SPAM FILTER ANALYSIS*

Flavio D. Garcia
Department of Computer Science, University of Nijmegen, the Netherlands
flaviog@cs.kun.nl

Jaap-Henk Hoepman
Department of Computer Science, University of Nijmegen, the Netherlands
jhh@cs.kun.nl

Jeroen van Nieuwenhuizen
Department of Computer Science, University of Twente, the Netherlands

Abstract Unsolicited bulk email (aka. *spam*) is a major problem on the Internet. To counter spam, several techniques, ranging from spam filters to mail protocol extensions like *hashcash*, have been proposed. In this paper we investigate the effectiveness of several spam filtering techniques and technologies. Our analysis was performed by simulating email traffic under different conditions. We show that genetic algorithm based spam filters perform best at server level and naïve Bayesian filters are the most appropriate for filtering at user level.

Keywords: Spam, unsolicited email, email abuse, security, networking, Bayesian filtering, genetic algorithms, text classification.

1. Introduction

Spam [16], officially called unsolicited bulk email (UBE) or unsolicited commercial email (UCE), is rapidly becoming a major problem on the Internet. At the end of 2002, as much as 40% of all email traffic consisted of spam[1,2], and recent reports estimate that this amount has risen to more than 50%[3].

To handle this increasing load of junk email [17], several spam filtering techniques exist to automatically classify incoming email as spam, and to re-

*Id: spam-filter.tex,v 1.31 2004/04/28 09:57:35 flaviog Exp
[1] `http://zdnet.com.com/2100-1106-955842.html`
[2] `http://www.linuxsecurity.com/articles/privacy_article-6369.html`
[3] `http://zdnet.com.com/2100-1105_2-1019528.html`

ject or discard email classified as such [20]. In this paper we investigate the effectiveness of these spam filtering techniques and technologies.

For users, receiving spam is quite a nuisance and costs money. In a recent study of the European Community [4], it was estimated that the cost for receiving spam for an average Internet user is in the order of 30 euro a year. But the costs of spam goes well beyond the total costs of all recipients. Each ISP pays for each email message received, because it must be stored in a mail box and it takes up a certain amount of bandwidth. The total cost has been estimated in the order of 10 billion euro a year [10].

A second problem with spam is the impact it has on the Internet backbone. Spam sent over the Internet backbone causes delays for all Internet users. Furthermore, because most spammers use mailing lists that have outdated addresses on them, many messages are rejected ("bounced"). This mandates the operator of the intended destination to send a return response, wasting even more bandwidth [13].

Bulk mailers use several different techniques to send their spam. Often, bulk mailers misuse the SMTP protocol or use badly configured MTAs (so-called open-relays) to hide their tracks [10][15]. We describe these techniques in detail in section 1.2.1.

There are at least three fundamentally different ways to counter spammers [5].First, bulk mailers can be prevented to send spam by blocking or limiting access to mail servers. Another method is make spamming less profitable, for example by incurring a cost on every email message sent [7]. A third method aims to detect and remove all spam once it is sent by applying different types of filtering techniques that use the special characteristics of spam to recognise it [2][3][9][12][19]. These techniques are discussed in section 1.2.2.

Our analysis of countermeasures against spam focuses on filtering techniques. We are interested in measuring the accuracy level of these filters in practice. Some of the filtering techniques not only look at the content of each message, but also consider the email traffic at large (e.g., methods that try to detect duplicate mail messages, or checksum schemes that match incoming messages with a database of known spam messages). To faithfully analyse such spam filters, we built a simulator to generate realistic email traffic and test the filters with it. In section 1.3 we give a description of our analysis method. We have analysed the performance of the filters in two settings: while being used at the server level and while being used by the end user directly. While running at server level, the filter might use the information about connections from the server. On the other hand, while running at user level, the filter is able to be trained or customised as a function of user specific characteristics. Moreover,

[4] `http://europa.eu.int/comm/internal_market/privacy/docs/studies/spamstudy_en.pdf`

we have measured the different behaviour of filters depending on the type of bulk mailer used to generate the spam. We refer to section 1.3 for details.

To summarise our results: we have found that filters based on genetic algorithms, perform best at ISP level and naïve Bayesian filters perform best at user level. We discuss the results of our analysis on a per filter basis in section 1.4.

2. Spam: producers and countermeasures

In this section we describe the most common techniques used by bulk mailers. We also describe current proposals for countering spam, with focus on filtering techniques.

2.1 Bulk mailing techniques

Spammers use so-called bulk mailers to send spam. These bulk mailers are capable of sending huge volumes of email without going through a specific mail server or a particular ISP. Some bulk mailers are capable of sending approximately 250,000 messages an hour over a 28.8kb/s modem line [10]. This enormous amount of messages is attained by contacting more than one mail server at the same time and misusing the resources of the ISPs.

The bulk mailers used by spammers have several features to hide their tracks. Most bulk mailers do not use the mail server of their ISP, but instead connect to the destination mail server directly or use a so-called open relay. This way, the spammer avoids to be detected by his ISP. An open relay is a SMTP or ESMTP server that allows everyone to use that server to relay mail. To make the tracking even harder when an open relay is used, most bulk mailers add so-called bogus received headers to the spam message (in front of the *real* received headers added by the SMTP protocol). By adding these bogus headers they hope to redirect any tracking to a site in the fake header.

Bulk mailers also include features which try to outsmart spam filters. The most processor consuming feature is to personalise every message for a recipient. This personalisation of messages can be classified into two types. In the first type, the spammer only uses the victim's mail address as the recipients address instead of using the `Bcc:` headers to send the message to recipients. In the second type, he also uses the name or mail address of the victim to personalise the body of the message. Less processor consuming techniques include the randomisation of the `Subject:` field and the `From:` address line. Some bulk mailers also forge the Message-ID and/or do not send the `To:` header in the SMTP session. Another technique especially developed to confuse Bayesian filters, is to add extraneous text in the body of a message. This text is usually a set of randomly selected words from a dictionary, or just some paragraphs from news or books.

2.2 Countermeasures

There are two fundamentally different methods to counter spam. The first method tries to prevent bulk mailers to send spam, e.g., by incurring a cost on every email message sent, or by blocking or limiting access to mail servers for spammers. The second method aims to detect and remove all spam once it is sent, by applying different types of filtering techniques.

2.2.1 Spam prevention. The most direct way to prevent spam is to close all open relays on the Internet, and to strengthen the SMTP protocol to disallow bogus received headers and require sender authentication, to facilitate bulk mailer tracking. This forces bulk mailers to send spam through their own ISP, but relies on these ISPs to block their accounts. More fundamentally, this approach goes against the open philosophy of the Internet and poses yet another threat to privacy on the Internet. Moreover closing all open relays would not be sufficient, as spammers are now shifting to the use of open proxies to hide their tracks[5], or even use hacked computers.

One interesting and perhaps more feasible approach proposed to stop junk email is by using economic based solutions. The principal attractiveness of spam is that sending large amounts of small email messages is relatively cheap compared to other direct marketing techniques. The idea behind such economic based methods is to make the sending of email more expensive, thereby making it less attractive to send huge amounts of mail. The two main categories of economic solutions are computing time based systems (that force the spammer to spend considerable amounts of his computing resources to send a single spam message) and money based systems (that charge a small amount of money for every email sent).

Computing time based systems. In computing time based economic solution, the sender of a message is required to compute a moderately expensive function. This function is called a *pricing function* [21][4]. The idea is that for a legitimate sender it is not too expensive in computer time to send a message to a recipient, but for a spammer, who has to send many messages, it is. This expensive use of computer time makes it much harder for a spammer to send large volumes of mail within an acceptable time.

The question whether such a system will work in practice is hard to answer. First of all, this feature has to be incorporated into the Internet, and this may not be easy (although, admittedly, it does not require changes to the SMTP protocol and could simply be enforced by mail user agents). Second, there is the problem of hardware backward compatibility. A user using an old computer must be

[5] `http://zdnet.com.com/2100-1106-958847.html`

able to send an email in a reasonable amount of time. This rules out the use of too costly pricing functions. But then for a spammer using modern hardware, the cost in time to send a message may become almost equal to zero. It seems impossible to find a pricing function that suits both needs (although methods based on memory bandwidth limitations appear a little more promising [1]).

Money based systems. Money based systems are based on channelised email systems where users require payment before reading message arriving on certain channels [21]. This payment can be in the form of electronic cash to automate the process. This approach would make it more costly to send Junk-E-mail, which makes it less attractive.

Problems with money based systems include (among others) the adoption of the system by users, and the absence of a global electronic cash system. It is therefore hard, if not impossible, in practice to introduce a money based spam prevention system.

2.2.2 Spam Filters. Filter based countermeasures against spam can be divided into two main categories: cooperative filtering and heuristic filtering. The main principle of the former is that there can be cooperation between spam originators and spam recipients. Such a cooperative filtering system requires a network-wide implementation of, and adherence to, a set of standards for identifying spam. Because most spammers try to hide to track such an implementation is not likely to appear, hence we will focus on heuristic filtering.

Heuristic filters work on the assumption that it is possible to distinguish between normal mail and spam by applying heuristic rules to a message. We can distinguish three types of heuristic rules: origin based filtering, filtering based on traffic analysis and content-based filtering.

Origin based filtering. Origin based filtering happens before a message is fully received by the computer of the recipient. The most prominent method in this class uses the so-called blacklists (e.g., the blacklist from ordb.org[6]). These blacklist can be used to refuse IP or TCP connections from spam originators, but also to refuse mail if the domain name given at the `FROM:` command is on the blacklist. This method can be circumvented by relaying mail through the SMTP servers of legitimate originators that are not on the blacklist. Another disadvantage is that this sites are frequently under denial of service attacks (e.g., the blacklist of osirusoft.com).

A second approach is to configure the SMTP server to perform a reverse DNS lookup to find the IP-address associated with the domain name given at the `MAIL FROM` command. If this IP-address is not equal to the IP-address of

[6] `http://www.ordb.org/`

the TCP-connection, the SMTP server can refuse to handle the message. This method can also be circumvented by using relay hosts.

Whitelists are another origin based filter method. They contain the senders (or domains) from which incoming mail is automatically accepted for delivery. All other mail is refused by default. To enable legitimate senders to reach the recipient, a whitelist based system will return a request for confirmation to the sender, who should reply to this message within a short period of time. When a whitelist is used it is almost certain that no spam will reach the inbox of the user. The disadvantage, however, is that a whitelist has a very high false positive rate, which may seriously confuse or irritate unknown but legitimate senders. Moreover, bulk mailers increasingly match the from header in their mails to the domain of the recipient, using known (and probably trusted) senders in the same domain.

Filtering based on traffic analysis. Traffic analysis based filtering can be used at the mail server of the ISP. Here the log files of the SMTP server can be used to detect anomalies in the normal traffic stream. Anomalies that can appear and indicate the spam are anomalies in the connection time to the server, anomalies in the amount of mail coming from a certain host, the fact that a message is send to more recipients as normal from a certain host or the fact that mail is relayed.

Content-based filtering. Content-based filtering happens after a message is fully received (including the body of the message). In this case, filtering can also be based on known keywords in the subject and body of the message, common features of spam and the use of signature/checksums from databases on the internet.

Naïve Bayesian filtering is a new content-based mechanism for spam filtering [19][2][11]. Before it can be used, a Bayesian filter must be trained with a set of spam and a set of legitimate emails (aka. *ham*) that have been previously classified. For each word w in the training sets the filter estimates the probability that it occurs in a spam message ($C = \mathbf{S}$) or in a ham message ($C = \mathbf{H}$) using

$$P(W = w|C = \mathbf{S}) = \frac{S(w)/N_S}{S(w)/N_S + H(w)/N_H},$$

where $S(x)$ is the number of occurrences of word x in the spam set, $H(x)$ is the number of occurrences of word x in the ham set, and N_S and N_H are the sizes of the spam and ham training sets respectively.

When a new message $M = \{w_1, ..., w_N\}$ arrives, the filter determines the n most *interesting* words $\{\dot{w}_1 \dots \dot{w}_n\} \subseteq M$, where interesting means $P(\dot{w}_i) \approx 1$ or $P(\dot{w}_i) \approx 0$). Using these 'interesting' words, the filter then computes the probability that M is spam using Bayes rule [8][3]

$$P(C = \mathbf{S}|\overrightarrow{W} = M) = \frac{P(C = \mathbf{S}) \prod_i P(W = \dot{w}_i | C = \mathbf{S})}{\sum_{k \in \{\mathbf{S},\mathbf{H}\}} P(C = k) \prod_i P(W = \dot{w}_i | C = k)}$$

Then if this probability is greater than a given threshold T, M is classified as spam.

A third method to filter on content is the use of *genetic algorithms*[7]. Genetic algorithms use so-called feature detectors to score an email message. In practice, these feature detectors are a set of empirical rules that apply to the message and return a numeric value. A genetic program is then represented as trees and has associated a training set and a fitness function. The evolutionary mechanism is accomplished by two basic operations, namely "crossover" and "mutation". This process intends to find a minimum in the fitness function. This score can then be used to classify a message as spam or ham. A more detailed explanation of spam filtering by genetic programming can be found in [14].

Another approach which has the ability to learn is the use of *neural networks*. Like Bayesian filters, neural networks must be trained first on a set of spam and non-spam messages. After this training the neural network can be used to classify incoming mail message based on common features in email messages [6].

It is also possible to classify mail based on the content by the use of *signature/checksum schemes*[8] in a cooperative system. When a mail message arrives a signature/checksum is calculated for this message and compared to the values in special spam databases on the Internet. If the checksum matches any of the values in the database, the message is regarded as spam.

3. Method of analysis

In this section we detail the method of our analysis. Our goal is to measure the effectiveness of several spam-filtering techniques on realistic email traffic patterns, both at ISP and at the user level. This traffic should reflect the fact that different users have different traffic patterns and their emails also differ in their contents. In an ideal situation we would perform these measurements on real email traffic, but due to privacy reasons, that is infeasible (unless we restrict attention to a very small and atypical consenting sample like staff at a university). Instead we perform this analysis by generating both normal and spam email traffic using a simulator, and measuring the accuracy level of each analysed spam filter.

[7] `http://spamassassin.org/`

[8] `http://www.rhyolite.com/anti-spam/dcc/`

3.1 Mechanism of the analysis

Our analysis aims to measure how good a spam filter is at preventing spam from being delivered to the end user, while still allowing legitimate emails to pass through unblocked. To this end, we compute for each spam filter, the false acceptance and rejection rate. Moreover, we propose a single measure of the filter's wrongness as a function of its false acceptance and rejection rates. Using this measure, we can rank the accuracy of the evaluated filters. We also study how the spam personalisation and new techniques like the inclusion of random words affects this performance.

For each message, a spam filter does one of four things:

$S \to S$: it correctly classifies a spam message as spam,

$S \to H$: it falsely accepts a spam message as ham,

$H \to H$: it correctly classifies a ham message as legitimate, or

$H \to S$: it falsely rejects a ham messages as spam.

When running the spam filter on n messages, n_S of which are spam messages and the remaining n_H are legitimate messages, we write $n_{S\to S}$ for the number of correctly classified spam messages, and define $n_{S\to H}$, $n_{H\to S}$ and $n_{H\to H}$ analogously. Then the false acceptance rate (FAR) and the false rejection rate (FRR) are defined as

$$FAR = \frac{n_{S\to H}}{n_S} \qquad FRR = \frac{n_{H\to S}}{n_H} \ .$$

Clearly, the false acceptance rate can be artificially decreased to 0 by blocking all messages. This increases the false rejection rate though. The same happens with the false rejection rate: it can be decreased to 0 by not blocking any messages at all. A good spam filter has a low FAR as well as a low FRR. We want to be able to rank the performance of spam filters, as expressed by their accept and reject ratios, using a single scalar value. At first, it would seem natural to define the *wrongness* W of a spam filter as the distance to the origin when plotting the performance of the filter in the FAR/FRR plane. However we do not consider these errors to be symmetric, given that it is much worse to falsely reject a legitimate message than to accept a spam message. We can think of a false positive as an error, and a false negative as an effectiveness indicator. Another way to approach this, is to say that one is willing to tolerate a small increase in the false reject ratio for a significant reduction of the false accept ratio. Trying to represent that we propose the function

$$W(FAR, FRR) = (FRR + \epsilon)^2(FAR + \epsilon) \ , \qquad (1)$$

where ϵ is a small constant (i.e., $\epsilon = .01$).

3.2 Modelling of the normal email traffic

The simulator needs to generate email traffic that faithfully emulates the behaviour of a normal user, to test the filters with. In order to do that we need content for the bodies of these emails, and an effective method to generate normal email traffic patterns.

Ideally, the bodies of the messages should come from real mail sent by different kinds of users. However, due to privacy considerations, this is not possible. Instead, we use bodies from a wide variety of USENET (news) messages. This variety is necessary to avoid creating email messages with a limited set of subjects and vocabulary. We do note however that USENET messages do not contain HTML code (whereas some of the spam messages considered (see section 1.3.3) do).

One important issue to model is the fact that the contents of the messages sent by different people varies. A person's vocabulary varies as a function of his occupation, education, hobbies, etc. In order to reflect that in our model, each sender of an email belongs to a specific topic group chosen randomly. Each topic group only takes bodies from a specific news group.

Another important issue to model is that in normal email traffic, people mostly get mail from people they know. To model this behaviour we enumerate senders and receivers and use a normal distribution to find people close to the sender of the message. The use of the normal distribution means that most mail will be send to people close to the sender in the list, but that there is still a chance of sending mail to people less close to the sender.

Mailing lists also are considered as a special case, because in some aspects they behaves like spammers, sending multiple copies of the same content to many different users, so this behaviour may confuse some filters. In our model, each mailing list has a email address database which is initialised at the beginning of the simulation. When it sends an email, it iterates over its database sending the same message to all the users in it. When the end of the database is reached, another message is chosen and the process starts again.

3.3 Modelling of the spam traffic

To generate realistic spam email-traffic, we have analysed the behaviour of bulk-mailers. An important characteristic of spam is that it arrives in bulk. Whenever a bulk mailer start sending, it keeps sending for some period of time, until the spam message is delivered to all the users in it's database.

An issue when generating the spam traffic is to make sure that the percentage of spam of the total email is realistic. At the end of 2002, 40%[9,10] of the total

[9] `http://zdnet.com.com/2100-1106-955842.html`
[10] `http://www.linuxsecurity.com/articles/privacy_article-6369.html`

email traffic consist of spam, and is rapidly growing, accounting for more than 50 percent on June, 2003[11].

In our model of spammers, each spammer has a database with email addresses. When a spammer starts sending, it continues sending as fast as possible, until the message is delivered to the whole database. A spammer can send personalised spam. In this case, each spam message has only one target address, say login@server, and the line “Dear login,” is added to the body of the message.

To generate spam mail we also need content for the bodies of the spam messages. We use a variety of spam archives to extract these bodies[12]. In a separate simulation, we also include a collection of very recent spam messages in order to show the impact of the spam evolution in the filtering techniques.

3.4 The simulator

The simulator must generate authentic-looking, email traffic and bulk mail patterns. We also believe that the simulator should be as general as possible. That is, it should be easy to use with existing filters and with filters that might be invented in the future. In addition it should also be easy to add new features of bulk-mailers.

The architecture chosen to fulfil these requirements is to split the simulator into four parts (see Fig. 1). The first part is the traffic generator. This is the main

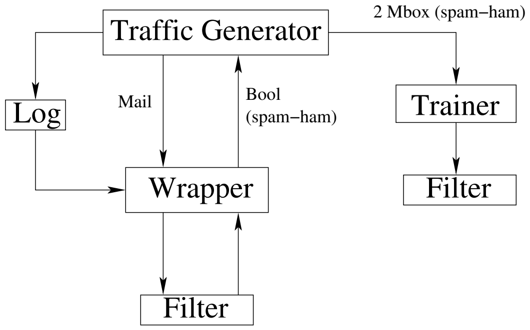

Figure 1. The design of the simulator

module, responsible for generating email traffic according to its configuration and deliver it. The traffic generator has a set of senders and receivers. Each sender has a probability to send an email every one step of the simulation and the traffic generator iterates over the set of senders in a round-robin fashion. After

[11] `http://news.zdnet.co.uk/internet/ecommerce/0,39020372,39118223,00.htm`
[12] `http://www.spamarchive.org/`

the sender is selected, if it is a normal user, the traffic generator first generates the number of targets this email will have, in which field (i.e., `To:`, `Cc:`, `Bcc:`) and then selects the receivers. In order to select the receivers, we first take the index of the sender (in the list of senders) and then we generate a offset using the normal distribution. The index plus the offset yields the receiver. In other words, the smaller the difference between your index and the index somebody else, the closer that person is to you, and the more likely it is that you exchange messages. In the case of a spammer, it changes to state sending, and it will keep sending until the spam message is delivered to all the users in it's database. In case of a mailing list, it behaves like a normal user except for the fact that their targets are selected from its database and the same body is sent to the whole database.

The second part is the wrapper, a program that receives an email on its standard input and is responsible for running the filter, take its classification (spam or ham) and send the response back to the traffic generator. If the filter is at the server level, the traffic generator will log every simulated connection to a log file, emulating the log file of the mail server. This file is accessible for the wrapper as well. Each filter may have different input formats, so the wrapper program is filter-specific.

The third part is the trainer. Given that some filters (e.g., Bayesian) need to be trained before they can be used, and the email traffic received for each user has different patterns, the simulator is able to generate a certain amount of correctly classified traffic, and send it to two different files, one for ham and one for spam. After that, the trainer is executed, and it should to be able to train the filter with this data.

3.5 The analysed filters

Some of the aforementioned filtering techniques are not included in our analysis for several reasons. Some filters are highly dependent on unknown constants (e.g., the number of spammers who are not in a blacklist). By setting those constants to an arbitrary value, we would basically be setting the results of those simulations. We also restrict our analysis to open source filters. We therefore choose to test the following set of popular spam filters: *Filtering based on traffic analysis*: Mail volume-based filter, and *Content Based Filters*: Distributed Checksum Clearinghouse (DCC), Genetic algorithm based spam filter (SpamAssassin), and Naïve Bayesian Filters (Bogofilter, Spamprove, Bmf).

4. Spam filter comparison

In this section we give for each analysed filter a short description of the results of the simulation. For each of the filters, we have measured their effectiveness both when applied at user level and at server level, against the following types

Table 1. Filter performance: non-personalised spam.

Filter	*Level*	*FRR*		*FAR*		$W * 10^5$
		mean	*conf* $*10^2$	*mean*	*conf* $*10^2$	
Bogofilter	U	.0000	.005	.144	1.750	1.56
Bmf	U	.0117	.518	.029	2.041	1.86
Mail volume	S	.0000	.000	.633	2.854	6.44
SpamAssassin	U/S	.0071	.093	.213	0.928	6.57
DCC	U/S	.0005	.034	.624	1.466	7.11
Bmf	S	.0004	.016	.656	3.817	7.20
Bogofilter	S	.0000	.010	.709	2.838	7.27
Spamprove	U	.0085	.476	.215	5.442	7.80
Spamprove	S	.0024	.239	.695	3.729	10.9

Table 2. Filter performance: personalized spam.

Filter	*Level*	*FRR*		*FAR*		$W * 10^5$
		mean	*conf* $*10^2$	*mean*	*conf* $*10^2$	
Mail volume	S	.0000	.000	.196	1.174	2.07
Bogofilter	U	.0000	.000	.233	4.819	2.43
Bmf	U	.0037	.401	.186	10.11	3.70
Spamprove	U	.0005	.033	.325	5.585	3.76
SpamAssassin	U/S	.0070	.059	.217	1.555	6.60
DCC	U/S	.0028	.071	.710	1.816	11.9

Table 3. Filter performance: non-personalised recent spam.

Filter	*Level*	*FRR*		*FAR*		$W * 10^5$
		mean	*conf* $*10^2$	*mean*	*conf* $*10^2$	
Bmf	U	.0030	.159	.070	4.630	1.37
Bogofilter	U	.0000	.006	.129	2.835	1.41
Spamprove	U	.0040	.386	.138	3.013	2.94
SpamAssassin	U/S	.0074	.080	.179	1.356	5.74
Mail volume	S	.0000	.005	.642	3.188	6.58
DCC	U/S	.0005	.034	.786	1.624	8.80

of spam traffic: (1) non-personalised spam, (2) personalised spam, and (3) non-personalised recent spam. The qualitative results (wrongness, FAR and FRR) are summarised (together with 95% confidence intervals) in several graphs and tables. The tables contain the performance of the filters both at user and at server level, indicated in the *Level* column by U and S respectively. If the

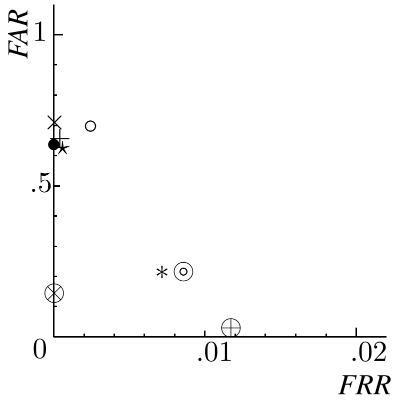

Figure 2. Non-personalised spam.

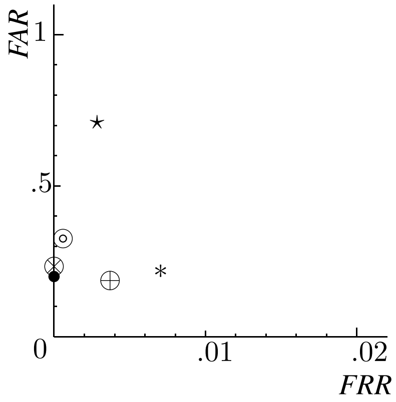

Figure 3. Personalised spam.

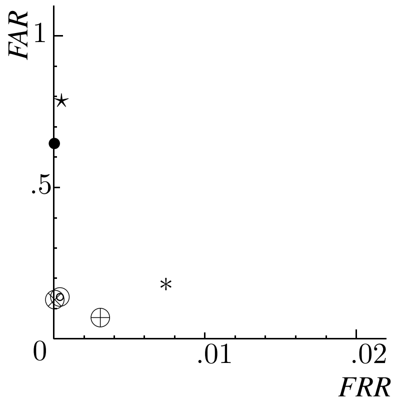

Figure 4. Non-personalised recent spam.

Filter	*Level*	*Symbol*
Bogofilter	S	×
Bogofilter	U	⊗
Bmf	S	+
Bmf	U	⊕
Spamprove	S	∘
Spamprove	U	⊚
Mail volume	S	•
SpamAssassin	U/S	∗
DCC	U/S	⋆

performance of the filter did not depend on its level, U/S is used to indicate this fact. Note that in case a Bayesian filter runs at server level, it did not get a per-user training but a general one. Table 1 contains the results for non-personalised spam. Table 2 shows the results of the simulation when personalised spam was used. Note that in this case the filters were only tested when running at user level. Finally, for Table 3 non-personalised spam was used but this time the spam mails were collected during the last two months. The purpose of this is to show the impact of the new spamming techniques (e.g., like including random words in the body of the message), especially on the Bayesian filters. The performance of the filters is also shown graphically in Fig. 2–4 (excluding the confidence intervals, because they are quite narrow).

In the remainder of this section we discuss the performance and behaviour of each of the tested filters individually, in separate subsections.

4.1 Mail volume-based filter

The volume-based filter uses an algorithm that checks how much email is received from a specific host during the last connections (the last 1500 lines from the log file in the simulation, which is the same as used by Kai's Spamshield[13]). If the amount of mail received is greater then a certain threshold, then the mail is classified as spam. This filter was able to correctly classify all the legitimate emails, for a high enough threshold. The drawback of this filter is that the FAR achieved is in general very high.

When personalisation is used, this filter performs very well. We consider this to be a side effect, because for personalised mail the bulk mailer has to establish many more connections in order to deliver the same amount of emails. That makes it easier to detect with mail volume analysis. Detection is easily avoided however using multiple open-relays at the same time.

4.2 Distributed Checksum Clearinghouse

The Distributed Checksum Clearinghouse filter tested is a modification of the standard DCC filter version 1.2.14[14]. The modification disables the report function to the Internet server. Instead, a local database was used and the filter reports the emails to this database.

The percentage of false positives of the DCC filter is small, but the performance filtering spam is small as well. This performance can be improved by a less conservative threshold, but this has a direct impact on the FRR.

When personalised spam is used the accuracy of this filter lowers slightly.

4.3 Genetic algorithm based spam filter

The genetic trained filter that we used in our simulations was SpamAssassin version 2.60[15]. The default filter configuration was used in our simulation.

This filter performs very well, achieving one of the best performances of the evaluated filters at ISP level, with the non-personalised spam. This accuracy is not largely affected when personalised spam is used. One drawback of this filter is that it is the most computationally expensive of the evaluated filters.

4.4 Naïve Bayesian Filters

Several implementations of Bayesian filters were evaluated and there were important performance differences between them. An outstanding performance was achieved by Bogofilter when it runes at user level, it had almost no false

[13] `http://spamshield.conti.nu/`
[14] `http://www.rhyolite.com/anti-spam/dcc/`
[15] `http://spamassassin.org/`

positives and more than 75% of the spam was filtered in every simulation. We suspect that this good performance is due to the Fisher's method, see Robinson [18]. Bmf has also a very good performance, comparable to Bogofilter, but less conservative. It archives lowest FAR of the evaluated filters, but the number of false positives in some circumstances is certainly high. Spamprove has a low efficacy compared with other Bayesian filters. We suspect this is related to the fact that Spamprove ignores HTML code.

A general characteristic of Bayesian filters is that their performance is lowered drastically when they run at ISP level. Another general characteristic of Bayesian filters is that most of the wrongly classified emails are very short. We suspect that in these cases there is not enough information to perform statistical analysis.

Personalisation of the spam does not have a big impact on the accuracy of Bayesian filters. If anything, it improves the accuracy of some of them. We suspect this improvement is related to the fact that keywords like "login" in the body of the message becomes a good spam indicator.

5. Conclusions

Filtering at the ISP level. The most efficient way for filtering at the ISP level seems to be using a genetic algorithm. This require a big amount of processing power, but when that is available it is certain a good option. A mail volume-based filter can be established as a first line of defence even when its performance is low because for a high enough threshold they give no false rejects and are computationally cheap.

Filtering at the user level. The best way for a user to filter spam seems to be a naïve Bayesian filter with the Fisher's method like Bogofilter. Depending on how important it is for the user to loose email, Bmf also can be considered as a good option.

The simulator is in the public domain, and can be downloaded from `http://www.cs.kun.nl/~flaviog/spam-filter/`.

References

[1] Martín Abadi, Mike Burrows, Mark Manasse, and Ted Wobber. Moderately Hard, Memory-bound Functions. In *Proc. of the 10th Annual Network and Distributed System Security Symposium*, pages 25–39, February 2003.

[2] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. Spyropoulos. An Evaluation of Naive Bayesian Anti-Spam Filtering. In G. Potamias, V. Moustakis, and M. van Someren, editors, *Proc. of the workshop on Machine Learning in the New Information Age*, pages 9–17. 11th European Conference on Machine Learning, 2000.

[3] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In H. Zaragoza, P. Gallinari, and M. Rajman, editors, *Proc. of the work-*

shop "Machine Learning and Textual Information Access", 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), pages 1–13, 2000.

[4] Adam Back. Hashcash. http://www.cypherspace.org/hashcash, March 1997.

[5] L.F. Cranor and B.A. LaMacchia. SPAM! *Communications of the ACM*, 41(8):74 – 83, 1998.

[6] Rich Drewes. An artificial neural network spam classifier. http://www.interstice.com/drewes/cs676/spam-nn/spam-nn.html, August 2002.

[7] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 139–147. Springer-Verlag, 1993.

[8] William Feller. *An Introduction to Probability Theory and Its Applications, Volume 1*. Wiley Text Books, third edition, 1968.

[9] Eran Gabber, Markus Jakobsson, Yossi Matias, and Alain J. Mayer. Curbing Junk E-Mail via Secure Classification. In *Financial Cryptography*, pages 198–213, February 1998.

[10] Serge Gauthronet and Etienne Drouard. Unsolicited commercial communications and data protection, 2001. Commission of the European Communities.

[11] Paul Graham. Better Bayesian Filtering. http://paulgraham.com/better.html, January 2003.

[12] Hall, R.J. Channels: Avoiding unwanted electronic mail. In *Proc. 1996 DIMACS Symposium on Network Threats*, pages 85–102. DIMACS, 1996.

[13] S. Hambridge and A. Lunde. DON'T SPEW, A Set Of Guidelines For Mass Unsolicited Mailings and Postings (spam). RFC 2635 http://www.rfc-editor.org/rfc/rfc2635.txt, 1999.

[14] Hooman Katirai. Filtering Junk E-Mail: A Performance Comparison between Genetic Programming and Naive Bayes. http://members.rogers.com/hoomank/papers/katirai99filtering.pdf, October 1999.

[15] G. Lindberg. Anti-Spam Recommendations for SMTP MTAs. RFC 2505 http://www.rfc-editor.org/rfc/rfc2505.txt, 1999.

[16] Monty Python's Flying Circus. *Just the words*, volume 2, chapter 25, pages 27–28. Methuen, London, 1989.

[17] Jon Postel. On The Junk Mail Problem. RFC 706 http://www.rfc-editor.org/rfc/rfc706.txt, November 1975.

[18] Gary Robinson. A Statistical Approach to the Spam Problem. http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html, 2003.

[19] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[20] Alan Schwartz and Simson Garfinkel. *Stopping Spam, stamping out unwanted email & news postings*. O'Reilly, 1998.

[21] S.Hird. Technical Solutions for Controlling Spam. In *proceedings of AUUG2002*, Melbourne, September 2002.