incoming emails. If the email is from the email honeypot, our CASEF system does not waste any more time to carry out other check items. The mails matching the white or black list will be removed from further processing by passing it through or by erasing it, respectively. This step also can save a lot of time since less work is directed toward the other modules. If some users do not want their mails delayed, they can add their mail addresses to the white list. The administrator can be given the responsibility to perform the procedure of adding mail addresses to the lists.

Validating the IP address

Policy filtering module will examine the reverse DNS record of the incoming connection and matches that with the DNS real-time black hole list (RBL). If the DNS record does not exist or is listed on the RBL, CASEF system will terminate the connection and report an error message to the sending machine.

Validating the HELO/EHLO parameter

Policy filtering module verifies the validity of HELO/EHLO parameter provided that the sending mail server domain has a DNS record. If the parameter is invalid, CASEF system will reply with an error message to the sending machine and close the connection.

Validating the envelope sender

Firstly, policy filtering module checks the MX records or A records in DNS for the email address of the envelope. And it connects to the original mail server to see if the mailbox is valid or invalid. Secondly, it checks whether the envelope sender has a fake local name or not. If an error occurs, CASEF system will also stop the transaction and send an error message back to the sending machine.

If there does not exist an error on IP address, mail addresses, SMTP protocol parameters, or the envelope of the mail, CASEF system will forward the mail to the digest matching module for more spam checking.

4.2.4 Digest Matching Module

Digest matching module has four caches that hold digests of mails released from the email honeypot and SpamAssassin filtering module. The first cache stores the digests calculated from the mails of the honeypot's spam folder in LRU order. The second cache then handles the digests calculated from the unknown folder of the email honeypot but replaces the digests in FIFO order. The third cache has the LRU policy and stores the digests calculated from spam mails judged by SpamAssassin filtering module. The fourth cache replaces its content in FIFO order and stores the digests calculated from normal mail that is judged by SpamAssassin filtering module.

When policy filtering module forwards the mails to the digest matching module, the digests of the mails are computed and are matched with against the digests stored in the four caches. In our CASEF system, Nilsimsa digest technique is chosen to be the digest algorithm. Nilsimsa is an open source digest-based technique for spam detection [6]. A Nilsimsa digest is 32-byte long. The digest computed using MD5 or SHA1 have the property that a small change in the content will produce a large difference in the digest. Contrarily, the digests computed by Nilsimsa will only differ by about 0%~10% when the two respective contents have a 0%~10% differences in their file size. Due to this nice property, Nilsimsa can be used for spam detection even if spam messages have little differences.

If the digest computed by the digest match module hit one of the four caches, the CASEF system will compare the digest computed with the digests stored in the **first**, the **third**, the **second**, and the **fourth** cache in this order. The digest matching scenario is given in Table 8.

Event	Action
Hit the First cache	The module classifies the mail as spam and gives a tag on the subject.
	The module sends the mail to the internal mail server.
	The module updates the digest that is hit in the First cache in LRU order.
Hit the Second cache	The module sends it to the spamassassin filtering module.
	The module moves the digest that is hit in the Second cache to the First cache.

Hit the Third cache	The module classifies the mail as spam and gives a tag on the subject.
	The module sends the mail to the internal mail server.
	The module updates the digest that is hit in the Third cache in LRU order.
Hit the Fourth cache	The module sends it to the spamassassin filtering module.
	The module moves the digest that is hit in the Fourth cache to the Third cache.
No Hit	The module forwards it to the spamassassin filtering module.

Table 8 The scenario of digest matching module

Figure 22 shows the move of the four caches.

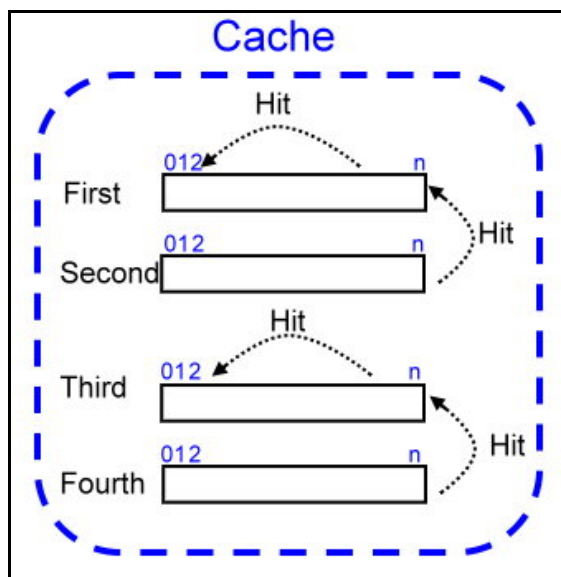


Figure 22 The move of the four caches

The mails coming from the unknown folder of the email honeypot not only produce digests to be stored in the second cache but the messages themselves are also stored in a temporary mail folder. Because the mails originate from dummy accounts and every mail should be checked by people or by machine, we can hire some mail janitors to help classify the mails manually. If the janitors recognize some mails in the temporary folder spams, they can transfer their digests from the second cache to the first cache. In the same way, they can remove the digest from the second cache if the mail is a non-spam mail.

4.2.5 Spamassassin Filtering Module

SpamAssassin filtering module uses SpamAssassin anti-spam engine to detect spam mails. After the mails progress through the digest matching module, they will take a series of tests conducted by SpamAssassin filtering module.

The anti-spam engine scans the entire mail and calculates a score according to the rules. If the score exceeds the limit set forth by the administrator, the mail will be given a tag on its subject field. The detailed procedure of SpamAssassin is illustrated in Section 3.3 and we refer the interested reader there. In addition to giving a tag on the subject field, some more works are still waiting to be performed. A scenario of SpamAssassin filtration process is given in Table 9.

Event	Action
Spam	The module classifies the mail as spam and gives a tag on the subject.
	The module sends the mail to the internal mail server.
	The module copies the digest of the mail to the Third cache.
Non-Spam	The module passes the mail and sends it to the internal mail server.
	The module copies the digest of the mail to the Fourth cache.

Table 9 The scenario of spamassassin filtering module

Chapter 5 Experimental Results

In this chapter, we will show the experimental results on CASEF system and SpamAssassin. And we use the simulation results to prove that CASEF system is better than SpamAssassin. CASEF system's components will need to be experimented on before we could simulate the CASEF system.

5.1 Experimental Environment

We simulate the CASEF system on one of the computers in our laboratory. The computer is running on the Linux Red Hat OS with sendmail installed to be a mail server that can receive the mails from the Internet. We also installed SpamAssassin to be the anti-spam engine for the mail server. The hardware and software configurations are detailed in Table 10.

The mails that we use to train SpamAssassin and also to test CASEF system and SpamAssassin are obtained from Spam Mails Archive [8].

CPU	Intel Pentium 1.7GHz
RAM	256M bytes
OS	Red Hat Linux 9.0 (Kernel 2.4.20-8)
Sendmail	sendmail-8.12.8-9.90
Spamassassin	spamassassin-3.1.3-1

Table 10 Experimental environment

5.2 Performance and Overhead

In this section, we will test the CASEF system components separately. The first experiment is designed to test the Nilsimsa digest. We wish to find out how many bits of Nilsimsa digest will be different provided a 10 bytes difference between two otherwise identical mails. Determining the optimal size for the first, second, third and fourth caches is the purpose of the second experiment. Many mails are used to determine a number of cache sizes that will let the system to have the maximum hit rate. After testing the components of the CASEF system, we will test the accuracy of SpamAssassin. Finally, we test the accuracy and performance of the CASEF system.

5.2.1 Digest Matching Threshold

We use Nilsimsa digest technique to summarize and store the content of the mails as the digests in the four caches. Nilsimsa digest algorithm is an open source and can be downloaded from the Internet. We run the Nilsimsa digest program, which is written in C language, on the Linux machine. The testing mails range in size from 256 bytes to 32 KB. We select two mails that have a 10 bytes difference in their content and calculate their digests. The two digests are compared to see how many bits are different. The experimental result is drawn in Figure 23. The maximum number of different bits between the digests is 14.

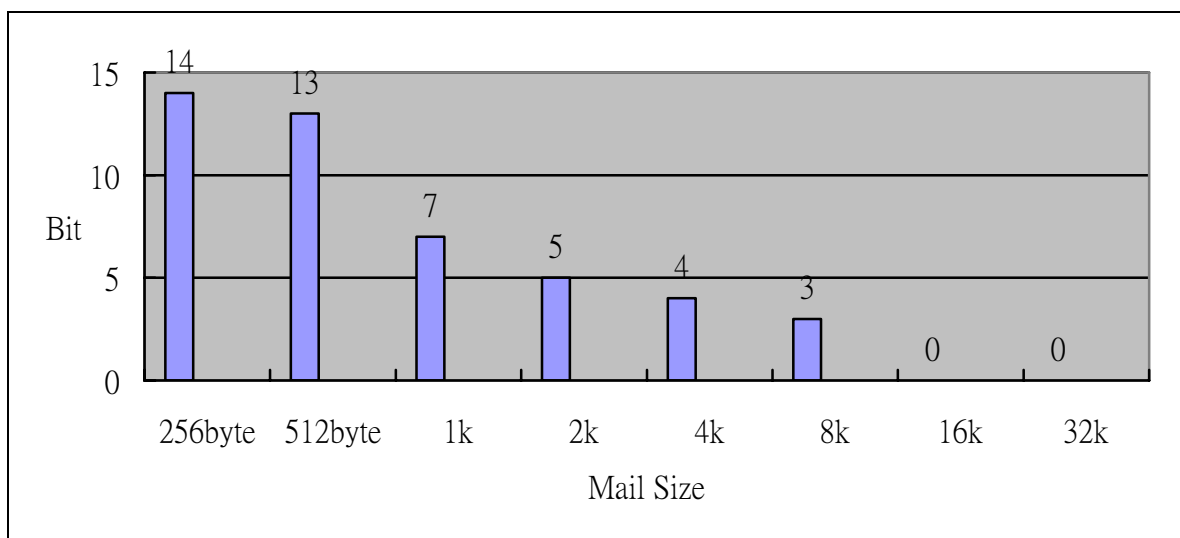


Figure 23 The first experimental result of digest matching

In the second experiment, the two mails contain the same information but the contents are not in the same order. We want to compare the digests and know how many bits are different between the two digests. The second experimental result is shown below (see Figure 24). The maximum number of different bits is 12.

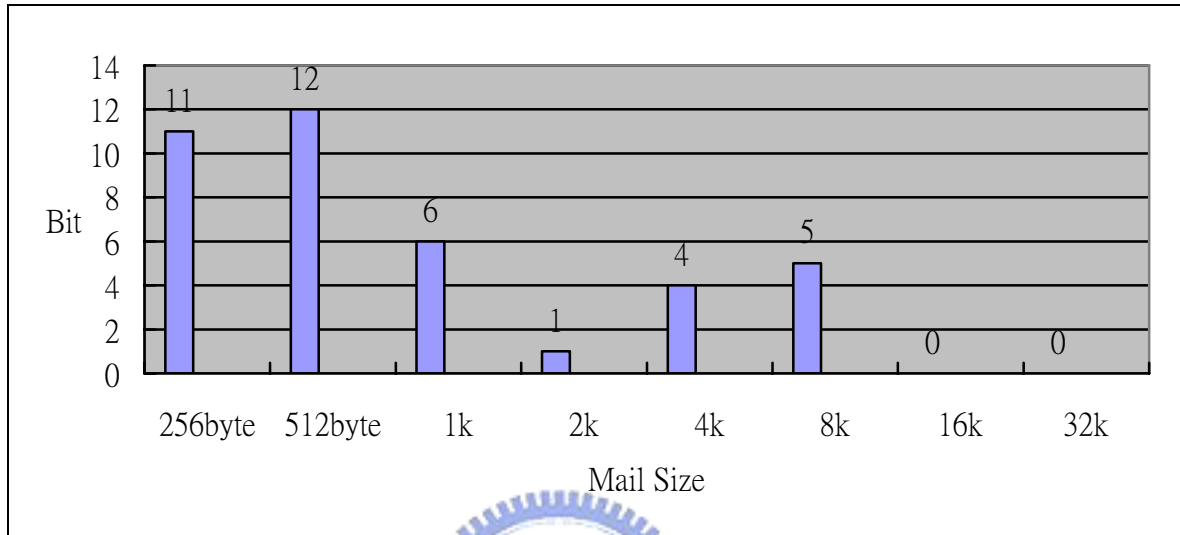


Figure 24 The second experimental result of digest matching

Next, we would like to know how many bytes difference between the two mails such that the digests will have 13-bit difference in the third experiment. The third experimental result is shown below (see Figure 25). For mails that are over 1 KB in size, we found that the difference between the contents needs not to be over 4.68%.

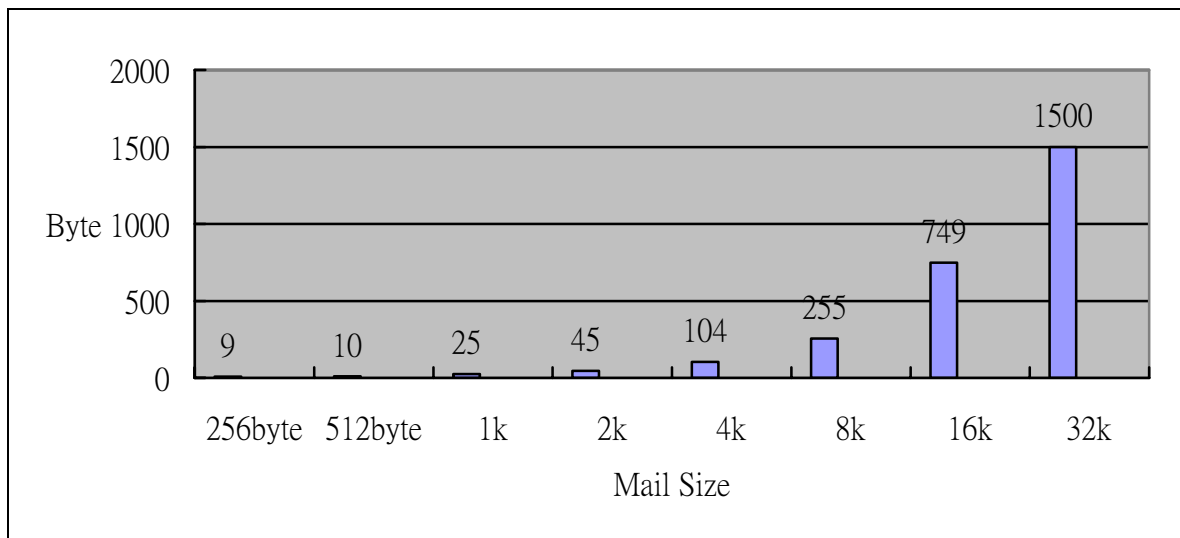


Figure 25 The third experimental result of digest matching

The fourth experiment is similar to the first experiment in that we replace 10 bytes in the

content for mails having sizes from 256 bytes to 32 KB. We just changed the 10 bytes difference to a different place of the content in the fourth experiment. The fourth experimental result is shown below (see Figure 26). The maximum number of different bits is 16.

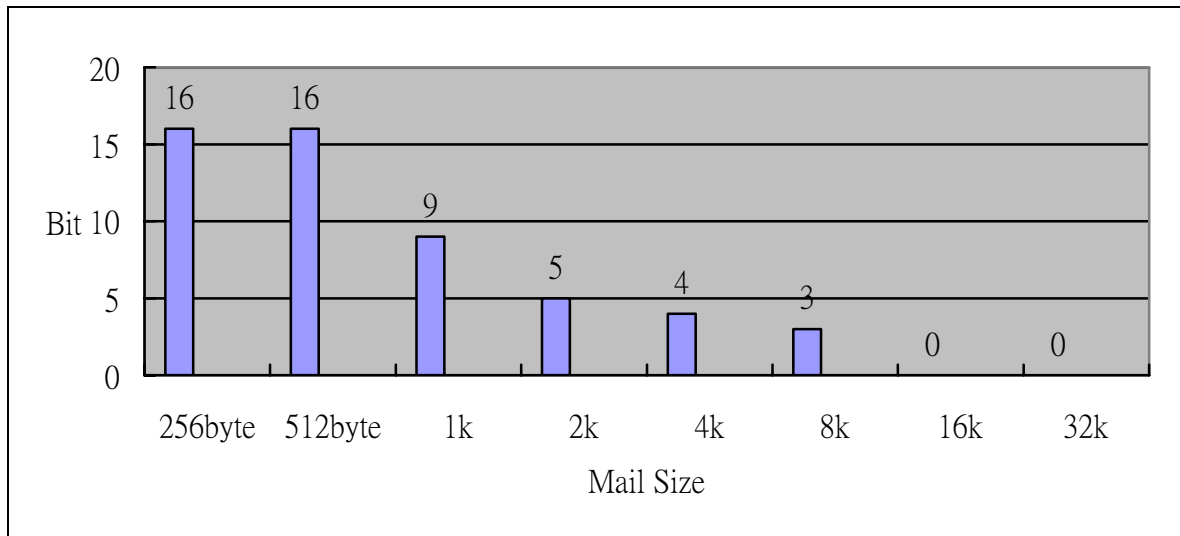


Figure 26 The fourth experimental result of digest matching

After testing the Nilsimsa digest, we decide that 16 bits should be the digest matching threshold in the CASEF system. If an incoming email has a digest that has a 17-bit or more difference when compared with the digests stored in the four caches, we will consider that the mail is not similar enough to hit the caches.

5.2.2 Four Caches Sizes

As mentioned in previous chapters, our digest matching module has four caches that are named the first, the second, the third and the fourth cache. The digests in the first and the third cache are replaced in LRU order. In the second and fourth cache, the digests are replaced in FIFO order. In this part of the experiments, we use the same set of 1000 mails to test the four caches. The objective is to try to determine the cache sizes that will cause the system to have the maximum hit rate.

To test the first cache, we use 1000 mails that include 700 spam mails and 300 duplicate mails taken from the 700 spam mails. The 1000 mails are inputted to the cache in random order. When the mails hit the first cache, the cache will do LRU process. Figure 27 shows the experimental result. When the first cache size becomes 600, the cache reaches the maximum hit rate. The hit rate does not increase even if we increase the cache size.

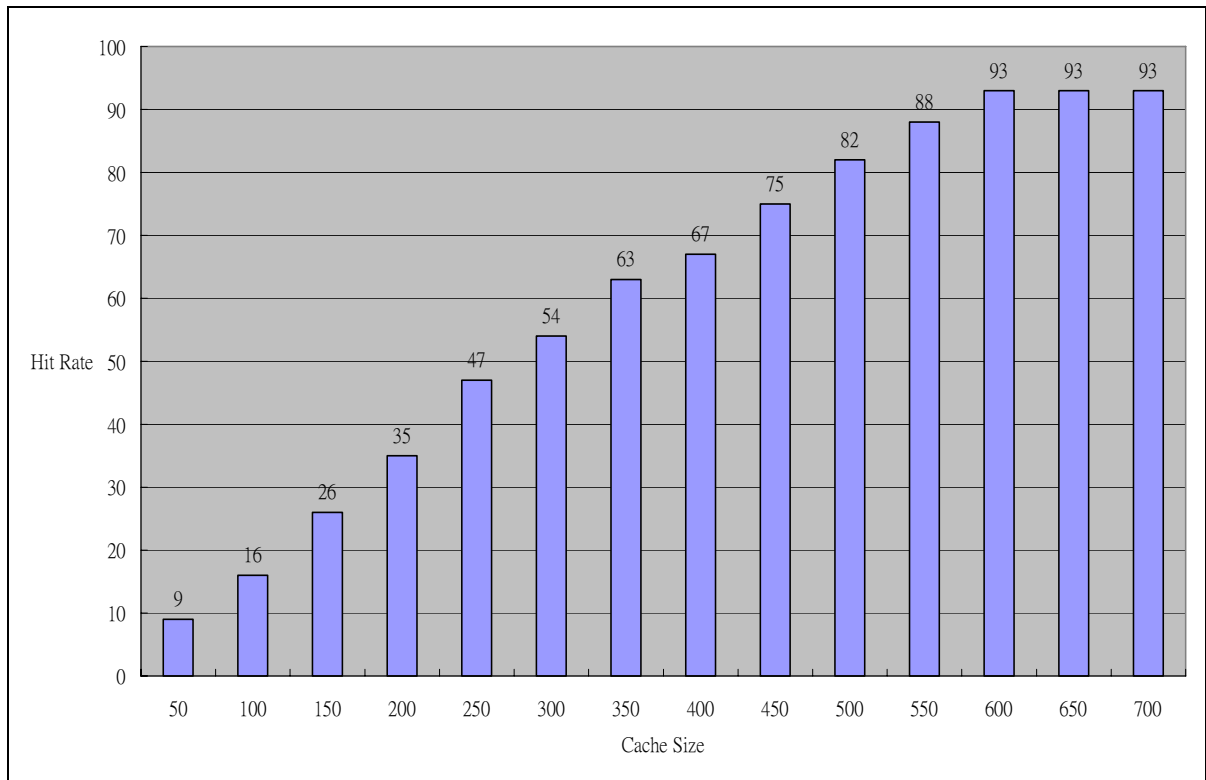


Figure 27 The experimental result of first cache

We also use the same set of mails in the first experiment as the test data for the second cache. The 1000 mails are inputted to the second cache in random order. When the mails hit the second cache, the cache will do FIFO process. The experimental result is shown in Figure 28. The cache has the maximum hit rate when the cache size becomes 400.

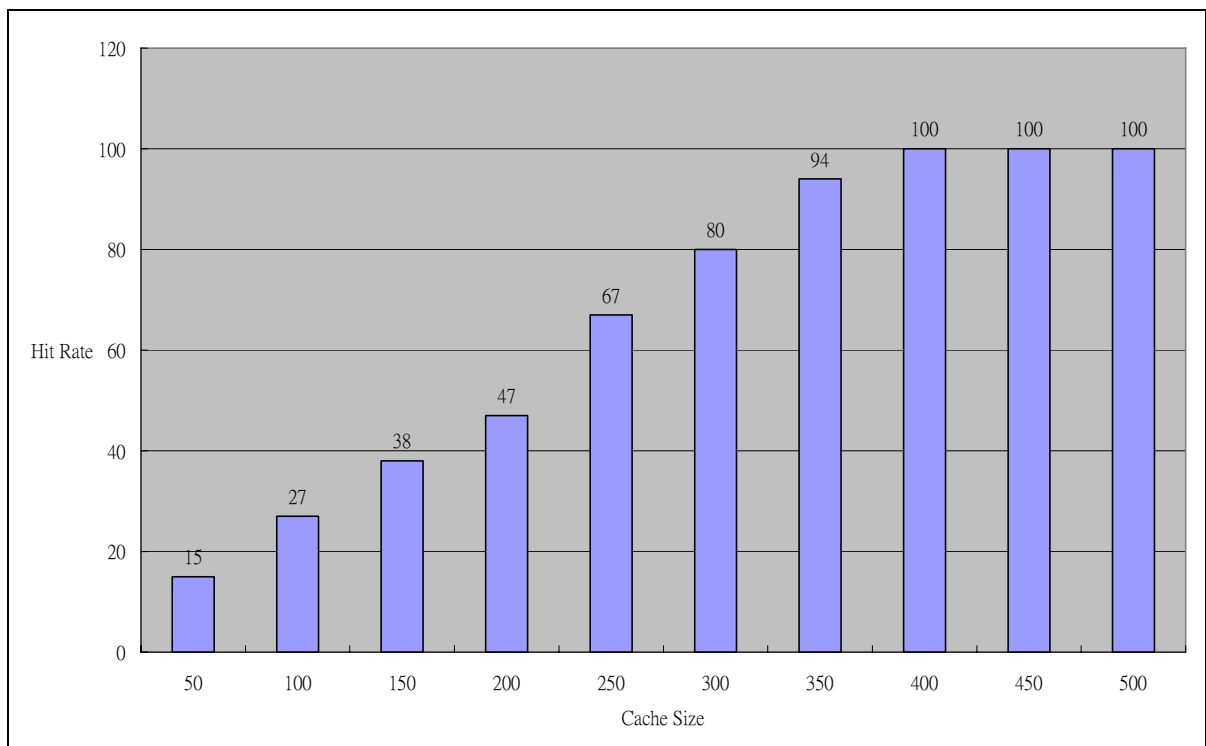


Figure 28 The experimental result of second cache

In the experiment of the third cache, we use 1000 mails that include 500 spam mails and 500 mails duplicated from the original 500 spam mails. The 1000 mails are inputted to the cache in random order. When the mails hit the third cache, the cache will do LRU process. This experimental result is shown below (see Figure 29). We keep increasing the cache size until the hit rate ceases to increase. This occurs when the cache size becomes 500.

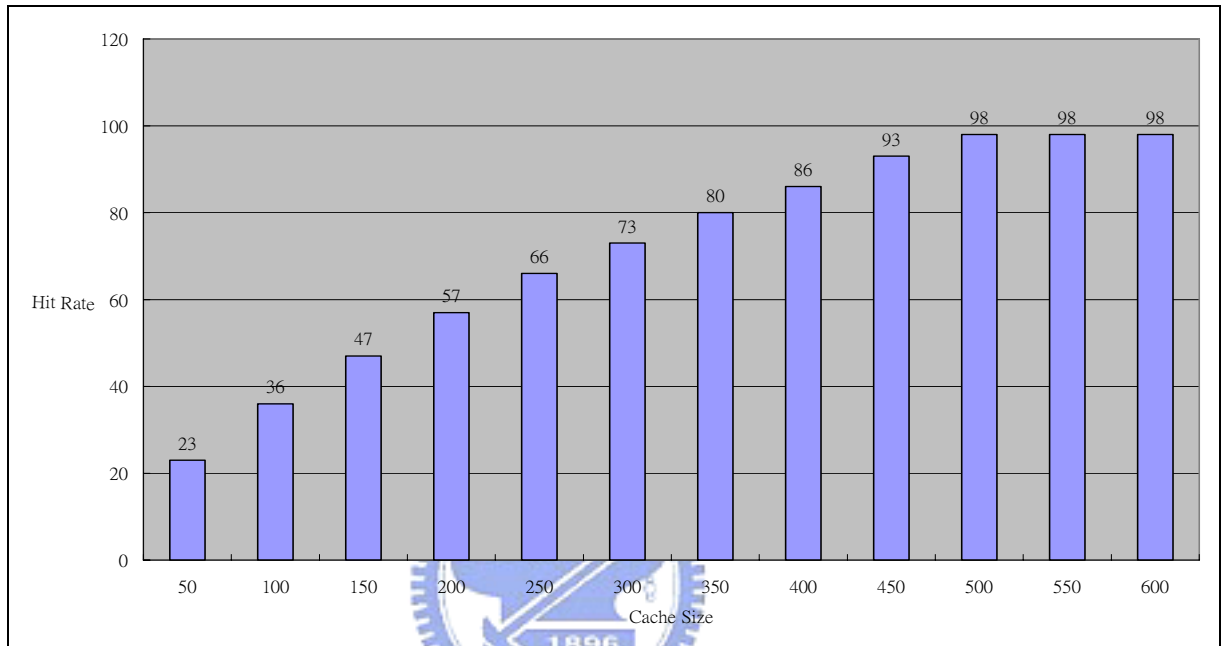


Figure 29 The experimental result of third cache

To test the fourth cache, the mails from the experiment of the first cache are utilized again. The mails are inputted to the cache in random order. The cache will do FIFO process when the mails hit the fourth cache. From the result of the experiment, we know that the hit rate is maximized when the cache size is 450. Figure 30 shows the experimental result below.

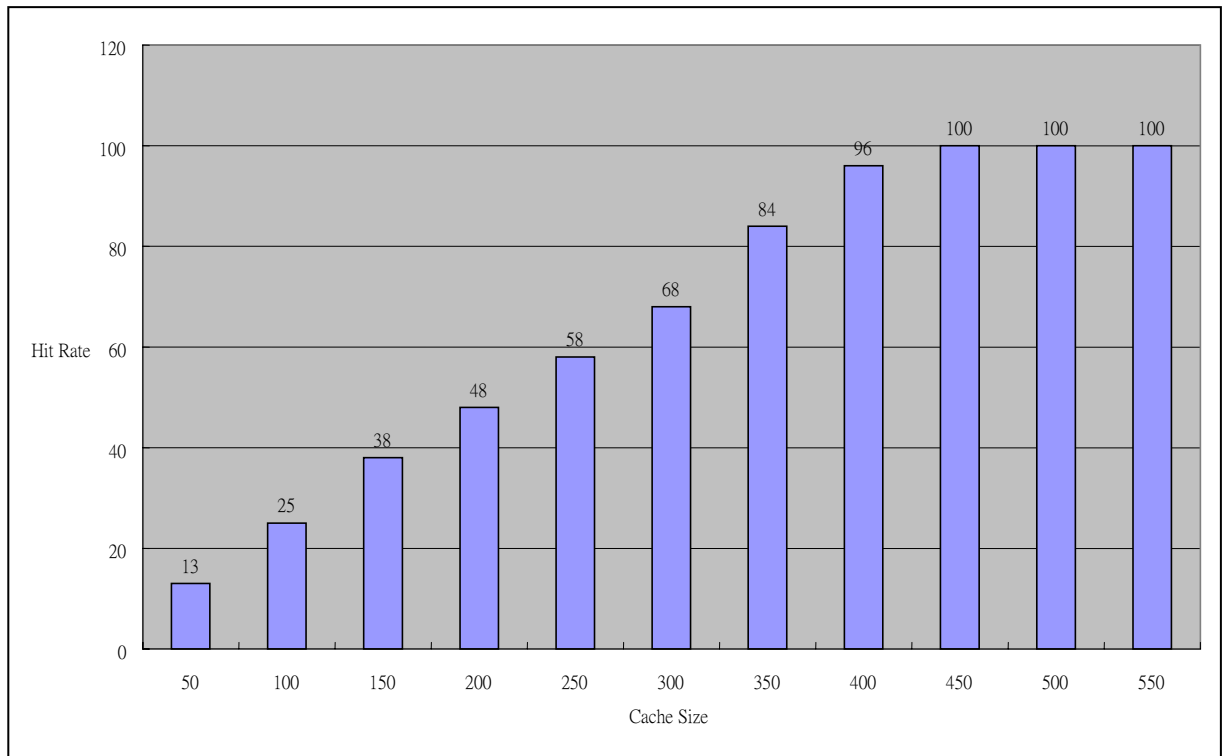
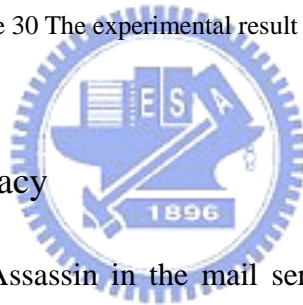


Figure 30 The experimental result of fourth cache



5.2.3 Spamassassin Accuracy

We have installed SpamAssassin in the mail server and are ready to perform, in this section, two experiments with different combinations of mails. In the first experiment, we use 10,000 mails with the makeup of 4,000 ham mails, 3,000 spam mails and 3,000 mails duplicated from the original 3,000 spam mails to test the filtering accuracy of SpamAssassin. We simulate this scenario because real networks typically have 60% spam mails. This experiment also simulates the situation where spam mails have one duplication. In addition, 2,551 normal mails and 2,398 spam mails are employed to train SpamAssassin and test the system performance on the runtime processor and memory requirements. The performance and accuracy of SpamAssassin will be the baseline to measure the respective statistics of our CASEF system. In Table 11, the performance and the accuracy of the SpamAssassin are shown. After the experiment, 5,312 mails are classified as spam mails by SpamAssassin.

Training	2551 ham mails and 2398 spam mails
Testing	10000 mails (3000 spam mails x 2 + 4000 ham mails)

Runtime	3906 seconds
Memory	10.9%
Accuracy	88.53%

Table 11 The first experiment of Spamassassin

In the second experiment, we use 1,0000 mails that include 4,000 ham mails, 1,200 spam mails and 4,800 duplicate mails taken from the original 1,200 spam mails to test the accuracy of SpamAssassin. This experiment also simulates the situation where the spam mails have four duplications. Otherwise, the testing environment is exactly the same as the first experiment. The performance and the accuracy of the SpamAssassin are shown in Table 12. The experiment shows that 5,320 mails have been classified as spam mails by SpamAssassin.

Training	2551 ham mails and 2398 spam mails
Testing	10000 mails (1200 spam mails x 5 + 4000 ham mails)
Runtime	3967 seconds
Memory	10.9%
Accuracy	88.66%

Table 12 The second experiment of Spamassassin

5.2.4 CASEF System Accuracy

In this section, we also conduct two experiments on our CASEF system. The mails used for testing CASEF system are identical to the two experiments performed on SpamAssassin, respectively. In the first experiment, the mails are imported to the CASEF system in random order. We assume that 1,500 spam mails are from spam folder, 1,500 spam mails are from unknown folder and 400 normal mails are from unknown folder. If the percentage of Internet mails captured by Honeypot at the same time is 100%, the digests of 3,400 mails from spam and unknown folder are all loaded into the first cache and the second cache, respectively. For this experiment, we simulate the situation where only two duplicate spam mails appear, and

we perform the test with 0% to 100% of mails captured by Honeypot from the Internet. The performance is shown in Table 13 and the accuracy of CASEF system is also shown below (see figure 31).

Testing	10000 mails (3000 spam mails x 2 + 4000 ham mails)
Runtime	3906 + 121 seconds (overhead 3.09%)
Cache Size	2000
Memory	10.9% + 0.3% (overhead 2.67%)

Table 13 The first experiment of the CASEF system

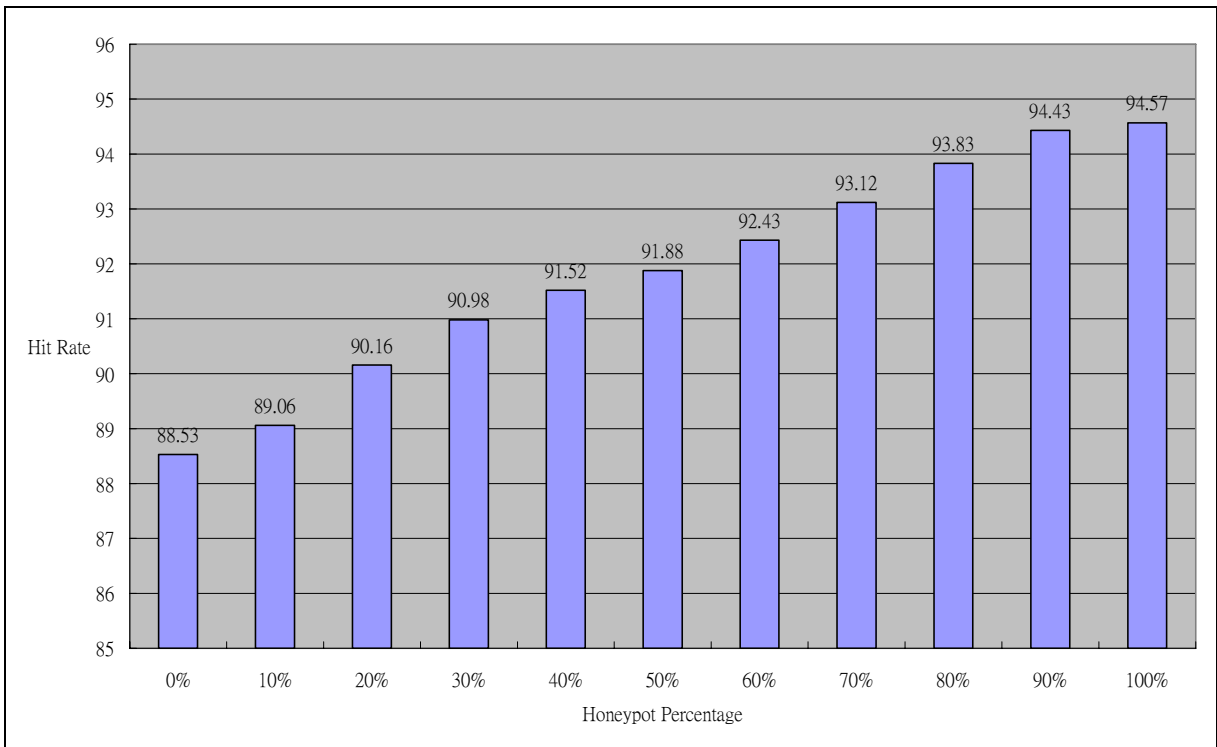


Figure 31 The accuracy of the first experiment

We do not change the testing environment in the second experiment. We assume 600 spam mails from spam folder, 600 spam mails from unknown folder and 400 normal mails from unknown folder. If the percentage of Internet mails captured by Honeypot at the same time is 100%, the digests of 1,600 mails from spam folder and unknown folder all make their way into the first cache and the second cache, respectively. We also repeat the test by setting Honeypot to capture between 0% to 100% Internet mails. The performance is shown in Table 14 and the accuracy of CASEF system is also shown below (see Figure 32).

Testing	10000 mails (1200 spam mails x 5 + 4000 ham mails)
Runtime	3967 + 116 seconds (overhead 2.92%)
Cache Size	2000
Memory	10.9% + 0.3% (overhead 2.67%)

Table 14 The second experiment of the CASEF system

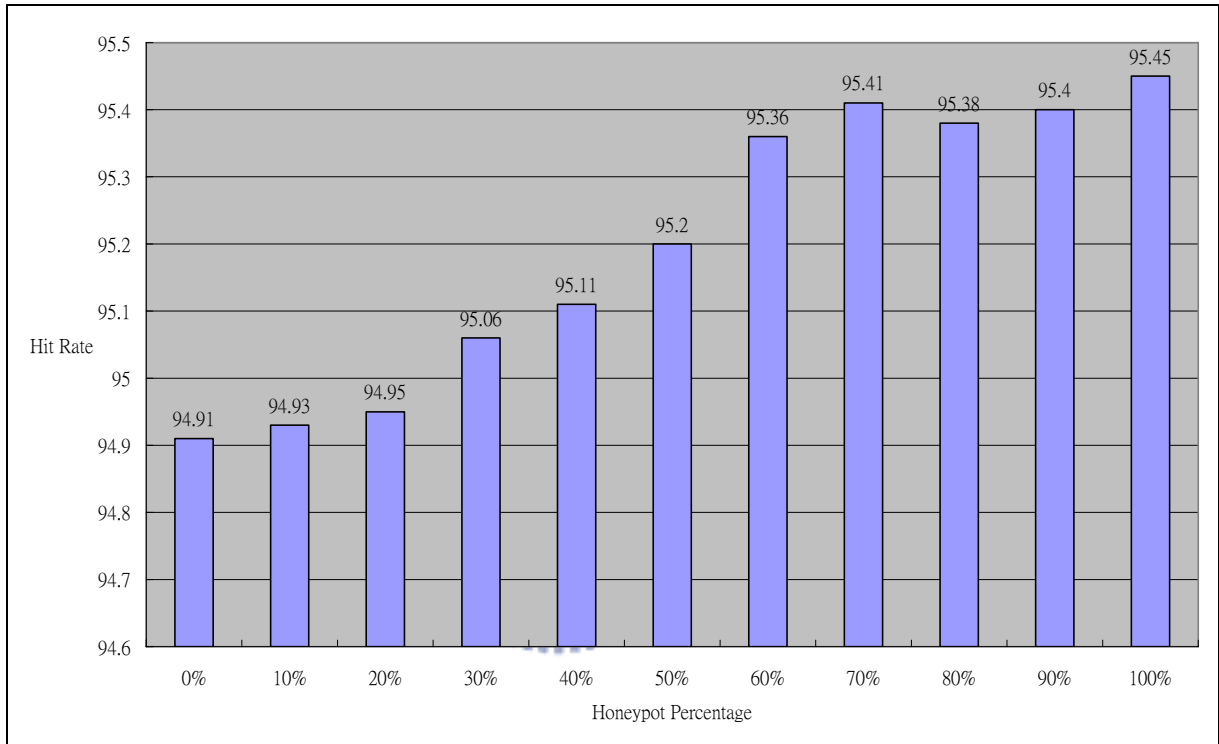
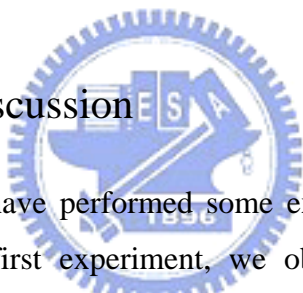


Figure 32 The accuracy of the second experiment

Chapter 6 Conclusion

In previous chapters, we have conducted some experiments to test the performances of SpamAssassin and CASEF system. We shall compare the CASEF system with SpamAssassin and summarize the advantages and disadvantages of the CASEF system in this chapter. Finally, we will describe the future works briefly.

6.1 Conclusion and Discussion



In previous chapter, we have performed some experiments to compare SpamAssassin and CASEF system. In the first experiment, we observed that when we increased the percentage of capturing mails from honeypot, the accuracy of the CASEF system also increases rapidly. However, the accuracy of the CASEF system is the same as SpamAssassin when the percentage is 0%. In the second experiment, the accuracy of the CASEF system has a large improvement over SpamAssassin even if the percentage is 0%. However, the accuracy of the CASEF system only increases by 0.5% even after we increased the percentage from 0% to 100%. In Table 15 given below, we compare the difference between SpamAssassin and our CASEF system.

From the experiments, we are able to compare our CASEF system with Spamassassin. Our CASEF system was shown to safeguard email users from spam mails. Our CASEF system has a low runtime overhead and a small memory footprint yet achieves high accuracy. Blocking spam mails at the router level does not require modifications to the original mail server. Consequently, any mail servers can use CASEF system to protect the mail users easily.

	CASEF System	Spamassassin
Runtime Overhead	Low	High
Memory	Low	High
Accuracy	High	Normal
Defend the repeated spam mails	Yes	No
Mail Delay	Yes	No
Extension	Yes	No

Table 15 Compare the CASEF system to Spamassassin only

6.2 Future Work

Our CASEF system utilizes SpamAssassin to be the base of our spam filtering system and promotes the accuracy of anti-spam. However, there are still some works that can be done to improve our CASEF system. We list some parts that can be improved below.

- (1) Adding an anti-virus engine allows our system to detect the virus infected emails. In this way, our CASEF system will not only defend against spam mails but also can detect virus mails.
- (2) Sharing the digests of the first cache with other systems. From our first experiment, it is quite obvious that the honeypot can improve the accuracy considerably.
- (3) Improving the scalability of our CASEF system. Maintaining the level of QoS may become difficult when we try to use the CASEF system in a big enterprise.
- (4) Supporting multilingual email filtering. Currently, our CASEF system can only classify English mails. In the future, we could add the capability to detect Chinese mails by writing a Chinese_rules.cf for SpamAssassin.

Reference

- [1] Wei-Li Huang, J.Y. Huang, and Wen-Nung Tsai, "Design and Implementation of E-mail Filtering System," National Computer Symposium (NCS 2005), Tainan, Taiwan, Dec. 15-16, 2005.
- [2] Geoff Mulligan, "Removing the Spam: Email Processing and Filtering," Addison-Wesley, March 16, 1999, ISBN: 0201379570.
- [3] Banit Agrawal, Nitin Kumar, and Mart Molle, "Controlling Spam E-mail at the Routers," In IEEE International Conference on Communications (ICC 05), Seoul Korea, 2005
- [4] Guido Schryen, "An e-mail honeypot addressing spammers' behavior in collecting and applying address," Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2005. Proceedings from the Sixth Annual IEEE 15-17 June 2005 Page(s):37 – 41.
- [5] The Apache SpamAssassin Project, <http://spamassassin.apache.org/>
- [6] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati, "An open digest-based technique for spam detection," In Proceedings of the 4th IEEE international conference on peer-to-peer computing, 2004.
- [7] Allister Cournane, Ray Hunt, "An analysis of the tools used for the generation and prevention of spam", Computers & Security, Volume 23, Issue 2, Pages 154-166, March 2004.
- [8] Spamassassin.org, "Spam Mails Archive," 2002-2005, <http://spamassassin.org/publiccorpus/>
- [9] MessageLabs, <http://www.messagelabs.com/>
- [10] MessageLabs Intelligence 2005 Monthly report, <http://www.messagelabs.com/intelligence>
- [11] Symantec, <http://www.symantec.com/>
- [12] Kevin Johnson, "Internet Email Protocols: A Developer's Guide," Addison-Wesley, January 15, 2000
- [13] RFC 822, <http://www.ietf.org/rfc/rfc0822.txt>

- [14] RFC 1123, <http://www.ietf.org/rfc/rfc1123.txt>
- [15] RFC 821, <http://www.ietf.org/rfc/rfc0821.txt>
- [16] RFC 1869, <http://www.ietf.org/rfc/rfc1869.txt>
- [17] RFC 2635, <http://www.ietf.org/rfc/rfc2635.txt>
- [18] RFC 2505, <http://www.ietf.org/rfc/rfc2505.txt>
- [19] RFC 2821, <http://www.ietf.org/rfc/rfc2821.txt>
- [20] RFC 2822, <http://www.ietf.org/rfc/rfc2822.txt>
- [21] RFC 1521, <http://www.ietf.org/rfc/rfc1521.txt>
- [22] RFC 1522, <http://www.ietf.org/rfc/rfc1522.txt>
- [23] Tom Merritt, "What is Email Spoofing?", May 09, 2000.
<http://www.techtv.com/screensavers/answerstips/story/0,24330,2566233,00.html>
- [24] Email Bombing and Spamming,
http://www.cert.org/tech_tips/email_bombing_spamming.html

