ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed
Computing Group

Spamato Revolutions

Provide spam filter capabilities to
every available mail client using a
Spamato Mail Proxy

Andreas Wetzel

Diploma Thesis

June 1, 2004 – September 30, 2004

Supervising Professor: Prof. Dr. Roger Wattenhofer
Supervising Assistants: Nicolas Burri, Keno Albrecht

Preface

Abstract

In this thesis we present the Spamato Mail Proxy.

Spamato is a collaborative spam filter system that acts as a stand alone component offering interfaces for mail clients and for spam filters. The Spamato Mail Proxy's task is to allow for the connection of arbitrary mail clients to the spam filter system.

Today's email clients retrieve mails either through POP or IMAP. Therefore, the Spamato Mail Proxy supports both of these protocols as well as encrypted communication through SSL tunnels. The mail client connects to the Spamato Mail Proxy which transparently forwards the connection to the real mail server. Emails processed by the Spamato POP3 Proxy receive two new header lines which indicate the result of the filter process. Spam mails detected by the Spamato IMAP proxy are automatically moved to a dedicated spam folder.

Since collaborative spam filter systems like Spamato depend on user feedback, an SMTP Feedback Proxy is also introduced that allows the user to provide feedback to the system by forwarding messages to the proxy. Again, ordinary messages are is transparently forwarded to an external SMTP server.

Acknowledgement

First of all, I would like to thank my parents and my brother for their support during all the years of my studies.

I am also thankful to all people who proof read this thesis and gave helpful hints and ideas.

I am highly grateful to my supervising assistants Nicolas Burri and Keno Albrecht for their extensive support during the last four months. Working together with you guys was a lot of fun.

I am also grateful to Prof. Dr. Roger Wattenhofer and his team for giving me the possibility to write this thesis in a very pleasant environment.

Last but not least I would like to thank my coffeebreak and tabletop football companions for our (brain-)refreshing meetings.

Contents

1 Introduction

Email spam is by far the most common way of distributing unsolicited advertisements. It involves sending identical or nearly identical messages to a large number of recipients. Unlike legitimate commercial email, spam is generally sent without the explicit permission of the recipients.

This thesis introduces a new mail client interface to SPAMATO – an existing, collaborative spam detection and filtering tool. Such systems do not rely on algorithms and rules to classify email messages but rather on humans. In simple words, the more users work with a collaborative spam filter system, the better it gets. Therefore, as many email clients as possible must be connected to the world of SPAMATO. As a first step, plug-ins for the two most used mail clients (Mozilla Mail Reader and Microsoft Outlook) have been developed. In this thesis we go on a step further and introduce the SPAMATO MAIL PROXY to support all email clients without the ability to integrate plug-ins.

1.1 Spam in Numbers

Spammers are capable to send more and more messages every day because of the constantly increasing bandwidth and computing power. According to the last statistics published by Brightmail [Bri04], the largest commercial anti spam company before the where absorbed by Symantec, in January 2004 60% of all mails they checked were spam. Their latest inquiry in July 2004 even showed a spam rate of 63%.

According to MessageLabs [Mes04], in July 2004 72% of all emails sent where spam. In September 2004 they found almost 1.2 billion spam emails which constitute about 64% of the emails checked.

These incredibly huge numbers make clear that the surge of spam is still growing and that more than something needs to be done.

1.2 Outline

The remainder of this thesis is organised as follows.

Chapter 2 provides an overview of the SPAMATO system the developed Mail Proxy depends on. The available filters as well as the supported mail clients are

briefly described. Then, the idea behind the developed Spamato Mail Proxy and its features are described in Chapter 3. Chapter 4 and 5 present the implemented Proxy services for the POP and IMAP protocol in more detail. It is shown where they hook into the protocol and how the filter mechanism is triggered. Chapter 6 is devoted to the SMTP Proxy that provides a rudimentary user feedback to the Spamato core.

Finally, in Chapter 7 the four month's work is summarised before Chapter 8 finishes this thesis by pointing out some possible improvements to the newly developed Spamato Mail Proxy.

2 Spamato

This thesis is based on th SPAMATO system – a collaborative spam filter system [Bur04, Sch04] initially implemented by Nicolas Burri and improved by Simon Schlachter.

In this chapter, the current SPAMATO system is summarised.

2.1 Framework

The SPAMATO system has been designed as a spam filter framework. Users of the system classify messages as good (ham) or bad (spam) messages. SPAMATO's task is to try to remove mails that have been classified as spam from as many other mail boxes as possible.

The framework is designed to handle multiple different spam filters at the same time. They all are accessed through a common Java interface. Currently, the SPAMATO consists of two filters (URL and Razor) which are briefly described here. An overview of the structure of the initial system is shown in Figure 2.1.

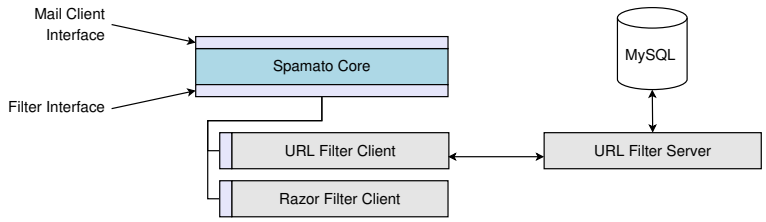

Figure 2.1: The setup of the original SPAMATO system.

The whole framework (referred to as SPAMATO core) is implemented in Java [Sun04a] and therefore independent of the underlying platform and operating system.

2.2 URL Filter

This filter consists of an "URL Filter Client" and an "URL Filter Server". The client part of the filter collects all domain names (used in URLs) in an email and

use them to calculate a fingerprint, which then gets submitted to the server. The server maintains a list of fingerprints of known spam mails. Therefore it then compares the received fingerprint to the ones stored in the database and returns the result according to the lookup result. If the fingerprint matches one from the database, the mail is considered as spam.

2.3 Razor Filter Client

Razor [Raz98] is a distributed, collaborative spam detection and filtering network. As in the Spamato system, users report mail messages that are identified as spam or revoke those that have mistakenly been marked as spam.

In fact, the Razor client works in a similar way like the URL Filter client does. It is designed to support several different algorithms to calculate a message's fingerprint. Two of them are open source: The first one is called *Ephemeral* and calculates a fingerprint based on random parts of the messages text and is currently the only one implemented in the Spamato Razor client. The second one is called *Whiplash* and quite similar to the URL filter but does not treat the URLs as one identifier but rather every single URL is taken as an indicator by its own.

2.4 Supported Mail Clients

Currently, only the Mail Reader from the open source Mozilla Suite [Moz98] and Microsoft Outlook are supported by the Spamato system. Buttons to configure the filter system and to report/revoke marked messages are added to the mail client's tool-bar providing a very simple and intuitive interface to the user.

2.5 Summary

Spamato is a collaborative spam filter system that does not rely on algorithms and rules to classify email messages but on humans. Every user reports his spam messages to the community and may in return benefit from the community helping him to filter his own mails. Up to now there are two implemented filters and the Mozilla Mail Reader and Microsoft Outlook are the only two supported mail clients.

3 Spamato Mail Proxy

As a collaborative spam filter system, SPAMATO needs to support as many mail clients as possible. The more mail clients are supported, the more users are reachable. And the more users that use a collaborative spam filter system, the better it gets.

Up to now, only tow mail readers have been supported by the SPAMATO system via plug-ins. The initial version of the spam filter included a plug-in for Microsoft Outlook. Another plug-in for the Mozilla Mail Reader is currently under development. For those applications, the use of plug-ins is obviously the way to go, since it allows the most straightforward access to messages. Unfortunately there are many more mail readers that do not support plug-ins, like pine or the (still) wide-spread Microsoft Outlook Express. Therefore, there is a need for a generic approach to support those clients.

3.1 Goal

The main goal of this thesis is to provide the possibility to connect any mail client to the SPAMATO system. This is achieved by using a proxy server. Mail clients no longer connect directly to a mail server but to the SPAMATO MAIL PROXY which transparently connects to the real mail server. The proxy's task is to fetch mails from the server and to filter them before they get delivered to the client.

Nowadays, email clients fetch new messages either via POP or IMAP. Consequently, these two protocols must be supported by the proxy.

Since collaborative spam filter systems like SPAMATO depend on user feedback, there is also a need for a feedback channel that enables the users to report spam mails and to revoke *false positives*[1]. We provide such a response channel by sending mails from the mail client to the proxy because all clients are necessarily supporting the Simple Mail Transfer Protocol (SMTP). The corresponding SMTP-proxy passes mails not affected by SPAMATO directly to an external mail server while catching reports/revokes for itself.

An overview over the whole system is provided in Figure 3.1.

[1] A message that a user wants to receive, but the spam filter system falsely tags as spam, is referred to as false positive.

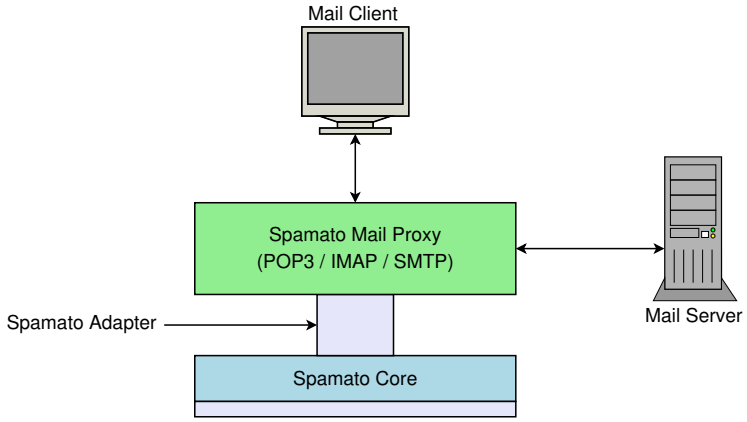

Figure 3.1: The setup of the developed Spamato Mail Proxy.

3.2 Design Philosophy

In order not to re-implement a complete client and server for the POP3 and IMAP protocols (which might have been easy for POP, but rather hard for IMAP), the proxy is – in simple words – implemented as a good "man-in-the-middle". It monitors the connection on mail protocol level waiting for some *interesting* commands trigger the filter-mechanism; everything else is piped through. In this context *interesting* means that a mail is accessed and has to be filtered. Since the two protocols are rather different, the *interesting* commands are described in more detail in chapters 4.2 and 5.2.

3.3 Spamato Adapter

All accesses from the Spamato Mail Proxy to the core are centralised into a single wrapper class for more flexibility. Therefore, connecting the Mail Proxy to another spam filter (that provides a Java interface) should be more easy.

3.4 SSL Encryption

POP and IMAP support data encryption through SSL tunnels for better security (usually referred to as POPS and IMAPS). Since SSL encryption is done on a lower level of the communication protocol, the whole process happens transparently to the proxy.. The only difference is that the proxy listens on a Java SSLServerSocket instead of a standard ServerSocket. Additionally, connections to the real mail server are established by using a Java SSLSocket instead of a standard Socket.

4 POP3 Proxy

The SPAMATO POP3 Proxy has to transparently pass client commands to the servers and the corresponding answers back. Each accessed mail is immediately spam checked before the clients access is performed.

4.1 The POP3 Protocol

Post Office Protocol version 3 (POP3, described in RFC 1939) is an application layer Internet standard protocol used to retrieve email from a remote server to a local client over a TCP/IP connection. Nearly all individual Internet service provider email accounts are accessed via POP3.

The earlier versions of the POP protocol, POP (informally called POP1) and POP2, have been rendered obsolete by POP3. In contemporary use, the less precise term POP often refers to POP3 in the context of email protocols.

Email clients using POP3 generally connect, retrieve all messages, store them on the user's PC as new messages, delete them from the server, and then disconnect, even though have an option to leave mails on the server.

For a simple illustration of the POP3 protocol, think of a (real life) letter box. Then mail checking is as easy as opening the box first, then check out if there is a new letter inside, if yes, grab it and close the letterbox.

Like many other older Internet protocols, originally POP3 only supported an unencrypted login mechanism. Although the plain text transmission of passwords in POP3 is still common, POP3 currently supports several authentication methods to provide varying levels of protection against illegitimate access to a user's mail box. The implemented POP3 Proxy only supports basic plain text authentication.

4.2 The POP3 Command Set

The basic POP3 protocol only of a small number of commands. Each command received from the client is transparently piped from the client to the server and vice-versa. Except the only *interesting* command `RETR`, which is used to retrieve a single mail from the server. Then the server answers with the corresponding mail (if available) in plain text.

4.3 Concept

As stated before, the proxy waits until the client sends an *interesting* `RETR` command. The proxy pipes the command directly to the server as usual but does not return the result to the client immediately. First the whole message is intercepted and sent to the SPAMATO core to spam check it. Next, two header lines are added to the message before the proxy delivers it to the mail client. An overview over the whole filter process is shown in Figure 4.1.

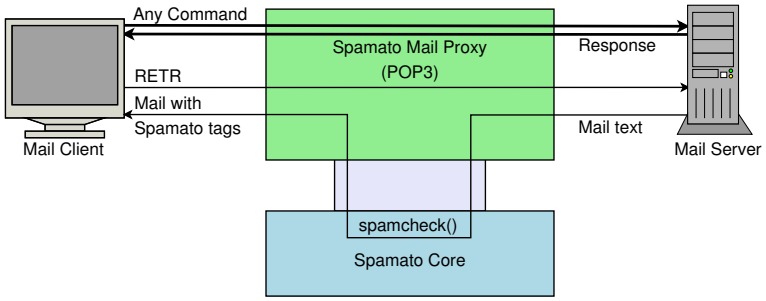

Figure 4.1: The POP3 Proxy's concept. Thick arrows represent the ordinary communication (pipe through) while the thin ones symbolise the *interesting* `RETR` command.

```
1 X-Spam-Checked-By: Spamato Mail Proxy v0.9
2 X-Spam: YES/NO
```

Listing 4.1: Added headers by SPAMATO POP3 Proxy

The first line denotes that SPAMATO has processed the message while the second one indicates if the configured filter system regards this mail as spam.

Picking up the letter box example, the functionality of the POP3 proxy can be described like this: Instead of checking for new letters yourself, you send your dog (serving as the proxy) which collects the letters for you. The (certainly intelligent) dog marks the messages as spam by biting a hole through the letter. On every letter you now see the dogs "I was here"-slobber-tag, so you know he checked it (line 1 in listing 4.3). Finally – if you completely trust your dog – you do not even have to open the perforated letters (line 2).

4.4 Implementation

The POP3 Proxy used in the SPAMATO MAIL PROXY is based on RFC 1939 which describes the basic POP3 functionalities. Proposed extensions from related RFCs have not been considered.

4.4.1 Overview

The implementation of the SPAMATO POP3 Proxy consists of the following three main components:

Server-Thread: This component creates a Java ServerSocket and waits for incoming connections. A new Connection-Thread is instantiated and started for every connection.

Connection-Thread: A single connection from a mail-client is encapsulated in this component. It establishes a connection to the real POP3 server, pipes data back and forth and waits for an *interesting* `RETR` command sent by the client.

Command-Parser: All client commands received by the proxy are parsed and classified so that the Connection-Thread can decide how to react.

4.4.2 Problems and Challenges

Altogether, the POP3 Proxy was easier to implement than expected. The only problem encountered – reading answers from the server – is briefly described here.

Single-line vs. Multi-line Answers

Most of the POP3 commands result in an answer only consisting of a single line, but some of them get a reply with more than one (obviously, fetching a whole message is a good example). Originally, the proxy service should only know about those commands that are *interesting* just passing by all the rest. As a result, the proxy may provide exactly the same functionality to the client as the real server does.

After sending a command to the server, the proxy reads one line from the server answer. Unfortunately, there is no way to determine at that point if there are some more lines to fetch. The only workaround was to provide a lookup table for command answers since for every command the answers type (single- / multi-line) is defined in the RFC. The drawback of the solution is that the proxy does no longer support the whole command set of the server but only those defined in the lookup table. Our testings with several clients and servers never revealed a problem related to that problem.

5 IMAP Proxy

The Spamato IMAP Proxy has to transparently pass client commands to the servers and the corresponding answers back. Each accessed mail is immediately spam checked before the clients access is performed.

5.1 The IMAP Protocol

The Internet Message Access Protocol (IMAP) is an application layer Internet standard protocol used to access emails on a remote server.

IMAP was designed by Mark Crispin as a modern alternative to the widely used POP3 email retrieval protocol. Fundamentally, both of these protocols allow an email client to access messages stored on an email server. But IMAP provides more features such as:

- Support for multiple clients simultaneously connected to the same mailbox.
- Access to MIME parts of messages and partial fetch.
- Keeping message state information on the server.
- Access to multiple mailboxes (folders) on the server.

Furthermore, the messages on an IMAP server are not only numbered sequentially starting at one (as in POP3), but every message has its own unified message identifier (UID) which uniquely identifies that message.

5.2 The IMAP Command Set

The IMAP protocol implemented in the Spamato Mail Proxy is accurately called IMAP version 4 revision 1 (IMAP4rev1) and is described in RFC 3501.

IMAP4rev1 has a long list of available commands. Luckily, there is only one command for fetching messages (or message parts) which may occur in two different forms.

FETCH requires two parameters. First, a list of messages to retrieve information for must be provided. Second, it needs to be specified which data items

from the selected messages should be returned by the server. This may be – just to mention some of them – the full message body, partial header fields, message flags and/or the message's unique identifier (the documentation of `FETCH` in the RFC is of about the same size as the whole RFC about POP3).

UID FETCH works the same as `FETCH`. The only difference is that messages are referenced by their unique identifier rather than by their current sequence number on the server.

5.3 Concept

The Spamato IMAP Proxy waits for an *interesting* `(UID) FETCH` command which then is kept back until the filter process is done.

Because parsing and reassembling `FETCH` requests and responses would have been rather hard, the proxy opens a new connection to the IMAP server using the JavaMail API [Sun04b]. This API provides an easy interface to fetch emails from an IMAP server as an Object that can be processed by the Spamato adapter. If Spamato identifies a message as spam, the proxy moves it into a dedicated spam folder and tags it as deleted in the current folder. Then, the client's initial `FETCH` command is passed to the server. An overview over the whole process is shown in Figure 5.1.

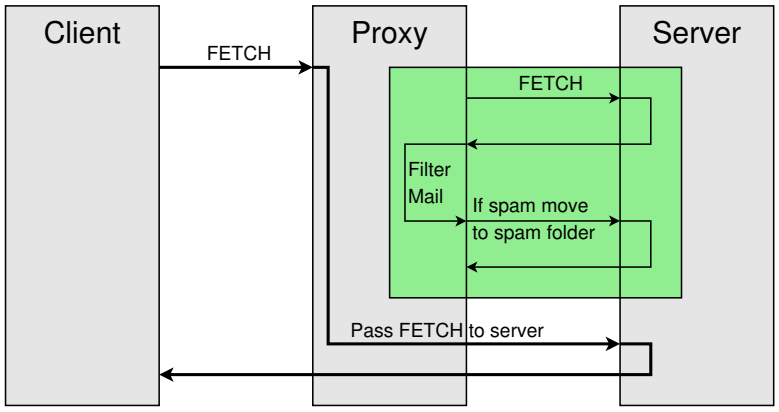

Figure 5.1: The IMAP Proxy's concept.

In the introduction, we have seen that an IMAP server supports more than one folder. The Spamato IMAP Proxy monitors each of these folders except the spam folder itself.

In order not to scan a message on every access – since it may be accessed several times before the mail gets displayed to the user – the proxy maintains a list of recently checked messages. The email gets scanned again only if a configurable period of time has passed since the last check (an it is still unread).

5.4 Implementation

The IMAP Proxy used in the Spamato Mail Proxy is based on RFC 3501 which describes the IMAP functionalities in detail. All other RFCs concerning extensions to the protocol have not been considered.

5.4.1 Overview

Because the IMAP protocol supports multiple clients simultaneously connected to the same mailbox, the client must be capable of notifying the other clients about what was going on. For example if one user deletes a message, all other clients must be informed of that. Therefore the IMAP protocol is designed asynchronous and two listeners are needed. One that waits for client commands and another one that reacts to spontaneous server status messages. The implementation consists of the following five main components:

Server-Thread: This component creates a Java ServerSocket and waits for incoming connections. A new Connection-Thread is instantiated and started for every connection.

Connection-Thread: A single connection from a mail-client is encapsulated in this component. It establishes a connection to the real IMAP server, pipes data back and forth and waits for an *interesting* `(UID) FETCH` command sent by the client.

Listeners: The proxy has its own listeners on the server- and on the client-side as well. Some special cases like raw data pass-trough also require some minimal communication between those two listeners.

Command-Parser: All client commands received are classified and partially parsed that the Connection-Thread can easily decide how to proceed depending on the result.

Mail-Checker: Because of the rather complicated protocol – to keep it as simple as possible – a separate component to check messages has been introduced. Mails that need to be checked are processed by this component trough a second connection opened to the server. This problem is discussed later on.

5.4.2 Problems and Challenges

IMAP Protocol

The IMAP protocol is very complicated to outline and hard to understand. Also, there are not many free server implementations available. Even the Java Apache Mail Enterprise Server (James) [Jam04] currently does not have IMAP support.

Asynchronous communication. The IMAP protocol specifies that not only the client may initiate data transfers, but also the server may submit spontaneous status messages. The proxy could not be implemented as simple as the one for POP3. Each IMAP command has its own tag that uniquely identifies a request since multiple commands may be processed simultaneously and the client has to know which answer belongs to which command (sent before). The challenge was to keep the two listeners synchronised with each other and to keep track of their different command tags.

Raw data transmission. A server answer may contain a tag specifying that some raw data will follow (the whole message's text for example) immediately. Thus, a single answer from the server may be interrupted several times with raw data. If the client wants to transmit raw data to the server, the procedure is slightly different. The client first has to announce that it wants to send some data. Then it has to wait until the server signals to be ready to receive the data. So parsing messages both of the client-side and server-side where two different kettle of fish.

Second server connection

The IMAP protocol allows compound and nested FETCH requests and answers. Different kinds of messages (plain-text and HTML messages) would have drastically increased the difficulty to implement message filtering on that level. Therefore, the Spamato IMAP Proxy opens a second (high level) connection to the server using the JavaMail API. That way, it is very easy to retrieve and filter any message with the drawback of two required connections.

5.4.3 Unsolved Tasks and Problems

Eliminating the second connection

As stated above, filtering a message in the Spamato IMAP Proxy requires a second connection to the mail server. Unfortunately, not every server allows more than one connection from the same IP address. Additionally, some mail clients (such as Mozilla) open multiple connections to a single server for performance reasons.

As the current proxy is unable to detect such multiple connections, each one requires an additional connection for checking mails – so the number of open connections is always doubled by the proxy. A possible solution might be to perform the high level JavaMail calls through the already opened connection, if that is possible at all.

Possible timeout

When a client is opening a huge mailbox, it may happen that the client does not get any feedback from the proxy in a reasonable amount of time leading to a timeout. This happens because before returning anything to the client, all messages are processed by the proxy.

If the client requests thousand mail headers, they usually come back one after another. But the current proxy design does not allow such a feedback. First all thousand mail headers are read by the proxy – some mails may be filtered – and then the normal client-request is performed. While the proxy is working, the client does not get any feedback, thus could conclude the server might be down.

6 SMTP Proxy

The SPAMATO SMTP Proxy has to transparently pass client commands to the servers and the corresponding answers back. For every sent email it has to decide in a way i f the message is dedicated to the SPAMATO core or must be forwarded to an external mail server.

6.1 The SMTP Protocol

The Simple Mail Transfer Protocol (SMTP, described in RFC 2821) is the de facto standard for email transmission across the Internet.

SMTP is a relatively simple, text-based protocol. It allows for sending messages to multiple recipients and then the message text is transferred. It is quite easy to test a SMTP server using telnet program. SMTP uses TCP port 25.

Since this protocol is designed as purely ASCII plain text one, it did not deal well with binary files. Standards, such as MIME (RFC 2045-2049), were developed to encode binary files for transfer through SMTP. Today, most SMTP servers support the 8BITMIME extension, permitting binary files to be transmitted almost as easily as plain text.

6.2 The SMTP Command Set

The basic SMTP protocol implemented in SPAMATO only consists of a small and perspicuous set of commands. At least, there is a small set of mandatory commands that must be supported. Today, a lot of extensions are possible, as described in RFC 2821, but they where of no interest for this task since we only want to catch a small number of mails for some specific SPAMATO receivers. Again, we want to provide the same functionality (including extensions) from the server to the client by just passing commands and answers back and forth.

The only *interesting* command during an SMTP transaction for the proxy is the `RCPT TO` command which is briefly described here.

RCPT TO tells the server to whom the following message is addressed and mostly appears right after the sending user is authenticated. That command may occur several times to define more than one recipient for a message.

6.3 Concept

When a client opens an SMTP connection to the proxy, the proxy immediately opens one to the external SMTP server. That allows us to provide any kind of authentication mechanism through the proxy.

Any occurrence of a `RCPT TO` command triggers the proxy to check the recipients address. If the receivers host does not match a configurable host name (referred to as the SMTP Trigger Host Name) all, subsequent command are piped through the proxy an nothing more has to be done.

If the mail is addressed to the Spamato Trigger Host name, the connection to the external SMTP server is stopped and further maintained by the proxy. Then, with the receiver name supplied the proxy decides whether the message is reported spam or a revoked *false positive*. (Currently, only the email addresses `report@<triggerhost>` and `revoke@<triggerhost>` are understood by the proxy. More may be introduced for future tasks.)

Finally, the proxy collects the full message and notifies the Spamato core to handle the given message as spam (on reports) or as a good mail (on revokes).

An overview over the whole process is shown in Figure 6.1.

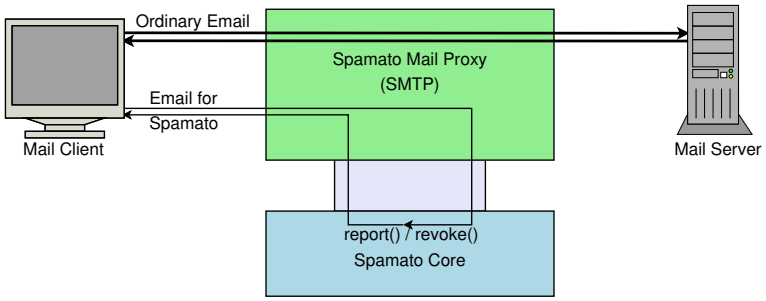

Figure 6.1: The SMTP Proxy's concept. Thick arrows represent the ordinary communication (pipe through) while the thin ones symbolise the case where Spamato receives the mail.

6.4 Implementation

The SMTP part in the Spamato Mail Proxy is based on RFC 821. The newer RFC 2821 introduces nothing new about the basics but is more focused on topics like user authentication and extensions.

6.4.1 Overview

The implementation of the SPAMATO SMTP Proxy consists of the following three main components:

Server-Thread: This component creates a Java ServerSocket and waits for incoming connections. A new Connection-Thread is instantiated and started for every connection.

Connection-Thread: Everything concerning a single SMTP connection from a mail client is encapsulated in this component. It establishes a connection to the real SMTP server, passes data back and forth and decides whether an email is feedback for the SPAMATO core or must be forwarded to the external SMTP server.

Feedback: If the proxy receives a mail destined for the SPAMATO core (currently only *reports* and *revokes*), this component fetches the message and notifies corresponding to the email receiver.

7 Summary

This thesis adds an new mail client interface to Spamato – an existing, collaborative spam detection and filtering system.

The Spamato Mail Proxy was designed and implemented in order to support all email clients that do not directly support plug-ins. It operates as a good "man-in-the-middle" between a mail client and an external mail server, transparently for both systems. For more privacy, the proxy also supports communication through SSL encrypted tunnels.

The first implemented mail retrieval proxy (POP3) constitutes a very simple form of mail filtering. The Spamato Mail Proxy adds tags to the mail header that allow a mail client to sort the incoming mails according to them.

Developing the IMAP proxy was by far the most challenging part of the whole thesis, even if the resulting code may not look like it. The protocol has so much complex capabilities, it was really demanding to keep track them. Testing the proxy was quite hard, because the IMAP protocol offers multiple ways to determine if there are new mails on a server and to fetch them. Additionally, some clients do heavy caching (Mozilla) while others consequently write all changes to the server immediately (K-Mail).

From the complexity point of view may be compared to POP3. Everything is kept small and easy to work with. So again, developing the SMTP proxy required for feedback to the Spamato core was a simple task.

Summarised, I have had a very good time working at the Distributed Computing Group. On the one hand because working with Spam and Protocols turned out to be very interesting and on the other hand the people surrounding me were very enjoyable. My two advisors Nicolas Burri and Keno Albrecht gave me all the support I could ever hope for and always had a good sense of humour.

8 Future Work

8.1 Spamato Mail Proxy

8.1.1 Improving the feedback capabilities

Currently the Spamato system can only receive user feedback through the implemented SMTP proxy. If a user has a spam mail in his inbox that was not detected by the system, he has to forward it to a certain email address (for example `report@local.spamato`) which allows the SMTP Proxy to recognise it as a spam report. Now, every mail client adds some lines to a forwarded mail in a different matter. It may also remove or change some content (mostly headers) of those mails. As a result, the reported mail is no longer identical to the spam message initially received and thus does not help other users to filter out the same message.

Particularly the IMAP proxy could provide a very handy feedback channel. Every mail manually moved to the spam folder could automatically be treated as a spam report. In return, every mail moved out of that folder to another place (except the trash folder) might be treated as a revoke. Using such a transparent feedback system, a user might not even realise that he is providing important feedback to the system.

8.1.2 Centralise the Mail Proxy

Instead of every user running his local copy of the proxy, the proxy could run somewhere else on a local LAN or the Internet where it may benefit of fast broadband connections to mail servers. Advantages of such a system are:

Reuse of resources: One single instance (or a small number of instances) of Spamato could filter mails for thousands of users.

Less traffic: At least for the IMAP protocol, spam messages may be filtered out even before they reach the users computer. They do not have to be transmitted through a possibly slow connection to the user.

Easy updates: One update has to be done and all users instantly profit of the new features.

Other difficulties might show up, such as where to store the users' configuration/statistics. Moreover every user must be clearly authenticated to prevent the user of foreign proxy accounts. But these things have been solved several times before in other environments, therefore it should not be too difficult to do it again.

8.1.3 TLS Encryption

A lot of today's mail clients also support TLS as an encryption mechanism. Instead of using a different port number for encrypted communications, the client initially opens a normal plain text connection. Right after the connection is initiated, the client may send a `STARTTLS` command that tells the server to switch to encrypted communication. This mechanism could easily be implemented into the current POP, IMAP and SMTP proxy.

9 Bibliography

[Bri04] Brightmail - Spam Percentages and Spam Categories. `http://www.brightmail.com/spamstats.html`, 2004.

[Bur04] Nicolas Burri. SPAMATO - A Collaborative Spam Filter System, 2004.

[Jam04] Java Apache Mail Enterprise Server. `http://james.apache.org`, 2004.

[Mes04] MessageLabs. `http://www.messagelabs.com`, 2004.

[Moz98] Mozilla Suite. `http://www.mozilla.org`, 1998.

[Raz98] Vipul's Razor. `http://razor.sourceforge.net`, 1998.

[Sch04] Simon Schlachter. SPAMATO Reloaded - Trust, Authentication and More in a Collaborative Spam Filter System, 2004.

[Sun04a] Sun Microsystems. Java 2 platform, standard edition. `http://java.sun.com/j2se/index.jsp`, 2004.

[Sun04b] Sun Microsystems. Javamail. `http://java.sun.com/products/javamail`, 2004.