Final Report Project “Spamabwehr II”

W. Gansterer, M. Ilger, A. Janecek, P. Lechner, J. Strauß

Technical Report FA384018-5

Institute of Distributed and Multimedia Systems

Faculty of Computer Science

University of Vienna, Austria

April 2006

This report summarizes the results of the project FA 384018 “Spamabwehr II” of the Institute of Distributed and Multimedia Systems at the University of Vienna, funded by IPA, mobilkom austria and UPC Telekabel.

About the Authors

Project "Spamabwehr II" was launched in May 2005 at the Institute of Distributed and Multimedia Systems at the Faculty of Computer Science at the University of Vienna.

The team:

Dr. Wilfried Gansterer (project leader),
Michael Ilger, Mag. Andreas Janecek, Mag. Peter Lechner and Jürgen Strauß.

Contact for Project "Spamabwehr II":

phone: +43-1-4277-39650
e-mail: Each team member can be contacted at *firstname.lastname@univie.ac.at*

The Institution:

The Faculty of Computer Science (Fakultät für Informatik) is lead by Dean Prof. Dr. Günter Haring. The Institute of Distributed and Multimedia Systems, headed by Prof. DDr. Gerald Quirchmayr, is one of the institutes within this faculty.

Institute of Distributed and Multimedia Systems
University of Vienna
Lenaugasse 2/8, A-1080 Vienna (Austria)

Table of Content

Executive Summary

This report summarizes the results of the research project FA384018 “Spamabwehr II” which was launched in May 2005 at the Institute of Distributed and Multimedia Systems at the University of Vienna. It was supported by Internet Privatstiftung Austria (IPA), mobilkom austria, and UPC Telekabel.

This document is structured as follows.

Section 1 provides an introduction into the topic by discussing definitions, summarizing recent spam statistics (like total amount of spam, sources of spam and damage caused by spam), and reviewing the relevant economical and technical background.

Section 2 takes a deeper look at the spammers’ business model and discusses spammers’ cost and revenue models. Several scenarios for decreasing spammers’ profit by limiting the amount of sent messages are analyzed and the SpamSim tool, used for simulating these scenarios is presented.

Section 3 categorizes anti-spam methods into three classes of approaches, pre-send methods, post-send methods and new protocols, reviews a categorization of potential anti-spam strategies and summarizes an evaluation of two popular state of the art software tools, the public domain e-mail client Mozilla Thunderbird and the rule set based spam filter SpamAssassin. Different trainings schedules for the Mozilla Thunderbird are compared and the performance of SpamAssassin tests is evaluated.

Section 4 describes a new approach developed in this project for fighting outgoing spam, based on a token bucket concept. The main idea is to limit the number of outgoing e-mail messages in such a manner that regular e-mail users are not affected while the business model of spammers is harmed. Different approaches of an adaptation of the token bucket concept for spam defense are discussed and the implementation of this concept is summarized.

Section 5 discusses a new prototype for spam defense and spam detection. The approach focuses on strategies that are capable of detecting spam as early as possible and therefore reduce the waste of resources caused by spam.

The Appendix contains a short documentation (in German) of the prototype we developed in this project.

1. Introduction

Before talking about "spam" we need to answer a couple of important questions. It is important to have a definition of spam in the first place. Additionally we will provide a number of statistics which illustrate the total amount of spam messages and the main regions of origin. Furthermore we include estimations of the damage caused by spam. Finally we show which approaches we identified as the most promising ones.

1.1. What is "Spam"?

First, we need to carefully define some central terminology. Obviously the commonly used word "spam" was derived from a completely different context and is not a scientific term. In this section, we summarize official and/or more technical terminology used for spam, such as unsolicited commercial e-mail (UCE) or unsolicited bulk e-mail (UBE).

Unsolicited Commercial E-Mail (UCE): *"E-Mail containing commercial information that has been sent to a recipient who did not ask to receive it"* [1], or: *"Unsolicited e-mail is advertising material sent by e-mail without the recipient either requesting such information or otherwise explicitly expressing an interest in the material advertised."* [2]

Unsolicited Bulk E-mail (UBE): *"E-Mail with substantially identical content sent to many recipients who did not ask to receive it. Almost all UBE is also UCE."* [1], or: *"Unsolicited Bulk E-Mail, or UBE, is Internet mail ('e-mail') that is sent to a group of recipients who have not requested it. A mail recipient may have at one time asked a sender for bulk e-mail, but then later asked that sender not to send any more e-mail or otherwise not have indicated a desire for such additional mail; hence any bulk e-mail sent after that request was received is also UBE."* [3]

In our view taken in this project, no solicited e-mail is considered to be spam. However, there may be spam which is not sent out in bulk or which does not involve (direct) commercial interest. In extreme cases, classification of an e-mail message as spam often becomes a highly subjective decision and it is very difficult – if not impossible – to establish common criteria covering a wide range of affected users. Nevertheless, based on the statements mentioned above, we identify three central features of an e-mail which we consider defining properties of spam (not always all three of them have to apply):

1. It is unsolicited, that is, the receiver did not request it.

2. It is sent out in bulk, that is, to many recipients.

3. Usually, there is commercial interest involved, for example, interest in advertising (and selling) some product.

1.1.1. The Amount of Spam on the Internet

The amount of spam sent over the Internet has been rising dramatically in recent years and no significant decline is to be expected in the foreseeable future. This is clearly illustrated by data about the share of spam in the total number of e-mail messages sent. In 2000, only 7% of the messages sent worldwide were spam [4], whereas in 2002 already 40% were spam [5]. The current percentage is estimated to be around 65% worldwide with much higher estimates for some regions (for example, up to 90% for the USA). According to that trend, some pessimists even announced the end of the e-mail infrastructure for 2007 [6].

Until July 2004 the anti-spam software developer Brightmail published monthly statistics about spam, as shown in Figure 1.

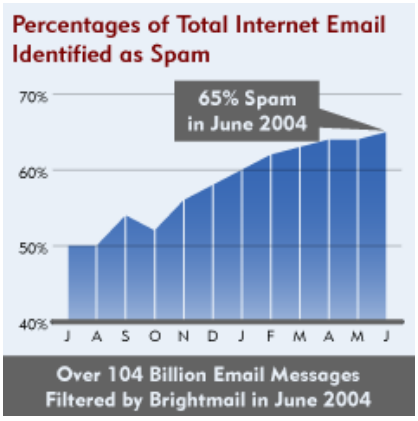

Figure 1. Percentage of e-mail identified as spam, June 2004 [7] (no newer data available)

Messagelabs [8] published Figure 2, which reflects the interaction between legislative measures / law enforcement and the percentage of spam in e-mail scanned.

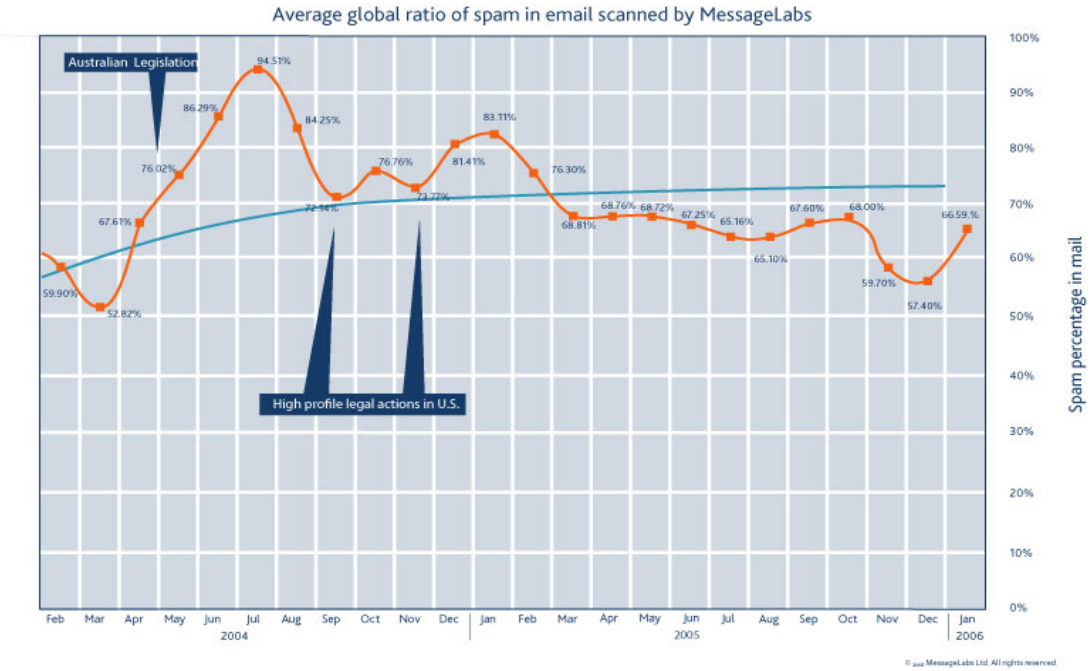

Figure 2. Interaction of legislative measures, law enforcement and percentage of spam [8]

It is clearly visible that the percentage of spam in all e-mail messages sent still has an increasing trend, but also that it tends to react significantly to the introduction of new legislative measures and to legal actions taken against spammers. This interpretation also has to be seen in the light of the conjecture that 80% of the spam sent worldwide comes from very few (roughly 200) distinct spammers [9].

1.1.2. Sources of Spam

Having illustrated that the percentage of spam in e-mail messages sent worldwide increases rapidly, we also tried to collect some statistics about the sources of spam, both in terms of geographic regions as well as in terms of Internet domains. Unfortunately, it is quite simple to forge the point of origin in an e-mail in the standard SMTP protocol and thus a distinction has to be made between the *pretended* point of origin and the *actual* point of origin of an e-mail (which may be different). Due to lack of information available, it was not possible to thoroughly judge which points of origin the sources quoted here refer to.

A ranking of the most spam producing countries (as of 24th August 2004) according to the anti-virus and anti-spam vendor Sophos [10] is shown in Table 1 in comparison to the equivalent reports of the years 2005 [11] and a news bulletin provided by Sophos in April 2006 [12].

2004		2005		2006	
1.United States	42.53%	1.United States	26.80%	1.United States	23.10%
2.South Korea	15.42%	2.South Korea	18.43%	2.China (& HK)	21.90%
3.China (& HK)	11.62%	3.China (& HK)	17.04%	3.South Korea	9.80%
4.Brazil	6.17%	4.France	3.88%	4.France	4.30%
5.Canada	2.91%	5.Canada	2.72%	5.Poland	3.80%
6.Japan	2.87%	6.Brazil	2.40%	6.Spain	3.30%
7.Germany	1.28%	7.Spain	2.28%	7.Germany	3.00%
8.France	1.24%	8.Japan	2.03%	8.Brazil	2.90%
9.Spain	1.16%	9.United Kingdom	1.51%	9.Japan	2.00%
10.United Kingdom	1.15%	10.Germany	1.34%	10.United Kingdom	1.90%
11.Mexico	0.98%	11.Poland	1.17%	11.Netherlands	1.80%
12.Taiwan	0.91%	12.Austria	0.92%	12.Taiwan	1.60%
Others	11.76%	Others	19.48%	Others	20.60%

Table 1. The top twelve sources of spam, geographically [10]

For collecting the data summarized in Table 1 researchers used honeypots[1] to collect spam. It is interesting to note that, compared to the data from February 2004, Canada reduced its rate from 6.8 % to 2.72% in 2005 and completely disappeared from the list in 2006. The USA also reduced their share of spam significantly, whereas China constantly increased its share. While making appearances in this list in 2005 Austria is currently not among the top spam senders. Other European countries such as Germany experienced a constant rise in their ranking in the last

[1] The term “honeypot” (or “spam trap”) refers to an e-mail address never published to humans. Any e-mail message sent to such addresses has to be spam.

couple of years. In general, about 40 % of the world's spam is sent out from "zombie computers"[2] [5].

Figure 3, provided by the anti-spam vendor Commtouch [13], illustrates which domains spammers preferred to send out spam in 2004 and which were the preferred domains in 2005. It shows that the percentage of spam coming from each of these known domains decreased significantly, with hotmail.com being the source of little more than two percent while yahoo.com is coming in at less than one percent.

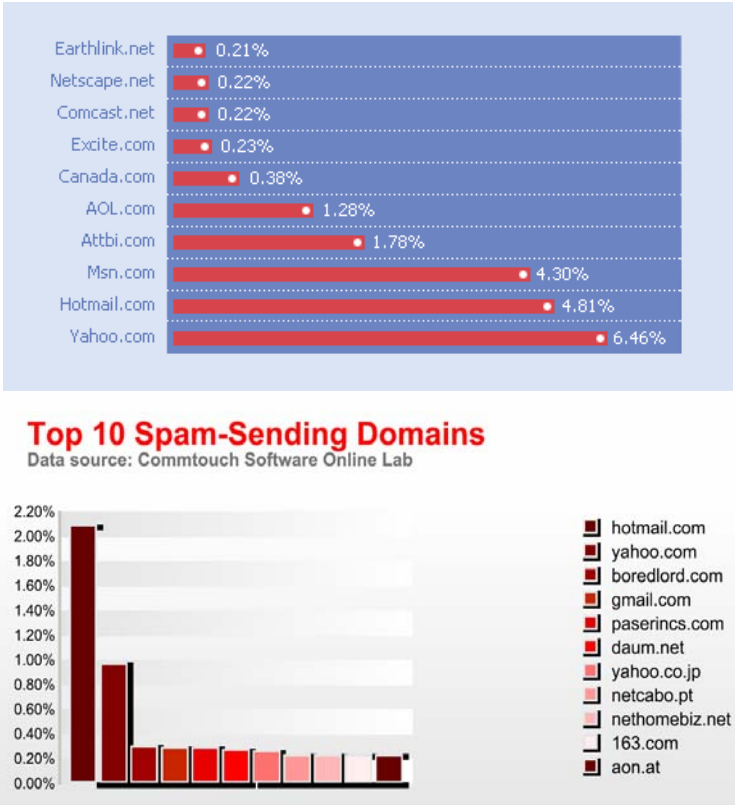

Figure 3. The top ten sources of spam (domains) [13]

Postini [14] provides some interesting statistics investigating the sources of spam and of directory harvest attacks[3]. On their graphical illustration the region of Austria and Germany seems to be among the hotspots of spammers' activity. Additionally, it also provides numbers for the daily amount of spam sent. Usually it lists a number of roughly 150 million messages checked spam messages per day.

1.1.3. The Damage Caused by Spam

This huge amount of spam not only occupies enormous resources, it obviously also causes a dramatic loss of productivity. Whereas individuals only lose bandwidth and time (which is hard to quantify), companies potentially lose much more. Ferris Research [15] estimated productivity

[2] The term "zombie computer" refers to a computer infected with viruses of all kinds and misused for sending out spam.

[3] A directory harvest attack is the theft of confidential e-mail directory information, for example of lists of e-mail addresses of all employees of an organization.

losses for US corporations with 10 billion dollar for 2003. The European Union estimated a loss of productivity of 2.5 billion Euros for 2003 [16]. Various tools to gauge losses for the own company are available online (see, for example [17] or the simpler version [18]). Recent surveys conducted by the University of Kiel (Germany) and by the OECD reported yearly costs per employee. These costs were listed as € 780 [19] or $ 874 [20] respectively, which creates a quite precise picture of the costs each company has to face due to spam.

1.2. Approaches Chosen in this Project

One of the goals of this project was the development of effective and longer lasting anti-spam methods and strategies which cannot easily be circumvented by spammers, even if the technical details are fully revealed. In particular, we have developed two technical approaches that differ in their point of action within the mail transfer process: a *pre-send* and a *post-send* method (cf. Section 2). Our pre-send method uses a token bucket adaptation to limit the sending of e-mail messages and harms the business model of spammers. A detailed discussion is given in Section 4.

The second approach is a multi-layer server-based solution for classifying incoming e-mail traffic and for detecting spam already at the incoming mail server (before the clients and individual users are involved). One central motivation for this strategy was to keep workload and the waste of resources imposed by spammers on the receiving Internet Service Provider (ISP) as small as possible. A detailed discussion is given in Section 5.

2. The Economy of Spam

The main motivation for sending spam is to make profit. Technical solutions used in common filters try to reduce the amount of spam reaching the recipients and therefore reduce the potential profits. This chapter analyzes the spammers' business model and in order to quantify the influence of some technical countermeasures like filters or a token bucket algorithm on the profit of some simple spamming cost and revenue scenarios. The first section describes the general cost and revenue model, the second section describes "SpamSim", a tool we produced for quantitatively analyzing spammers' business model and the third section describes three scenarios which we analyzed using SpamSim [21].

2.1. Cost and Revenue Factors for Spammers

Before different cost and profit models can be analyzed, the most important factors must be identified. Cost factors can be measured in real money or in a time consuming procedures. Here in our simplified model they can be classified in four categories - hardware cost *H*, software cost *S*, labor cost *L* and operating costs *O*. Figure 4 provides an overview over the most important cost factors.

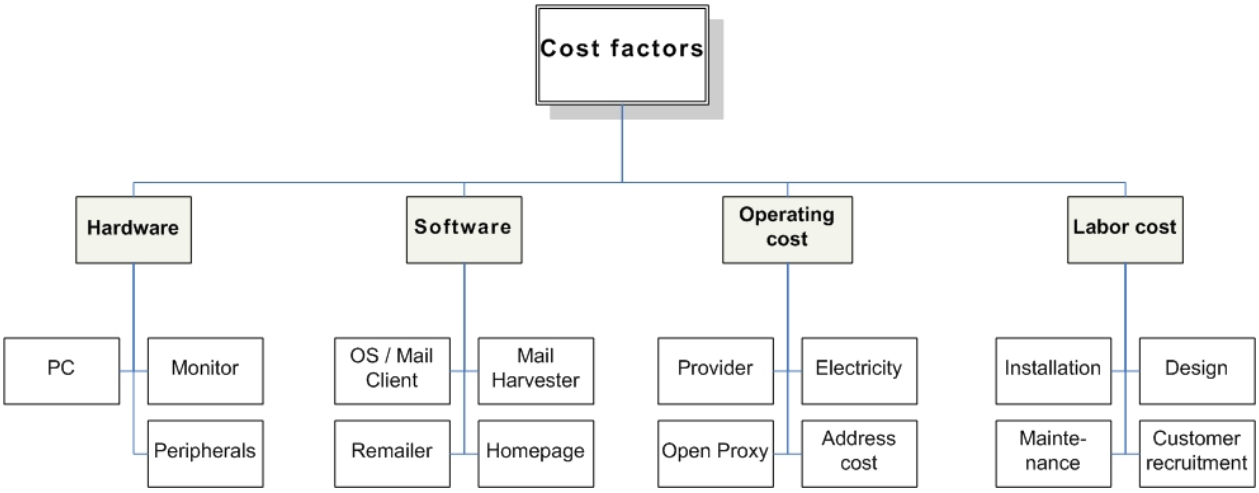

Figure 4. Cost factors of spammers

Some costs - like hardware costs are easy to determine, but others can only be estimated (like time for software installation or updates, time for spam e-mail creation, how much effort is required for customer recruitment and so on). Here a simple model is defined to show the basic procedure of sending spam and to estimate the costs of a single spam e-mail. With the help of SpamSim (see Section 2.2) different cost factors can be varied easily.

According to the above model cost *c* are defined as sum of hardware costs *H*, software costs *S*, operating costs *O* and labor costs *L*.

$$c = H + S + O + L$$

Assuming a single spammer using his own home computer, hardware costs are defined as sum of a computer *C*, a monitor *M* and peripheral devices *P*.

$$\boldsymbol{H = C + M + P}$$

Software costs can be divided into costs for basic software which each computer needs, like an operating system *OS* and costs for software for special spamming activities like remailers[4] *R*, mail address harvesters[5] *MAH* or web hosting *WH*.

$$\boldsymbol{S = OS + R + MAH + WH}$$

Operating costs are a sum of internet service cost *I*, electricity cost *E* for running the system, address collection cost *A* (addresses can be bought or self-collected) and open proxy[6] cost *OP*.

$$\boldsymbol{O = I + E + A + OP}$$

Labor costs can be divided into costs for installation *IN*, maintenance *MT*, e-mail design *MD* and costs for customer recruitment *CR*.

$$\boldsymbol{L = IN + MT + MD + CR}$$

All these cost factors must be monitored over a certain period of time (for example over a month) and afterwards be divided through the possible number of messages to get the per message cost.

While the cost factors can be estimated easily, revenue factors are very hard to define, because spammers do not declare their business model in public. There are two main payment schemes - a brokerage system, where the marketer gets a fee per sold item and a pay-per-mail campaign system (an e-mail campaign to one million customers costs a certain amount of money). As it is not easily possible to get reliable data about the former, we took data from the mail campaign system described in [21] (the revenue per message is 0.00434 Euro).

2.2. SpamSim Application

The different business models motivating the spam phenomenon have been summarized in [21] [31]. In order to design and to calibrate an anti-spam method which is capable of interfering with these business models we created a simulation tool which allows us to model the effect of various parameters on the profit achieved by spammers.

2.2.1. System Overview

Our tool implements the most important cost and profit factors and models their interdependencies. The most important questions are: How many messages must be filtered out and how many messages have to be sent out by a spammer to be profitable? By examining the

[4] A remailer is an Internet site to which you can send e-mail for forwarding to an intended destination while concealing your own e-mail address. E-mail sent through a remailer is sometimes known as anonymous e-mail.
[5] An email harvester is a software robot which spiders websites across the internet looking for e-mail addresses.
[6] An open proxy is an Internet proxy server which is accessible by unauthorized users, specifically those from elsewhere on the internet.

break-even point for spammers in terms of cost and profit, we are able to quantitatively evaluate the effectivity of anti-spam methods.

Our tool is implemented as a Windows GUI application, programmed in Visual C#, operating on the Microsoft .NET framework. Future versions may also provide the logic of this program as a web-service. To provide a flexible solution the scenarios are defined in MathML - each scenario is defined by the elements of the cost factors. MathML (Mathematical Markup Language), is an application of XML which includes additional elements to work with calculations. Our tool works in a standard and an advanced mode - the advanced mode allows fine tuning of all parameters, whereas the standard mode uses pre-defined values. The results of the simulation can be displayed as a line diagram or exported as an XML or CSV file.

2.2.2. Configuration Details

The application consists of an executable file, a library which is used to create graphics, and a configuration (config) file. Additionally, the config file provides the path to further input files. One of these input files determines the factors used for the computation, three more files are used to compute the values for the x and y axis of the diagrams.

The program itself offers four different views. These views include a preferences window to choose the right configuration and an editor which allows easy access to the scenario via an XML file. The costs can be modeled in the main window. Finally, a scenario window (compare Figure 5) provides the graphical solution.

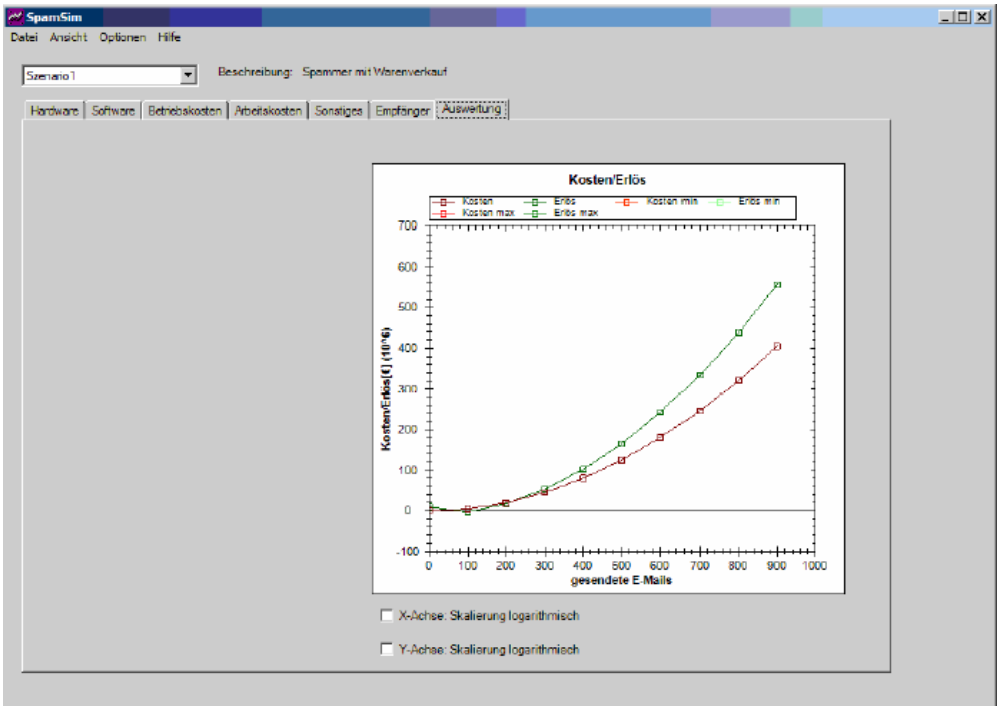

Figure 5. Graphical output of SpamSim

2.3. Selected Case Studies

In this chapter we describe three possible approaches to destroying spammers' business models. First, we analyze the influence of our token bucket approach described in [22] which dynamically limits outgoing (from ISP side) spam traffic (compare Chapter 4), second, we investigate the impact of filter performance, and third, we explore the leverage of the response rate on the spammers profit.

2.3.1. Single Spammer – Token Bucket Usage

This section shows a simple example of a spammer who uses his own home PC and a leased line[7] for spamming. The different cost factors are summarized in the following table. We assume that hardware, software, operating and working costs are monthly fixed costs. Only open proxy costs (usage of open proxies obfuscates the own IP-address) that are charged per e-mail are listed as running costs.

Type of cost	Single cost [Euro]	Economic life [month]	Cost per month [Euro]
Fixed cost			
Hardware			
Average Computer	1,000.-	36	27.78
Average Monitor	300.-	36	8.33
Peripherals	200.-	36	5.56
Software			
Operating System	1.- (open Source)	36	0.03
Harvester	50.-	36	1.39
ReMailer	100.-	36	2.78
Webhosting	20.-	1	20.00
Operating Cost			
ISP cost	49.-	1	49.00
Electricity	35.-	1	35.00
Addresses	300.-	12	25.00
Labor Cost			
Installation	30.-	12	2.50
Maintenance	300.-	1	300.00
Mail-design	300.-	1	300.00
Customer recruitment	300.-	1	300.00
		Sum:	1,077.64
Running cost			
Open Proxy	0.000125	per mail	

Table 2. Cost factors for a single spammer sending from his home PC via open proxy

[7] Leased line means normal internet access via an ISP

If the upload line has a bandwidth of approximately 256 KBit/sec and the average size of a spam message equals 4 KByte [23], a spammer is able to send around 691,200 messages per day (or 20,736,000 messages per month). Figure 6 shows the cost and the revenue of the described scenario. The cost curve is taken from the above data, the overall revenue is 0.00434 Euro per mail (compare [31]), which leads to big profits for the spammer.

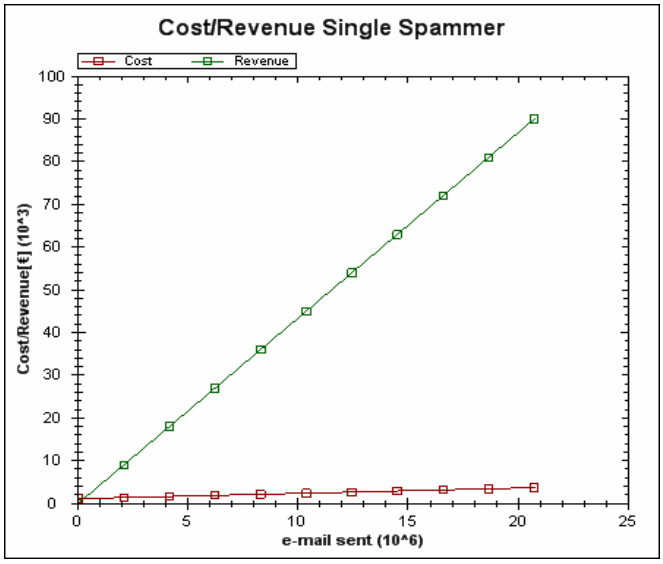

Figure 6. Cost and revenue – single spammer, no technical limitations (upper line=revenue, lower line=cost)

It is clear that the profit of a spammer depends on the number of messages sent. If the outgoing traffic is restricted to 8,400 messages per day, the diagram looks as depicted in Figure 7. It can be seen that in this case the cost and revenue curves intersect near 250,000 messages per month. Below the spammer does not make any profit or even does not have any marginal income.

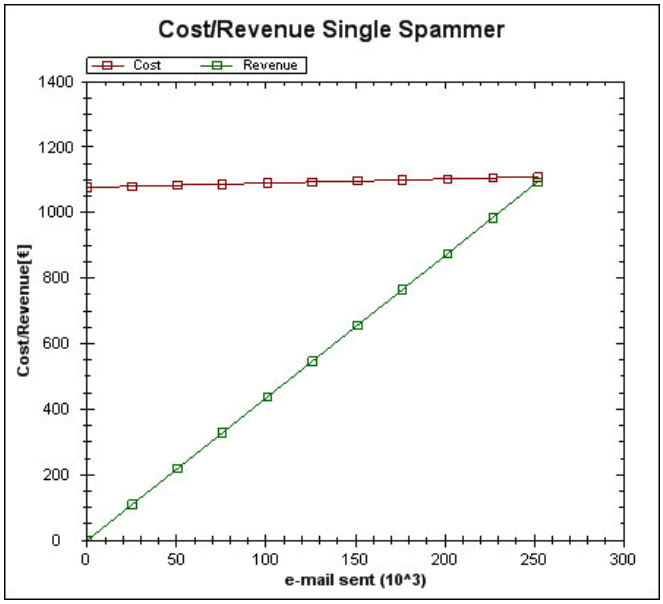

Figure 7. Cost and revenue – single spammer limit = 8,400 messages per day / 252,000 per month (upper line=cost, lower line=revenue)

The following table shows the cost and revenue values for 252,000 messages per month (equals above 8,400 messages per day).

e-mail sent	cost	revenue
0	1,077.64	0
25,000	1,080.48	108.50
50,000	1,083.61	217.00
75,000	1,086.73	325.50
100,000	1,089.86	434.00
125,000	1,092.98	542.50
150,000	1,096.11	651.00
175,000	1,099.23	759.50
200,000	1,102.36	868.00
225,000	1,105.48	976.50
250,000	1,108.61	1,085.00

Table 3. Cost and revenue values – single spammer limit = 8400 messages per day

With the given cost and revenue data and the given scenario of a single spammer we can summarize that a sending limit below 8,400 messages per day will render spamming unprofitable. For implementing such a limit in an acceptable way, we suggest a token bucket algorithm (see Chapter 4).

2.3.2. Single Spammer Rents a Server – Filter Performance

A second scenario is based on a single spammer, who uses a leased line to connect to a leased server called LegalMail[8]. The spammer connects himself remotely to the LegalMail server, the server then sends out user defined messages. Unlike in scenario one there are no running costs – all the monthly costs can be divided by the number of e-mail messages sent and then compared to the revenue.

Type of cost	Single cost [Euro]	Economic life [month]	Cost per month [Euro]
monthly fixed cost			
Hardware			
Average computer	1,000.-	36	27.78
Average Monitor	300.-	36	8.33
Peripherals	200.-	36	5.55
Software			
Operating System	1.- (open Source)	36	0.03
Homepage	200.-	36	5.56
Operating costs			
ISP cost	35.-	1	35.00
Electricity	17.-	1	17.00
Addresses	200.-	12	16.67

[8] http://www.americaint.com/bulk-email-software/legalmail/legalmail.html

LegalMail	1,000.-	1	1,000.00
Working costs			
Installation	30.-	12	2.50
Maintenance	300.-	1	300.00
Mail-design	300.-	1	300.00
Customer recruitment	300.-	1	300.00
		Sum:	2,018.42

Table 4. Cost factors for a single spammer renting a server

LegalMail guarantees that (under certain conditions, e.g. size limitations, not too many invalid addresses, addresses ordered by provider) a minimum of 30,000,000 messages is sent out per month. Monthly fixed costs are computed for the scenarios summarized in Table 4 as 2,018.42 Euro, one message generates 0.00434 Euro of revenue. Without any countermeasures the revenue is huge (compare Figure 8).

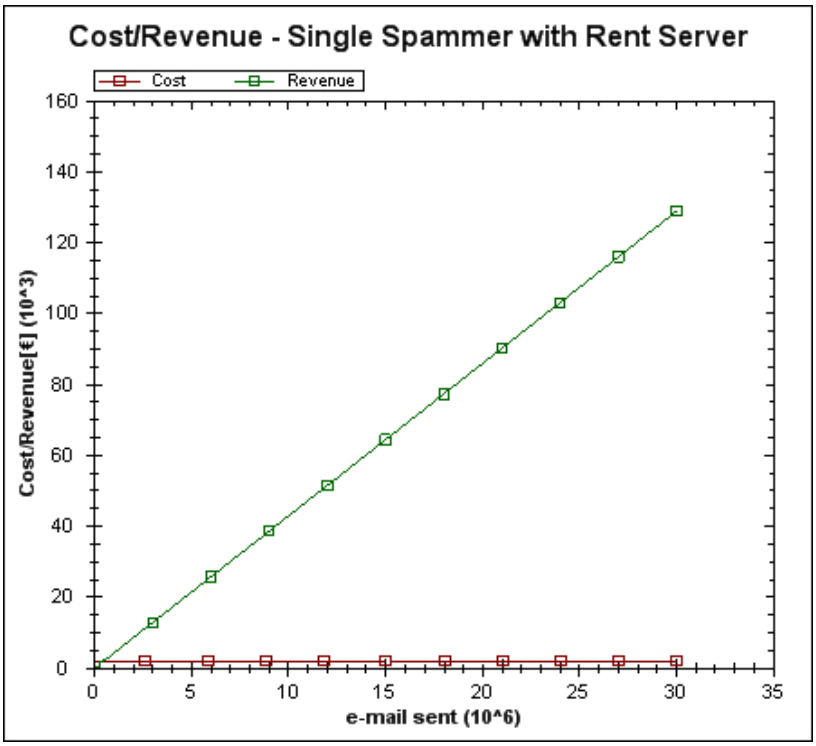

Figure 8. Cost and revenue – single spammer server with no countermeasures (upper line= revenue, lower line=cost)

In this scenario we illustrate the potential impact of the spam filter performance on spammers' business. Here filter performance is denoted as the detection rate of an anti-spam-filter. The better the detection rate the smaller is the profit. With a detection rate of 90% spammers continue making large profits (compare Figure 9).

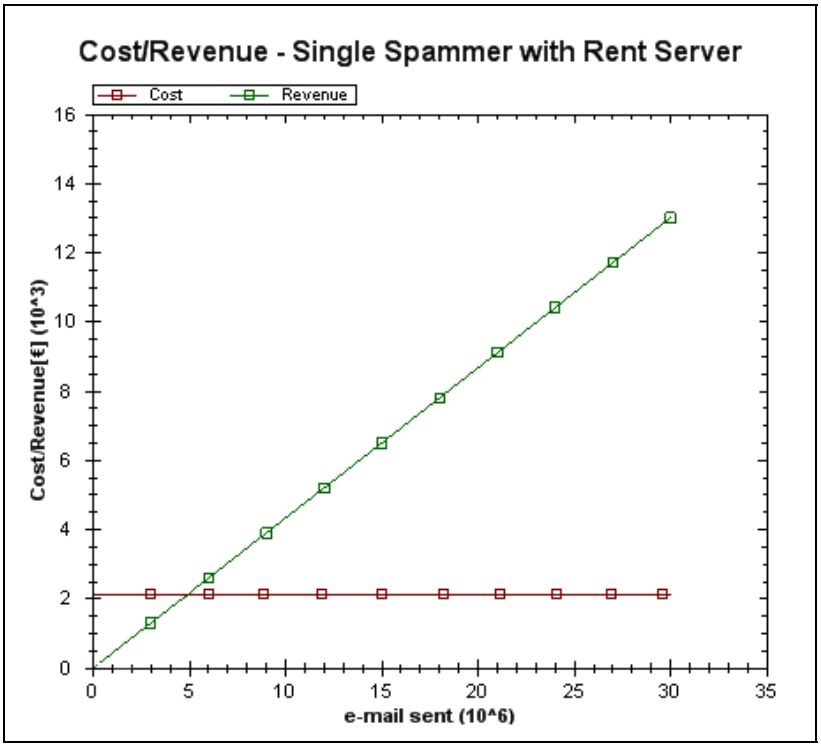

Figure 9. Cost and revenue – single spammer server (filter performance = 90%) (upper line= revenue, lower line=cost)

Only if the filter achieves a detection rate of 98.45 % or more the spammers' costs exceeds the revenue.

e-mail sent	cost	revenue
0	2,018.42	
3,000,000	2,018.42	201.81
6,000,000	2,018.42	403.62
9,000,000	2,018.42	605.43
12,000,000	2,018.42	807.24
15,000,000	2,018.42	1,009.05
18,000,000	2,018.42	1,210.86
21,000,000	2,018.42	1,412.67
24,000,000	2,018.42	1,614.48
27,000,000	2,018.42	1,816.29
30,000,000	2,018.42	2,018.10

Table 5. Cost and revenue – data sheet single spammer with server (filter performance = 98.45%)

For the given configuration in Table 4 a filter must have a detection rate of 98.45% or higher to harm the spammers business model.

2.3.3. Single Spammer – Response Rate

A third scenario describes a spammer who has fixed costs of 883.19 Euro per month and running costs of 0.000125 Euro per e-mail. This spammer is not paid per e-mail, but per sold item instead. In this example we describe the influence of the response rate on spammers' business model. If

the spammer gets a provision of 8 Euro per item sold and the response rate is 0.036 % (the spammer must send out 2,777 messages to get one response) (compare [24]) the graph looks as depicted in Figure 10.

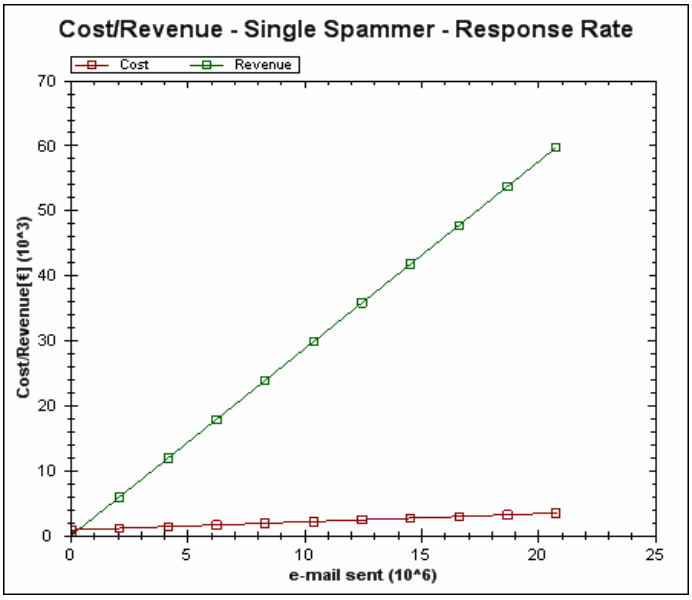

Figure 10: Single spammer – response rate = 0.036% (upper line=revenue, lower line=cost)

Figure 11 shows the cost and revenue factors for a response rate of 0.001%.

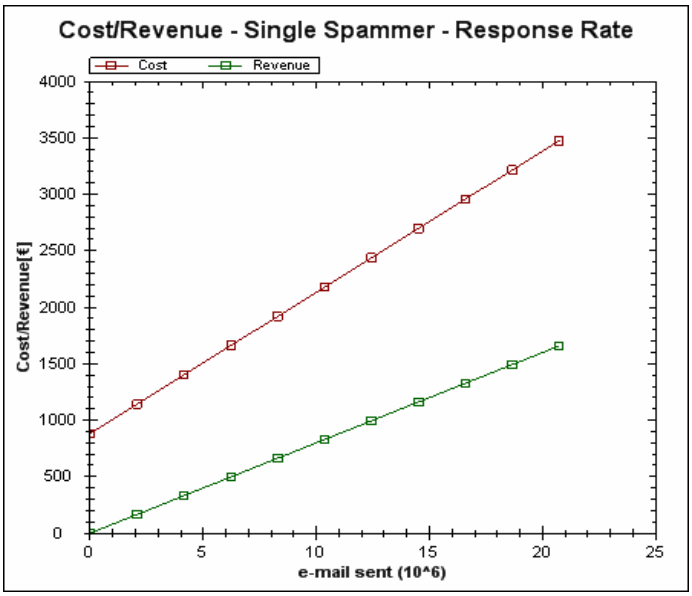

Figure 11. Single spammer – response rate = 0.001% (upper line=cost, lower line=revenue)

With a response rate of 0.001% (compare [25]) a spammer would in this scenario never make any profit. The break-even is around a response rate of 0.002% (which corresponds to one response per 50,000 messages sent-out).

e-mail sent	cost	revenue
0	883.19	0

2,073,600	1,142.39	331.78
4,147,200	1,401.59	663.55
6,220,800	1,660.79	995.33
8,294,400	1,919.99	1,327.10
10,368,000	2,179.19	1,658.88
12,441,600	2,438.39	1,990.66
14,515,200	2,697.59	2,322.43
16,588,800	2,956.79	2,654.21
18,662,400	3,215.99	2,985.98
20,736,000	3,475.19	3,317.76

Table 6. Single spammer – response rate = 0.002 %

2.3.4. Conclusion

The investigations in this project clearly illustrate that the spammers' business model is based on sending out large quantities of messages. Several factors influence the number of messages which actually reaches potential customers and thus the business model of spammers. We quantitatively analyzed the influence of three of those factors which are well suited as the basis for anti spam strategies – sending limitations, message filters and response rates – in the context of simple quantitative models. For these simple models, it was shown that the business model underlying spamming can be affected negatively by sending limits, by filter detection rates or by reducing response rates.

The models investigated in this project serve as prototypical examples and allow for illustrating the basic principles of the spam economy. For concrete quantitative statements applying in practical situations, more complex models will be needed. For this purpose we developed the simulation tool "SpamSim", which facilitates the efficient construction and evaluation of various anti spam methods in the context of more complex models.

3. *Anti-Spam Methods*

Anti-spam methods can be categorized according to their point of action in the mail transfer process. Consequently, we distinguish three classes of approaches. *Pre-send* methods act before the e-mail is transported over the network, whereas *post-send* methods act after the e-mail has been transferred to the receiver. A third class comprises new protocols, which subsumes approaches based on modifying the transfer process itself. A categorization of anti-spam methods is given in Figure 12.

Most of the existing research in the area of anti-spam methods has focused on post-send methods, more specifically, on filtering messages at the receiver side. Important examples are black- and whitelisting, Bayes filters [26], or rule sets (e.g. SpamAssassin [27]). From the point of view of the individual user, some post-send methods are able to achieve reasonably satisfactory results provided they are trained, tuned and maintained permanently. This effort required for maintaining satisfactory performance of post-send methods is one of their main disadvantages. An even more substantial drawback is the fact that these methods act – by definition – *after* the message has been transferred and received. Consequently, a big part of the damage caused by spam is done *before* post-send methods can become active. They are not able to reduce the overhead, the waste of bandwidth, processing power, memory and time caused by spam. Moreover, privacy issues may arise since most of them have to access the message content for satisfactory results.

Suggestions for new protocols extending or replacing SMTP have been made in order to overcome its deficiencies in the context of the spam problem, for example, AMTP [28] or IM2000 [29]. The main problem is that a common worldwide agreement on changing the protocol used for e-mail transfer and a coordinated transition seems unrealistic in the foreseeable future.

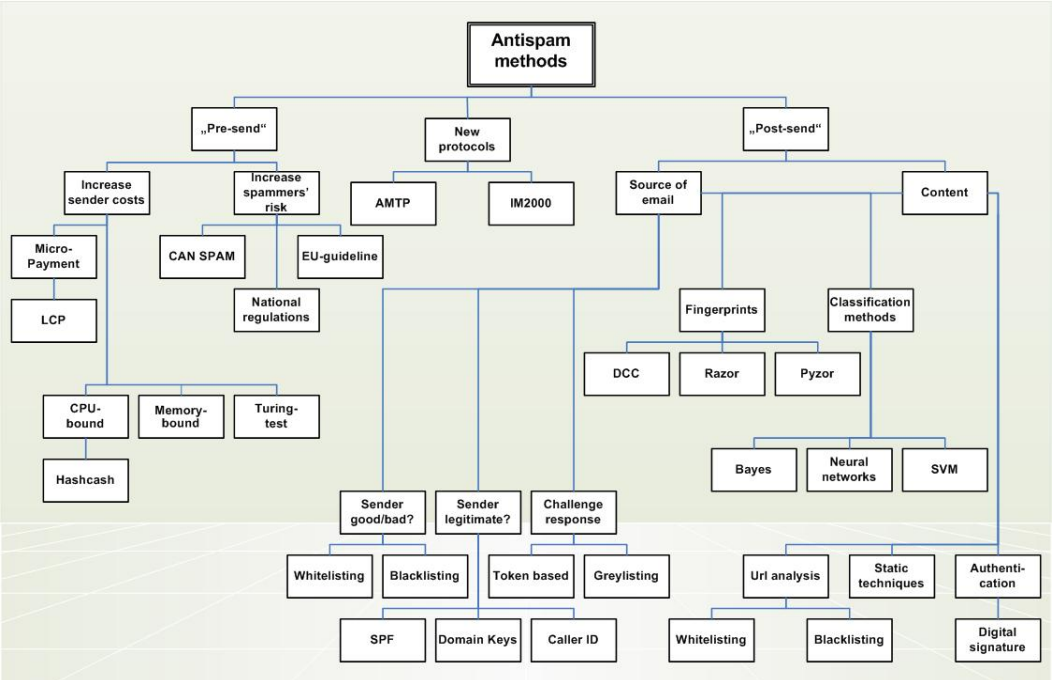

Figure 12. A Categorization of anti-spam methods [24]

In the following we summarize some detailed performance evaluations of two very popular and widespread open source software tools performed in this project. Both tools fall in the category "post-send" filtering approaches: the Bayes filter implementation in the public domain e-mail client Mozilla Thunderbird [30] and the more comprehensive rule set based spam filter SpamAssassin [27] (which also includes a Bayes filter).

3.1. Mozilla Thunderbird

Mozilla Thunderbird uses a Bayesian classifier to identify spam messages, i.e. a statistical approach based on textual features. The underlying idea is to compute a conditional probability for an e-mail being spam based on the words (tokens) it contains. Any Bayes filter must be trained, at least initially. In order to evaluate the dependency of the performance of the Thunderbird Bayes filter on the frequency of training and, more generally, on the training effort, we evaluated the filter performance with three different training schedules over a period of 52 weeks.

More specifically, we compared the following three training schedules: (*i*) "static"- train the Bayes filter only once (initially), (*ii*) "dynamic four weeks" – train the Bayes filter periodically every four weeks, and (*iii*) "dynamic two weeks" – train the Bayes filter periodically every two weeks. Intuitively, one would expect the best detection rates when the filter is trained most frequently (biweekly) and a significant decrease of the detection rate over time if the filter is trained only initially. It was our goal to substantiate this intuition and to quantify it using empirical evaluations. For a complete description of the test setup see [31].

Summarizing, we note that

1. as expected, the detection rate of the static variant is significantly lower than the detection rates of the dynamic variants.

2. the detection rates of the two dynamic variants do not differ much (which is not very surprising).

3. there seems to be no general decreasing trend in the detection rate of the static variant (which is rather surprising).

4. after a first sharp decrease of all detection rates after around five weeks, the performance of all three training schedules improves again (which is very surprising!).

The results of our evaluation are depicted in Figure 13. At first sight, especially observations 3 and 4 are very surprising, because they contradict the intuitive expectation. However, a more detailed investigation of the test data reveals some interesting aspects which have been unnoticed so far (due to the big volume of spam data it is usually impossible to screen the messages manually): There are some periods when the fraction of "bounces" (error messages such as delivery failure notices) in the data from the spam traps is very high. This is due to the fact that spammers sometimes seem to randomly put harvested e-mail addresses also into the "From:" field of a message (and not only into the "To:" field). In Figure 13 we show the approximate fraction of bounces in all the test samples as the blue (light) parts in the bars of the bar charts. Note that the samples between weeks one and five have a very high portion of bounce messages

(around 50%!), whereas the bounce fraction between weeks six and thirteen is very low. After that, this fraction increased again. These bounces distort the results and have a big impact on the classification performance. For a more detailed discussion see [31].

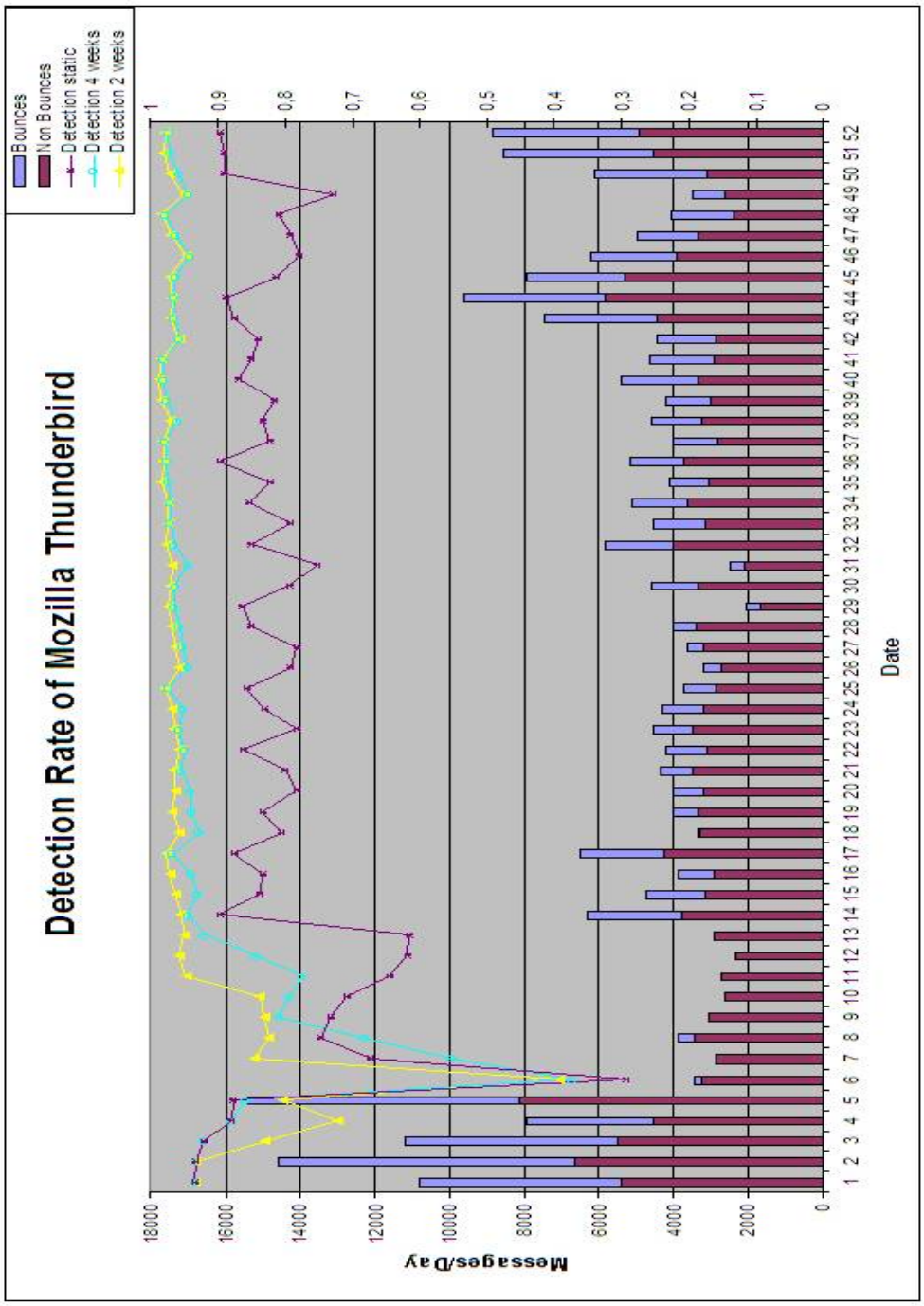

Figure 13. Detection rates of Mozilla Thunderbird (Bayes filter) for different training schedules over a 52 week period

3.2. SpamAssassin

SpamAssassin (SA) [27], developed by the Apache Software Foundation, is a de-facto standard of a rule set based spam filtering system. Using a big set of tests and rules (including a Bayesian classifier) SA extracts features (currently more than 700) from every incoming e-mail message, comprising header, body, and the full text of the message (see Figure 14). Each test is assigned a certain value (positive, if the test points to a spam message, negative, if the test points to a ham message). Since SpamAssassin v3.0, these values are derived using a neural network trained with error back propagation. The overall rating of a message is computed by summing up all values of the tests. If this sum exceeds a user-defined threshold the message is classified as spam. The tests are trained for a standard threshold of 5.0. Increasing this threshold will decrease the spam detection rate but can be used to work against false positives; decreasing it will increase the spam detection rate, but the false positive rate as well.

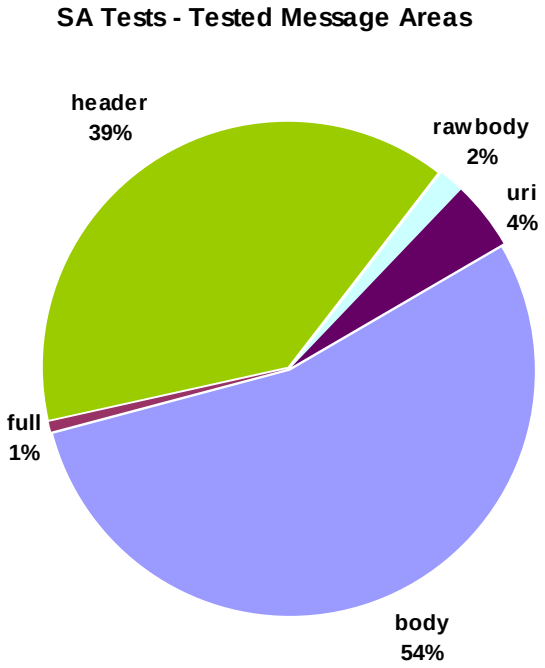

Figure 14. Areas covered by SpamAssassin Rules

SpamAssassin tests. As mentioned before, SpamAssassin uses more than 700 tests and rules. Although this seems to be a huge amount, the number of tests with useful information decreases as experimental analysis shows. The following experiments were performed with a hand selected test set of about 10.000 e-mail messages. Only about the half of the tests triggered at least once. As SA contains many language specific tests, this does not seem to be very surprising. Moreover, the SA test set is unbalanced: about 96% of all tests test for spam and only about 4% test for ham. (see Figure 15).

Figure 15. SA Test Categories

Tests triggering incorrectly. Looking at the tests we can identify well discriminating tests as well as tests that often trigger false, i.e. tests, that trigger for the wrong group of messages. These tests are a big problem as they can lead to a wrong classification of messages. There are two groups of wrong classifications, false positives and false negatives. False positives (ham classified as spam) are a bigger problem than false negatives (spam classified as ham), as important messages may get lost. Table 7 contains a list of tests triggering incorrectly with a wrong classification of more than 2% (with respect to the overall amount of messages). It is clearly visible, that there are some tests that trigger more often for the wrong group than for the correct group, and therefore can lead to wrong classifications of messages. Fortunately, for SA users, most of these tests have a very low value and therefore only a small effect for the classification of the message.

NAME	Correctly triggered	Incorrectly triggered
NO_FORMS	5.25%	9.17%
BLANK_LINES_70_80	1.34%	8.31%
HTML_MESSAGE	51.34%	7.17%
BAYES_50	11.92%	6.88%
MIME_BASE64_BLANKS	2.96%	5.63%
MIME_QP_LONG_LINE	9.71%	5.25%
SUBJECT_DRUG_GAP_P	3.67%	3.79%
MIME_HTML_ONLY	32.48%	3.70%
BLANK_LINES_90_100	0.19%	2.91%
MISSING_SUBJECT	1.42%	2.54%
MSGID_FROM_MTA_ID	1.06%	2.42%

Table 7. Top Wrong Triggering Tests of SA

Top triggering tests. A list of the top triggering tests can be found in Table 8. This list does not distinguish between right and wrong classification, but contains the percentages for (*i*) correct classifications, (*ii*) incorrect classifications, and (*iii*) classifications overall, i.e. number of mails, for which this test has triggered. This list also contains some tests from Table 7, i.e. tests that often triggered incorrectly. These tests must be treated very carefully. On one hand these tests trigger often and therefore contain much information, on the other hand they can easily lead to wrong classifications.

NAME	Correctly triggered	Incorrectly triggered	Triggered overall
BAYES_00	87.91%	0.12%	88.02%
BAYES_99	76.70%	0.04%	76.74%
ALL_TRUSTED	71.17%	0.47%	71.64%
URI_OFFERS	62.11%	0.00%	62.11%
RAZOR2_CF_RANGE_51_100	60.06%	0.00%	60.06%
RAZOR2_CHECK	59.70%	0.09%	59.80%
URI_UPPER_LOWER	56.98%	0.00%	56.98%
HTML_MESSAGE	51.34%	7.17%	58.50%
RCVD_IN_SORBS_DUL	47.60%	0.47%	48.08%
RCVD_IN_NJABL_DUL	37.02%	0.19%	37.21%
MIME_HTML_ONLY	32.48%	3.70%	36.19%
URI_UNSUBSCRIBE	27.39%	0.00%	27.39%
INVALID_MSGID	23.20%	0.08%	23.28%
RCVD_IN_DSBL	23.09%	0.05%	23.15%
MPART_ALT_DIFF	19.30%	0.60%	19.90%
SUBJECT_IN_BLACKLIST	16.32%	0.00%	16.32%
MISSING_MIME_HB_SEP	13.88%	1.19%	15.07%
HTML_90_100	13.03%	0.55%	13.58%
MSGID_OUTLOOK_INVALID	11.99%	0.00%	11.99%
BAYES_50	11.92%	6.88%	18.80%
RCVD_NUMERIC_HELO	11.71%	0.11%	11.82%
DRUGS_PAIN_OBFU	10.90%	0.04%	10.95%
HTML_FONT_BIG	10.43%	0.82%	11.25%
DNS_FROM_AHBL_RHSBL	9.97%	0.06%	10.03%
MIME_QP_LONG_LINE	9.71%	5.25%	14.96%
MIME_BOUND_DD_DIGITS	9.65%	0.00%	9.65%
HTML_IMAGE_RATIO_02	8.63%	0.00%	8.63%
RCVD_IN_NJABL_PROXY	7.92%	0.15%	8.07%
HELO_DYNAMIC_IPADDR	7.32%	0.00%	7.32%
HTML_TAG_EXIST_TBODY	6.86%	0.05%	6.92%
FORGED_OUTLOOK_TAGS	6.64%	1.07%	7.71%
FORGED_YAHOO_RCVD	6.64%	0.05%	6.69%
DOMAIN_RATIO	6.63%	0.02%	6.65%
MSGID_SPAM_CAPS	6.62%	0.03%	6.66%
MSGID_RANDY	6.26%	0.00%	6.26%
HTML_40_50	6.22%	0.40%	6.62%
HTTP_ESCAPED_HOST	6.21%	0.03%	6.25%
HTML_80_90	6.05%	0.76%	6.81%
HTML_IMAGE_ONLY_12	5.89%	0.00%	5.89%
HTTP_CTRL_CHARS_HOST	5.79%	0.00%	5.79%

Table 8. Top 50 Triggering Tests of SA

4. Outgoing Spam Prevention

A new approach to fight outgoing spam we developed in this project is based on the adaptation of the token bucket concept. Existing methods in this area of outgoing spam prevention have various drawbacks. In particular, they impose restrictions which may also affect regular e-mail users. Ideally, one would like to have a method which (*i*) limits the sending of e-mails in such a way that the business model of spammers is harmed while (*ii*) its effects are unnoticed by regular e-mail users. Our concept, summarized in the following, is designed accordingly. In order to motivate its structure we need to analyze potential spamming scenarios first.

Let us assume that an individual S intends to send spam to a number of recipients. S is connected to the Internet through ISP X. We distinguish two relevant scenarios:

1. S sends e-mail through the outgoing mail server of X.

2. S connects to an open proxy, an open relay or to a third party e-mail provider outside X's network directly and uses it to send out e-mail.

In principle, two more situations are conceivable. Theoretically, S could connect to an open proxy, to an open relay or to a third party e-mail provider *inside* X's network and use it to send out e-mail. S could also run his own e-mail server (or a spamming tool) locally. However, ISPs tend to prohibit the installation of an open proxy, open relay or of a private e-mail server inside their network. Consequently, the latter two situations mainly involve questions concerning the enforcement of the internal structure and regulations of the ISP and were thus beyond the scope of this project. The strategy we proposed in [22] comprises two core components in order to handle both of the relevant scenarios.

A token bucket component limits the flow of e-mail through the outgoing mail server of X. If parameterized properly, this happens unnoticed by the regular e-mail user. A component consisting of whitelists (or blacklists) restricts or inhibits the outflow of e-mail through channels other than the outgoing mail server of X (open proxies, open relays or third party e-mail providers).

In the following, we describe how to design a token bucket shaper for outgoing e-mail traffic such that regular e-mail users are within the envelope, whereas (profitable) spamming is outside the envelope.

4.1. How Many Tokens per Message?

When applying a token bucket strategy to spam prevention, the system can either use one token per message or it can determine the number of tokens needed based on the size of the e-mail (which corresponds to using one token per fixed amount of data to be sent). We focused on the former variant since the essential information to be transferred in spam tends to be compact (often only a URL), and thus many spam messages tend to be rather short. Nevertheless, generalizations are easily possible.

4.2. Design of the Envelope

The commercial success of spamming is directly related to the number of spam messages sent out per time unit (see Section 2). This is explained by the underlying business model: Spammers try to send out big numbers (millions) of often small-sized messages. Although the relative response rate tends to be rather low – between 0.00001% and 0.35% [32], this is sufficient in terms of absolute numbers (assuming a fixed income associated with each response) as long as the number of messages sent out is large enough. Due to this dependency on high sending rates, a limitation of the number of outgoing messages is an effective measure. In the case studies presented in Section 2 it was illustrated that without limitations in the number of messages sent out a single spammer can create large daily revenues (basically only limited by the bandwidth of the internet connection). A limit of a few hundred messages per day is normally sufficient to reduce the spammers' potential revenue below the costs.

4.3. Restricting Outgoing E-mail Traffic

A simple and straightforward solution would be to put a static limit on the number of e-mails which can be sent out per time unit, for example, allowing each user to send out at most 100 e-mail messages per day, or on the number of recipients for a single e-mail (cf., for example, the terms of service of Yahoo[9]). Obviously, such rigid limits easily cause notable restrictions for regular e-mail users, because the sending of e-mail is usually not distributed uniformly over a day. Generally speaking, for the success and widespread acceptance of any measure against spam it is crucial to be as transparent and imperceptible as possible for a regular e-mail user. In summary, the objective has to be the following: Limit the number of e-mail messages sent out in order to compromise the spammers' business model without negatively affecting a regular e-mail user (ideally, a regular e-mail user should not notice anything).

A token bucket mechanism is a suitable strategy for achieving this goal, because it provides a very flexible way of limiting the number of e-mail messages which can be sent out. It can accommodate for traffic bursts, and, if parameterized and adjusted properly, can achieve the objectives formulated above. We have investigated proper parameterization and adjustment of a token bucket mechanism for controlling outgoing e-mail traffic.

4.4. Parameterization

A token bucket is specified by two parameters: capacity β and token rate ρ. In order to determine good choices for these two parameters, a rough estimate of the number of e-mail messages which a regular user (not a spammer) sends out per day is needed. Data from ISPs indicates that this number tends to be below 30 messages per day (averaged over all customers, see [33]). A global estimate may be found on the basis of estimates for the total worldwide daily volume of e-mail messages, the total number of internet users, and the overall percentage of spam. Radicati[10] estimates that around 130 billions e-mail messages are sent out per day. The number of internet users worldwide is currently roughly 950 Millions,[11] and Postini[12] estimates that more than 80%

[9] www.yahoo.com
[10] www.radicati.com
[11] www.internetworldstats.com

of e-mail messages are spam. Based on these numbers, we can estimate that around 26 billions non-spam ("ham") messages are sent out worldwide per day. Consequently, on average every internet user sends fewer than 30 ham messages per day. We (conservatively) estimate the average sending rate s per user as $s \approx 50$ messages per day. Due to legitimate (ham) mass mailers, the sending rate of most users can be expected to be significantly lower. Consequently, with a choice of $\beta = 2 * s$ and $\rho = 2 * s / 86\,400$ (per second) we can limit the outgoing traffic to at most 100 messages per day, which is clearly below the volume needed for commercially successful spamming. At the same time, regular users would usually not notice any restrictions, because they may send out single bursts of up to twice their average message volume.

4.5. Implementation

Our token bucket approach is implemented as a policy delegation server for the open source mail server Postfix [34]. For each user, a triplet consisting of (*i*) the mailbox name, (*ii*) the time t_0 of the last successful submission of the "RCPT TO:" command (SMTP status code "250"), and (*iii*) the number T of tokens currently available in his bucket is stored. Every time a client transfers a message to the outgoing mail server, a plug-in is triggered after the execution of the "RCPT TO:" command within the SMTP dialogue. If the e-mail has r recipients, the plug-in is executed r times. Each execution of the plug-in computes the number of available tokens based on the following parameters: capacity β (= maximum number of tokens available per user), token consumption per recipient T_c, and token growth per time unit ρ. Based on the current system time t (when the sending takes place) and on t_0, the new amount T of tokens currently available in the bucket can be computed using the following equation:

$$T(t) = min\,(T(t_0) + (t - t_0) * \rho;\ \beta)$$

This number is the basis for the decision whether the e-mail can be sent to a certain recipient or not. If $T(t) - T_c >= 0$, the recipient is accepted, the number of available tokens is updated in the user's triplet as $T := T(t) - T_c$, and the plug-in returns status code "250 OK" to the mail server. If $T(t) - T_c < 0$, the e-mail cannot be sent, the recipient is refused, and the plug-in returns "554 Not enough tokens available" back to Postfix. Alternatively, one could *delay* sending of this e-mail until enough tokens are available.

4.6. Conclusion

In [22], we presented a concept for dynamically limiting the number of outgoing e-mail messages, to be implemented at an outgoing e-mail server. It is based on the adaptation of a token bucket mechanism. This concept has various advantages over existing approaches. Most importantly, it provides a very flexible and adaptive way of limiting outgoing e-mail traffic. In contrast to static approaches, traffic bursts can be taken into account. We have discussed how to parameterize and tune our concept such that the restrictions it imposes are virtually unnoticed by regular e-mail users while at the same time the business model of spammers is compromised. Thus, in contrast to existing approaches, our strategy does not affect all e-mail users, but selectively targets spammers. As a disadvantage of the proposed mechanism it has to be voted that the impact of this concept will only be significant if it is adopted by many ISPs, ideally

[12] www.postini.com

globally. However, this “weakness” is present in all approaches for outgoing spam prevention at the sender side.

5. *A Prototype for a new Anti-spam System*

In this project, we also developed a prototype for spam defense and spam detection which consists of several components. A central design goal was to minimize the negative effects caused by spam. In particular, our objective is to focus on methods that are capable of detecting spam as early as possible, ideally, before the receiving mail server has assumed responsibility for delivering the message to the final recipient. Compared to conventional (mostly client-based) spam filters where e-mail messages are classified after they have been accepted and stored, our approach significantly reduces the waste of resources caused by spam.

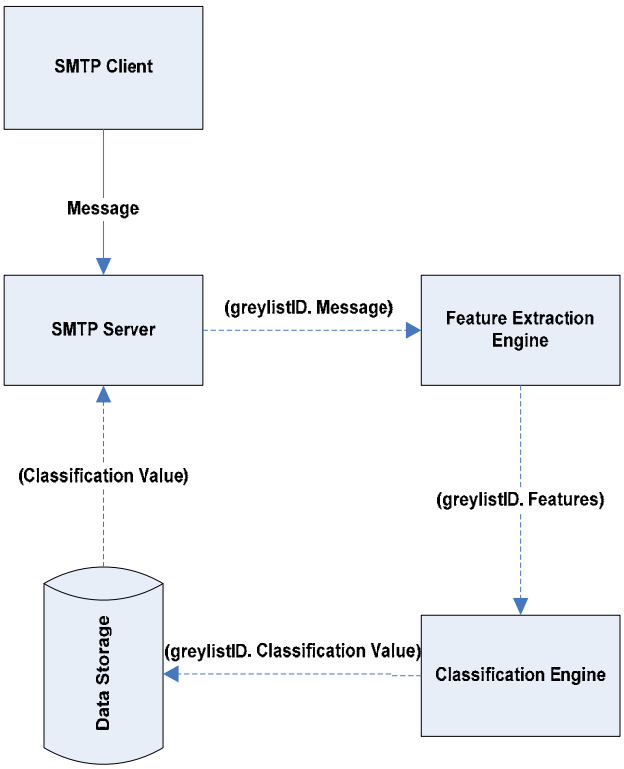

Figure 16. Architectural Overview

Our architecture, depicted in Figure 16, comprises three main components: (*i*) an SMTP server providing enhanced variants of greylisting [35], (*ii*) a feature extraction engine, and (*iii*) a classification engine. Any incoming new e-mail first passes through the SMTP server (and is potentially greylisted, that is, temporarily rejected. After it has passed the greylisting component or while it is greylisted, the data goes through the feature extraction engine. Here, several features of the message are determined, for example, the result of a Bayesian classifier, the distributed checksum clearinghouse (DCC, [36]) counter of the message and whether it is listed in a real time blacklist (RBL) or not. The extracted features are the input for the classification engine, which assigns the message to the spam or ham category, potentially with a certain “spam value”. The classification and the spam value are stored together with an identifier for the message. If the same message is received a second time (for example, as a reaction to the temporary rejection by the greylisting component), the previously computed classification is available immediately.

In contrast to existing purely text-based approaches (see, for example, [37]), we currently use the e-mail features determined by SpamAssassin in the feature extraction engine. Improvements of feature selection and feature extraction strategies are currently work in progress.

The main advantages of our architecture are the following:

- All components can be implemented as separate processes which improves scalability. Hence, it is possible to deploy an arbitrary number of SMTP servers, feature extraction engines and classification engines on different machines to balance the load of the incoming mail traffic and to avoid performance bottlenecks.

- The processes for feature extraction and classification can utilize the time available between two delivery attempts in the greylisting process. This makes it possible to classify each message before actually accepting it (and taking full responsibility for it) and reduces the number of spams reaching the scope of responsibility of the receiving mail server. Moreover, since potentially resource-intensive mining and filtering techniques are performed without open SMTP-connections, this allows for strategies to protect from denial of service attacks.

The two core components of our framework are described in [38].

5.1. Performance Metrics

We evaluated the greylisting component and the classification engine experimentally. Two different types of experiments were performed independently, a test on a live spam stream for evaluating the greylisting component and an offline test for a comprehensive in-depth evaluation of our classification engine. Our test infrastructure comprised several software and hardware components, primarily a message store containing the spam and ham messages for the offline test and three server machines each one deploying one module of our three-component architecture. Table 9 summarizes the results of the greylisting component.

Greylisted	Greylisted due to invalid timestamp	Rejected due Classification	Accepted
833 [90.45%]	85 [9.23%]	2 [0.22%]	1 [0.11%]

Table 9. Greylisting Results

Figure 17 compares the rates for true and false positives for SpamAssassin and the classification engine of our prototype with various thresholds.

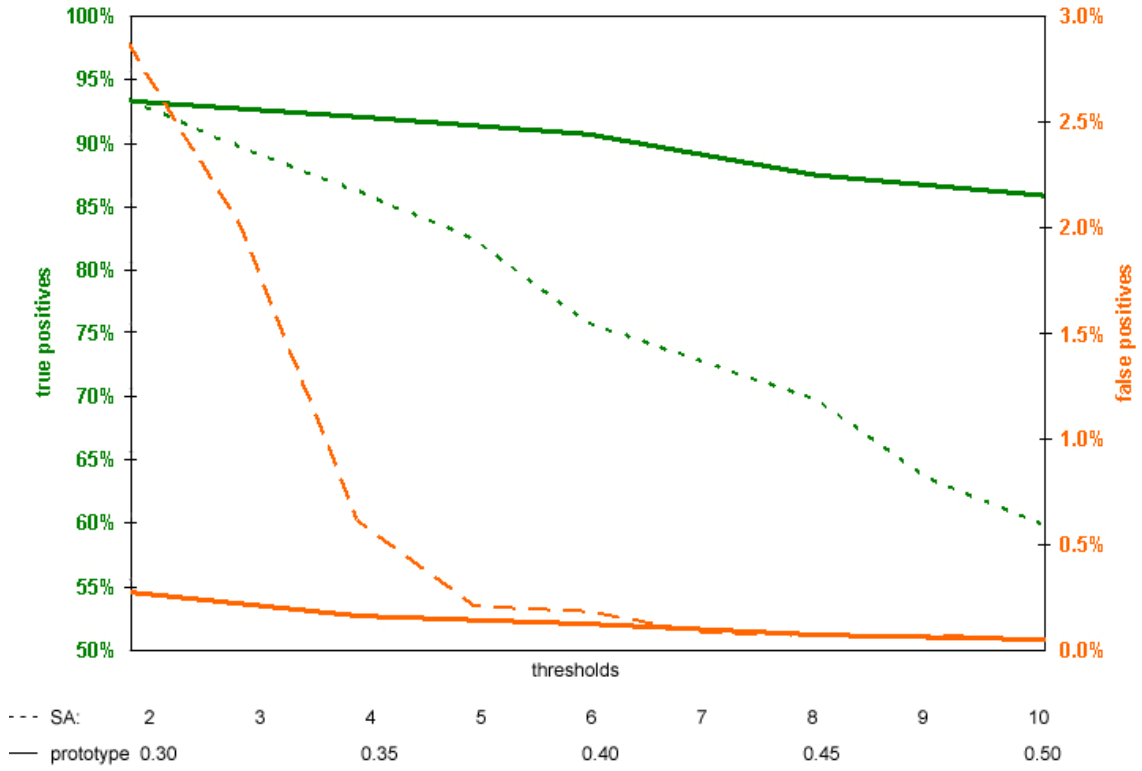

Figure 17. True Positives and False Positives for SpamAssassin and the classification engine of our prototype

5.2. Conclusion

In this project, we developed a prototype for filtering (detecting and repelling) unsolicited commercial or bulk e-mail. It comprises three major components, a greylisting component, a feature extraction component, and a classification engine. The latter one forms the central core of our framework. We have implemented and experimentally evaluated two components of our framework. We achieved very satisfactory performance and could show significantly improved results of the classification engine over the de-facto standard SpamAssassin for high ham recall rates, which are crucial in practice.

6. Appendix

Benutzerhandbuch

Voraussetzungen zum Betrieb und Installation des Prototyps:

1. Installation des Java SDK 5.0 (http://java.sun.com/j2se/1.5.0/download.jsp)
2. Installation der Berkeley DB Java Edition (http://www.sleepycat.com/products/bdbje.html)
3. Installation von SpamAssassin (http://www.spamassassin.org)
4. Installation von ClamAV (http://www.clamav.net/). ClamAV ist ein Open Source Virenscanner. Jede eingehende Nachricht wird auf Viren überprüft. Das Ergebnis wird in der Datei „classificatin.log“ mitgeloggt.

Installation und Start des Prototyps:

1 Installation:
1.1 Entpacken von PrototypSpamabwehr.zip
1.2 Wechseln in .\James
1.3 Ausführen von .\James\build.bat (build.sh)

2 Start des Email-Servers mit Greylisting-Komponente:
2.1 Wechseln in .\James\dist\james-2.2.0\bin\
2.2 Ausführen von .\James\dist\james-2.2.0\bin\run.bat (run.sh)

3 Start des ClassifictionResultManagers:
3.1 Wechseln in .\James\dist\james-2.2.0\bin\
3.2 Ausführen von .\James\dist\james-2.2.0\bin\run_clmanager.bat (run_clmanager.sh)

4 Start der ClassificationEngine:
4.1 Wechseln in .\ClEngine
4.2 Ausführen von .\ClEngine\run_clengine.bat (run_clengine.sh)

Vergewissern Sie sich, dass der JAVA CLASSPATH gesetzt ist und alle Bibliotheken im Verzeichnis .\James\lib darin verfügbar sind. Für den Betrieb des Prototyps unter Linux muss die Datei db.jar durch die Linux-Version ersetzt werden.

Konfiguration des Prototyps:

Die Konfiguration des Prototyps findet in mehreren .config Dateien statt. Diese befinden sich im Verzeichnis .\James\dist\james-2.2.0\bin\ bzw. im Verzeichnis .\ClEngine. Generelle

Einstellungen des JAMES SMTP-Servers können in .\James\dist\james-2.2.0\apps\james\SAR-INF\config.xml vorgenommen werden. Für Details diesbezüglich siehe http://james.apache.org/.

Konfiguration der Greylisting-Komponente:

Im Folgenden werden die einzelnen Variablen zur Konfiguration der Greylisting-Komponente erläutert. Diese Parameter können in der Datei „greylisting.config“ verändert werden.

GREYLIST_DELAY:
Der GREYLIST_DELAY spezifiziert die minimale Zeitperiode in Sekunden, die zwischen dem ersten und einem potentiellen zweiten Zustellversuch einer Nachricht vergehen müssen, um diese Nachricht nicht zu greylisten. (z.B.: GREYLIST_DELAY=60)

EXPIRATION_QT:
EXPIRATION_QT spezifiziert die Gültigkeitsdauer eine Quartupels in Sekunden und gleichzeitig die maximale Zeitperiode, die zwischen dem ersten und einem potentiellen zweiten Zustellversuch einer Nachricht vergehen darf, um diese Nachricht nicht zu greylisten. (z.B.: EXPIRATION_QT=1209600)

EXPIRATION_REL:
Unser Prototyp implementiert dynamisches Whitelisting für IP-Adressen (IP-Ranges). EXPIRATION_REL spezifiziert die Gültigkeitsdauer eines Eintrags (einer IP-Adresse/Range) auf der Whitelist in Sekunden. Nach Ablauf von EXPIRATION_REL wird der Eintrag automatisch gelöscht. (z.B.: EXPIRATION_REL=604800)

RELIABILITY_THRESHOLD:
Bevor ein Sender „whitelisted“ wird, muss er seine Vertrauenswürdigkeit durch die zuverlässige Übertragung von Nachrichten verifizieren. Diese Nachrichten müssen von der Classification Engine als Ham klassifiziert worden sein. Der Wert von RELIABILITY_THRESHOLD spezifiziert hierbei die Anzahl der Nachrichten, die notwendig für ein Whitelisting sind. Um Wenn Greylisting deaktiviert werden soll, ist diese Variable auf *-1* zu setzen. (z.B.: RELIABILITY_THRESHOLD=5)

RELIABILITY_START:
RELIABILITY_START spezifiziert den Startwert des Reliability-Counters für jeden unbekannten Sender. Wenn Whitelisting deaktivieren werden soll, ist diese Variable auf *-1* zu setzten. Gleichzeitig muss der Wert von RELIABILITY_THRESHOLD größer als *-1* sein. (z.B.: RELIABILITY_START=-1)

EXPIRATION_SUSP:
Unser Prototyp implementiert dynamisches Blacklisting für IP-Adressen (IP-Ranges). EXPIRATION_SUSP spezifiziert die Gültigkeitsdauer eines Eintrags (einer IP-Adresse/Range) auf der Blacklist in Sekunden. Nach Ablauf von EXPIRATION_SUSP wird der Eintrag automatisch gelöscht. (z.B.: EXPIRATION_SUSP=259200)

SUSPICIOUS_THRESHOLD:
Bevor ein Sender „blacklisted“ wird, muss er eine bestimmte Anzahl von Spam Nachrichten zuverlässig übertragen. Der Wert von SUSPICIOUS_THRESHOLD spezifiziert hierbei die

Anzahl der Nachrichten, die notwendig für ein Blacklisting sind. (z.B.: SUSPICIOUS_THRESHOLD=5)

SUSPICIOUS_START:
SUSPICIOUS_START spezifiziert den Startwert des Suspicious-Counters für jeden unbekannten Sender. Wenn Blacklisting deaktiviert werden soll, ist diese Variable auf *-1* zu setzen. Gleichzeitig muss der Wert von SUSPICIOUS_THRESHOLD größer als *-1* sein. (z.B.: SUSPICIOUS _START=1)

SPAM_THRESHOLD:
Spezifiziert den Spam-Schwellwert, ab dem eine Nachricht als Spam klassifiziert wird. (z.B.: SPAM_THRESHOLD=0.35) Wird der Wert erhöht, so sinken sowohl die false-positive Rate als auch die Spamerkennungsrate.

GREYLIST_LEVEL:
GREYLIST_LEVEL spezifiziert die Anzahl der Bytes der Sender IP-Adresse von links, die für die Generierung der Sender-Identifikation herangezogen werden. Diese wird in der Black- und Whitelist verwendet, wie auch zur Generierung der Greylist-Id. Der GREYLIST_LEVEL muss größer oder gleich dem Wert *1* und kleiner oder gleich dem Wert *4* sein. (z.B.: GREYLIST_LEVEL=3 --> letztes Byte der IP-Adresse wird nicht für die Generierung verwendet, d.h. aus 135.122.108.128 wird 135.122.108)

Konfiguration von Feature Extraction und Classification Komponente:

Die Konfiguration der anderen Komponenten besteht lediglich aus der Spezifikation der Verbindungseinstellungen (IP-Adresse und Port). Details entnehmen Sie bitte den entsprechenden .config Dateien.

Training des Prototyps:

Aufgrund der prototypischen Entwicklung des Spam-Filters ist bis jetzt noch kein automatisierter Trainingsprozess verfügbar. Um zu trainieren, müssen zuerst zwei IMAP-Folder mit Trainingsemails befüllt werden, wobei in einem Ordner nur nicht-SPAM (HAM) Emails liegen, in dem anderen nur SPAM Emails. Diese Emails werden nun zu SpamAssassin geschickt und von diesem bewertet, die Ergebnisse werden in zwei .dat Files abgespeichert. Diese Files haben die Namen HamTrainingsSample.dat bzw. SpamTrainingsSample.dat und liegen im Verzeichnis .\ClEngine\Files\Samples. Sind diese Dateien erstellt, so kann die ClassificationEngine mittels „train matrix“ trainiert werden, wobei die Informationen aus den eben genannten Files gelesen werden. Diese Art des Trainings ist natürlich nur ein rudimentäres Verfahren für den Prototyp und wird in weiterer Folge durch adaptive Trainingsmethoden erweitert. Diese werden ein automatisiertes Training sowie eine Erweiterung der Trainingssets während des Betriebs des Spam-Filters ermöglichen, d.h. es wird möglich sein, eindeutige (bzw. gut diskriminierende) Emails im laufenden Betrieb des Programms zu einem der beiden Trainingssamples (HAM, SPAM) hinzuzufügen.

7. Bibliography

[1] A. Amor, J. Martin: “Civic Networking: The Next Generation”, 1998.
http://www.more.net

[2] REDNET, “Networking and Internet”, British ISP since 1992.
http://www.red.net/support/resourcecentre/mail/email-aup.php

[3] P. Hofmann: “Unsolicited Bulk E-mail: Definitions and Problems“, October 5, 1997.
http://www.imc.org/ube-def.html

[4] Mitteilung der Kommission an das europäische Parlament über unerbetene Werbenachrichten, 22.01.2004.
http://europa.eu.int/information_society/topics/ecomm/doc/useful_information/library/communic_reports/spam/spam_com_2004_28_de.pdf

[5] ORF online, August 24, 2004.
http://futurezone.orf.at/futurezone.orf?read=detail&id=245906

[6] Standard online Report, December 30, 2004.
http://derstandard.at/?url=/?id=1857456

[7] Anti-Spam Software Vendor Brightmail, Spam percentage June 2004.
http://www.brightmail.com

[8] Anti-Spam Software Vendor MessageLabs, Spam Statistic November 2006.
http://www.messagelabs.com/publishedcontent/publish/threat_watch_dotcom_de/threat_statistics/spam_intercepts/DA_136867.chp.html

[9] The Spamhaus Project, List of the 200 biggest spammers called ROKSO.
http://www.spamhaus.org/rokso

[10] Anti-Spam and Virus Software Vendor Sophos, Dirty Dozen, the 12 most spamming countries.
http://www.sophos.com

[11] Sophos Security Threat Management Report 2005.
http://www.sophos.com/virusinfo/whitepapers/SophosSecurity2005-mmuk

[12] Sophos News Archive, April 20 2006.
http://www.sophos.com/pressoffice/news/articles/2006/04/dirtydozapr06.html

[13] Anti-Spam Software Vendor Commtouch.
http://www.commtouch.com

[14] Anti-Spam and Virus Software Vendor Postini.
http://www.postini.com

[15] Ferris Research.
http://www.ferris.com

[16] Heise News: “Spam belastet Europas Unternehmen”, January 4, 2003.
http://www.heise.de/newsticker/meldung/33417

[17] M. Gibbs: “Spam Cost Model”, NetworkWorldFusion.
http://www.gibbs.com/msg

[18] Ikarus Software (Spam Wall and Anti-Virus).
http://www.mymailwall.at/spamcal.html

[19] M. Clement, H.-J. Boie, „Die Kosten von Spam“, 2006.

[20] OECD Report: “Spam Issues in Developing Countries“, 2005.

[21] W. Gansterer, M. Ilger, P. Lechner, R. Neumayer, J. Strauss, „Phase 2 and 3 of Project Spamabwehr – SMTP based concepts and cost-profit models“, Technical Report FA384018-2, Institute for Distributed and Multimedia Systems, University of Vienna, 2005

[22] W. Gansterer, H. Hlavacs, M. Ilger, P. Lechner, J. Strauss, „Token Buckets for Outgoing Spam Prevention“, Proceedings of the IASTED International Conference on Communication, Network and Information Security, 2005

[23] M. Daberger, „Quantitative Untersuchungen der Eigenschaften von Unsolicited Bulk E-Mail und Unsolicited Commercial E-Mail”, Master’s thesis, Institute for Distributed and Multimedia Systems, University of Vienna, 2006, in preparation.

[24] W. Gansterer, M. Ilger, P. Lechner, R. Neumayer, J. Strauss, „Anti-spam methods – state-of-the art”, Technical Report FA384018-1, Institute for Distributed and Multimedia Systems, University of Vienna, 2005.

[25] P. Graham, „Stopping spam“, 2003
http://www.paulgraham.com/stopspam.html

[26] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, & C. D. Spyropoulos, “An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages”, SIGIR ’00: Proc. 23rd annual international ACM SIGIR conference on research and development in information retrieval, Athens, Greece, 2000, 160–167.

[27] The Apache SpamAssassin Project.
http://spamassassin.apache.org

[28] W. Weinman, Authenticated mail transfer protocol, 2003.
http://amtp.bw.org

[29] D. J. Bernstein, Internet mail 2000.
http://www.im2000.org

[30] The Mozilla Foundation: Mozilla Thunderbird open-source spam filter. Internet, open source (2005).
http://www.mozilla.org

[31] W. Gansterer, M. Ilger, A. Janecek, P. Lechner, J. Strauss, Phase 1 of Project “Spamabwehr II”, Technical Report FA384018-3, Institute for Distributed and Multimedia Systems, University of Vienna, 2005.

[32] W. Gansterer, M. Ilger, J. Strauss, “The Economy of Spam”, in preparation.

[33] R. Clayton, “Stopping outgoing spam by examining incoming server logs”, 2nd Conf. on Email and Anti-Spam (CEAS), Stanford, USA, 2005.

[34] W. Venema, Postfix mail transfer agent, 2005.
http://www.postfix.org

[35] E. Harris, “The Next Step in the Spam Control War: Greylisting”. Technical Report, 2005.
http://projects.puremagic.com/greylisting/whitepaper.html

[36] Distributed Checksum Clearinghouse, 2006.
http://www.rhyolite.com/anti-spam/dcc

[37] Gee, K. “Using latent semantic indexing to filter spam”, In Proceedings of the 2003 ACM Symposium on Applied Computing (SAC), 2003, ACM, 460-464.

[38] W. Gansterer, A. Janecek, P. Lechner, “Vector Space Models for E-mail Mining and Spam Filtering”, submitted.