*Article*

# Using Live Spam Beater (LiSB) Framework for Spam Filtering during SMTP Transactions

**Silvana Gómez-Meire [1], César Gabriel Márquez [1], Eliana Patricia Aray-Cappello [1] and José R. Méndez [1,2,3,*]**

[1] Department of Computer Science, Universidade de Vigo, ESEI—Escola Superior de Enxeñaría Informática, Edificio Politécnico, Campus Universitario As Lagoas S/N, 32004 Ourense, Spain
[2] CINBIO—Biomedical Research Centre, Universidade de Vigo, Campus Universitario Lagoas-Marcosende, 36310 Vigo, Spain
[3] SING Research Group, Galicia Sur Health Research Institute (IIS Galicia Sur), SERGAS-UVIGO, Hospital Álvaro Cunqueiro Bloque técnico, Estrada de Clara Campoamor, 341, 36312 Vigo, Spain
* Correspondence: moncho.mendez@uvigo.es; Tel.: +34-988-387015

**Abstract:** This study introduces the Live Spam Beater (LiSB) framework for the execution of email filtering techniques during SMTP (Simple Mail Transfer Protocol) transactions. It aims to increase the effectiveness and efficiency of existing proactive filtering mechanisms, mainly based on simple blacklists. Since it implements some proactive filtering schemes (during SMTP transaction), when an email message is classified as spam, the sender can be notified by an SMTP response code as a result of the transaction itself. The presented framework is written in Python programming language, works as an MTA (Mail Transfer Agent) server that implements an SMTP (Simple Mail Transfer Protocol) reverse proxy and allows the use of plugins to easily incorporate new filtering techniques designed to operate proactively. We also include a plugin to perform proactive content-based filtering through the analysis of words included in the body of the email message. Finally, we measured the performance of the plugin and the framework (time required for operation and accuracy) obtaining values suitable for their use during SMTP transactions.

**Keywords:** spam filtering; proactive spam filtering; SMTP transactions; machine learning; profile-based filtering; anomaly detection

check for updates

**Citation:** Gómez-Meire, S.; Márquez, C.G.; Aray-Cappello, E.P.; Méndez, J.R. Using Live Spam Beater (LiSB) Framework for Spam Filtering during SMTP Transactions. *Appl. Sci.* **2022**, *12*, 10491. https://doi.org/10.3390/app122010491

Academic Editors: Akemi Galvez Tomida and Andres Iglesias Prieto

Received: 20 September 2022
Accepted: 16 October 2022
Published: 18 October 2022

## 1. Introduction

Email has become an extremely important resource for communication because it greatly facilitates the exchange of information. The first attempt to build a mechanism similar to modern email (electronic mail) was made in 1965 at the Massachusetts Institute of Technology (MIT) as part of a university file-sharing system that could be accessed from remote terminals. Later, the first email standard was proposed in 1973 by the US DARPA (Defense Advanced Research Projects Agency). It was then introduced in Arpanet in 1977, already including fields such as "From" and "To" and functionalities such as forwarding emails to other users.

As expected, with the widespread use of email for both personal and professional purposes, the proliferation of practices such as spamming and even the sending of malicious emails also began. The first documented case of the commercial practice of spam occurred in April 1994 (available at https://www.wired.com/1999/04/the-spam-that-started-it-all/, accessed on 1 September 2022). By the late 1990s, spam had already become a major problem, as more and more advertisers made use of it as a zero-cost advertising practice that allows for wide dissemination. Thus, in 2002, the European Union launched the "Directive on privacy and electronic communications (available at https://eur-lex.europa.eu/legal-content/ES/ALL/?uri=CELEX%3A32002L0058, accessed on 1 September 2022). This directive includes a specific section on spam, which makes it illegal to send unsolicited commercial communications without the receiver's prior permission. However, the legal measures have not been particularly effective. According to Statista (available at https://www.statista.com/statistics/420400/spam-email-traffic-share-annual/, accessed on 1 September 2022), there has been a considerable reduction in spam traffic in recent years, from 66.76% in 2014 to 45.56% in 2021. This decrease has been possible thanks to hardware-based anti-spam solutions (e.g., Cisco's Email Security Appliance, available at https://www.cisco.com/c/es_es/products/security/email-security/index.html, accessed on 1 September 2022), software-based solutions (e.g., ZeroSpam, SpamAssassin, etc. cited in the document available at https://www.getastra.com/blog/knowledge-base/best-spam-filters-for-emails/, accessed on 1 September 2022), anti-spam firewalls, or spam filtering services included in email servers.

Many of the current spam filtering techniques were not developed and applied until the beginning of the 21st century. Among the first techniques to appear, we highlight collaborative approaches such as the use of DNS whitelists and blacklists (e.g., DNSWL available at https://www.dnswl.org/ and Spamhaus blacklists available at https://www.spamhaus.org/, accessed on 1 September 2022) or based on the use of Nilsimsa [1], such as Razor, Pyzor, and DCC (Distributed Checksum ClearingHouse). Soon after, some standards focused on domain authentication [2,3] (such as SPF, DKIM, or DMARC), which emerged as mechanisms to prevent attacks such as email spoofing and email phishing. Furthermore, intelligent filtering techniques based on Machine Learning (ML) gained popularity very quickly [4]. Other mechanisms employed include the use of compression models [5] or regular expressions [6].

Until recently, most filters were based on content analysis and SMTP protocol information. Identifying keywords that determine the classification of the email as spam or ham is an example of content-based analysis, while searching for the presence of origin servers on blacklists is related to the SMTP protocol. However, in recent years, some works have proposed alternative methods such as spam detection based on the behavior of SMTP servers. The concept of behavior-based detection is not new. The best example can be credit card fraud detection, where security systems that depend on the user's behavioral profile have been deployed. Finally, to complement the server profiling information, completing the detection of errors and inconsistencies by analyzing the data sent through protocols involved in the email transfer (SMTP, TCP-Transmission Control Protocol, and IP-Internet Protocol) has been considered.

Solutions to fight against spam can be classified into several categories depending on the particular time they are applied [7]: (i) preventive, (ii) reactive, and (iii) proactive. The first group, preventive solutions, is mainly based on legislative measures, but their effectiveness is limited. Reactive solutions involve the application of filters after the SMTP transaction has been completed (post-SMTP). These measures are often the most effective as they allow the application of complex techniques, such as content analysis, without a time restriction imposed by the SMTP transaction. Finally, the proactive group solutions aim to apply filters to detect spam during the SMTP transaction.

Due to time restrictions, filters applied during SMTP runtime are in most cases limited to the use of blacklists synchronized with RSync (available at https://rsync.samba.org/, accessed on 1 September 2022) protocol and/or SPF checks. On the other hand, SpamAssassin (available at https://spamassassin.apache.org/, accessed on 1 September 2022) is an example of a post-SMTP filter (reactive) whose application requires a runtime of typically several seconds, which means a delay in the email delivery process to the client; although, it allows the use of complex techniques and content analysis to achieve high efficiency (accuracy). Post-SMTP solutions have the disadvantage of not being able to notify the email sender of the cancellation of message delivery. Consequently, although reactive solutions are the most effective, they have a huge disadvantage in that they are applied after the SMTP transaction, which results in the use of network and server resources. In addition, if a message is rejected, there is no implicit mechanism to inform the sender that his message will not be delivered. In contrast, the use of proactive measures avoids this problem: if a message is discarded during the SMTP transaction, the sender is notified by a valid SMTP code.

Traditional spam filtering methods used during SMTP transactions, such as blacklists, whitelists (domains, IP addresses, and email addresses), or SPF logging, do not provide enough accuracy to reduce the threat of spam messages [8]. Blacklists have the disadvantage of not being updated as fast as spammers change their addresses or domains. In addition, legitimate email providers run the risk of having one of their domains or addresses on a blacklist. Moreover, one of the main problems with SPF is the incompatibility with email forwarders. SPF checks can fail when forwarding an email because SPF components authenticate the forwarding server instead of the original sending server [9].

In this study, we introduce a framework designed to use a wide variety of techniques or tools for spam filtering during SMTP transactions (proactive filtering). The goal is to create a new email filtering framework (LiSB) that ensures that the sender of the email knows if the email has not been delivered to the receiver. We have also developed some profile-based filters (modules) that can be run at SMTP time (thus, proactively). These modules are able to check features such as whether the behavior of a given email server changes over time (which is anomalous). We have also built other modules to check for inconsistencies in the information sent. Considering the disadvantages discussed above, this paper also proposes the use of different Machine Learning (ML) schemes for the development of highly reliable spam filters that operate during SMTP transactions. Although the execution of an ML-based filter during the SMTP transaction may seem a simple task, it is actually a major challenge because it is necessary to find a trade-off between the accuracy achieved by the filter and its complexity (number of features) to avoid an SMTP transaction timeout. With this issue in mind, we have also provided another module to execute simple content-based filters using ML schemes (in particular, Random Forests seems the most suitable one [10]). Finally, we analyzed the performance and computational requirements of our proposal. The framework can be easily extended by incorporating new techniques. The source code of the LiSB framework has been publicly shared in a GitHub repository (https://github.com/sing-group/LiSB, accessed on 1 September 2022).

The rest of the paper is organized as follows: Section 2 summarizes previous efforts and proposals on spam detection during SMTP transactions; Section 3 presents the architecture of the LiSB framework and the evaluation of some ML models that have been considered for use during SMTP transactions; and finally, Section 4 shows the main conclusions and outlines directions for future work.

## 2. State of the Art

Most of the research works on spam filtering [11] are based on analyzing the content of the message body. However, some researchers have focused on the use of message header characteristics for email filtering with good results. A few have even made use of both body and header, as proposed in the present study. Furthermore, it is very important to properly select the input features [12,13] ensuring that they provide enough information for classification and can be extracted/computed quickly (extraction and classification processes must be executed during the SMTP transaction). In the review by Karim et al., it is suggested that the future of Spam detection models can be considered to evaluate email headers, URLs, and domain characteristics.

Most proposals take advantage of ML techniques to classify messages. Among the ML techniques that have performed best in spam classification are Naïve Bayes (NB), Support Vector Machines (SVM), Decision Tree (DT), and Random Forest (RF) [11]. ML techniques have long been applied to spam filtering [14,15], but not in a proactive, real-time way. We have also found some examples of real-time anti-spam techniques but based on continuous training of the ML models [8]. However, the aim of this work is for the filter to be executed during the SMTP transaction, so retraining the model after each successful classification seems to be a major limitation.

To our knowledge, there are no relevant advances in the context of classifying message classification techniques during SMTP transactions. Therefore, although the time to accept/reject the delivery of the message is quite flexible, it is reasonable to think that the shorter the time the email takes to be classified and accepted for delivery, the more emails the server will be able to handle. In addition, it is mandatory to reduce the time needed to accept/reject the delivery of a message in order to avoid delays in the delivery of legitimate emails.

Many approaches to spam detection have been developed and tested, but they are not accurate enough or cannot be applied during the SMTP transaction. Thus, given the lack of solutions for proactive spam filtering, we have developed a framework that allows email administrators to apply a wide variety of checks during SMTP transactions. The proposal is introduced in the next subsection.

## 3. Proposed Solution

This study presents an email filtering system (LiSB) that is implemented before the MTA server, which is able to execute the filtering process during the SMTP transaction time and, therefore, decide whether the message will be accepted (or not) for delivery. The system allows different techniques to be applied concurrently to classify the message as spam or ham, and combine their results to make a single decision. In the solution, we implemented a set of simple techniques including: (i) schemes based on anomaly detection, (ii) SMTP profile-based techniques, and (iii) an ML-based scheme for content-based filtering during SMTP transactions. The available techniques can be easily extended by implementing new modules in Python (see https://www.python.org). As a result, it will be possible to notify the sender that spam is being sent through its server and he will be able to take appropriate actions. Furthermore, as these techniques are executed during the SMTP transaction, all post-SMTP filtering tasks would be avoided, optimizing resource usage and response times.

Anomaly detection strategies are based on checking the content of some fields of the SMTP and email headers and verifying their consistency. Some examples of consistency are: (i) the SMTP MAIL FROM command, and the *From* and *Return-Path* headers of the message are equal and (ii) the SMTP RCPT TO command and the *To* and *Delivered-To* headers of the email are the same.

In addition, some message delivery parameters will be examined to obtain profile information about the MSA and MUA software and the configuration of different domains/users, in order to check the consistency of the information included in a particular message and the profile of the sender (user/domain) stored in the server. To build a domain/user profile, we collect some information from the messages: (i) the *User-Agent* header that reports the message, (ii) the *Received* headers that contain the path of the messages through the SMTP servers, and (iii) the DKIM (DomainKeys Identified Mail) [16] parameters. Since these parameters are not frequently modified, identifying changes can reveal anomalous situations which, in many cases, are associated with the sending of spam messages.

Finally, although it seems that the application of ML techniques during SMTP transactions is impossible, in practice most of the time required is for model building rather than execution. It is also possible to run small ML models stored in RAM memory and based on a limited number of easy-to-extract features. The challenge is to find a small number of features that are easy to extract and provide a lot of information to the process. For example, a recent work [17] has highlighted the number of URLs (Universal Resource Locator) as a very relevant feature for identifying spam content. Considering the ease of counting the number of URLs that are present in a text (by simply applying a regular expression), this feature is very suitable for filtering during SMTP transactions.

The next subsection introduces the architecture used by our proposal and summarizes the details of the software design.

### *3.1. Proposed Architecture*

The filtering mechanism is based on the application of different filtering modules, where each module examines a number of features of the email header and body and, based on these features, classifies the email as possible spam or ham. The combination of the results of the different modules will allow the email to be forwarded or rejected. For this purpose, a message is considered junk if at least one module classifies it as spam. Our proposal supports the definition of filtering exceptions for messages that (i) have been sent from a specific IP address, (ii) whose source address belongs to certain domains, and/or (iii) whose sender has a specific email address. The modularity of the system gives great flexibility to incorporate new filters and even to disable some filtering modules that are not of interest. In addition, this modular system makes use of concurrency techniques that allow the simultaneous execution of each module in a different thread. Figure 1 shows the architecture of our proposal.

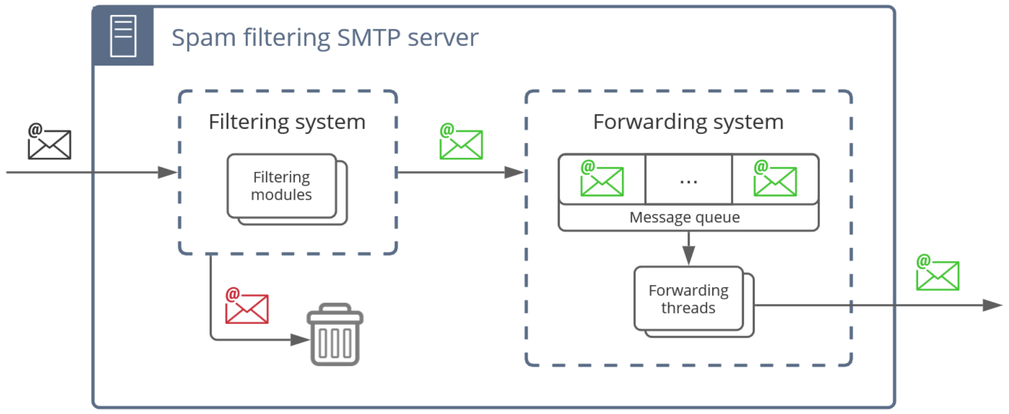


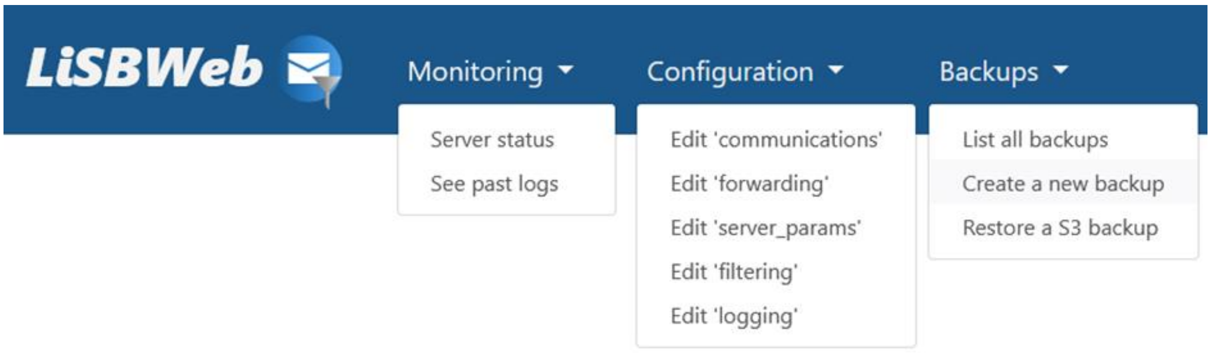
**Figure 1.** This is a figure: LisB architecture.

As shown in Figure 1, LiSB supports the execution of a wide range of filtering modules. Among them, we highlight the following: (i) *FromFilter* that checks that the *From* email header and SMTP MAIL FROM command are consistent, (ii) *ToFilter* that ensures that the *To* email header and SMTP RCPT TO commands are consistent, (iii) *ReturnPathFilter* that ensures *ReturnPath* and *From* email headers are consistent, (iv) *SPFFilter* that executes an SPF authentication [18], (v) *XFilter* that verifies that optional headers (those starting with X) from a particular domain are always the same, (vi) *DKIMFilter* that ensures that the s and d parameters of emails sent from the same domain always have the same values, (vii) *BlackListFilter* that implements a dynamic blacklist scheme, and (viii) ML schemes applied on message content.

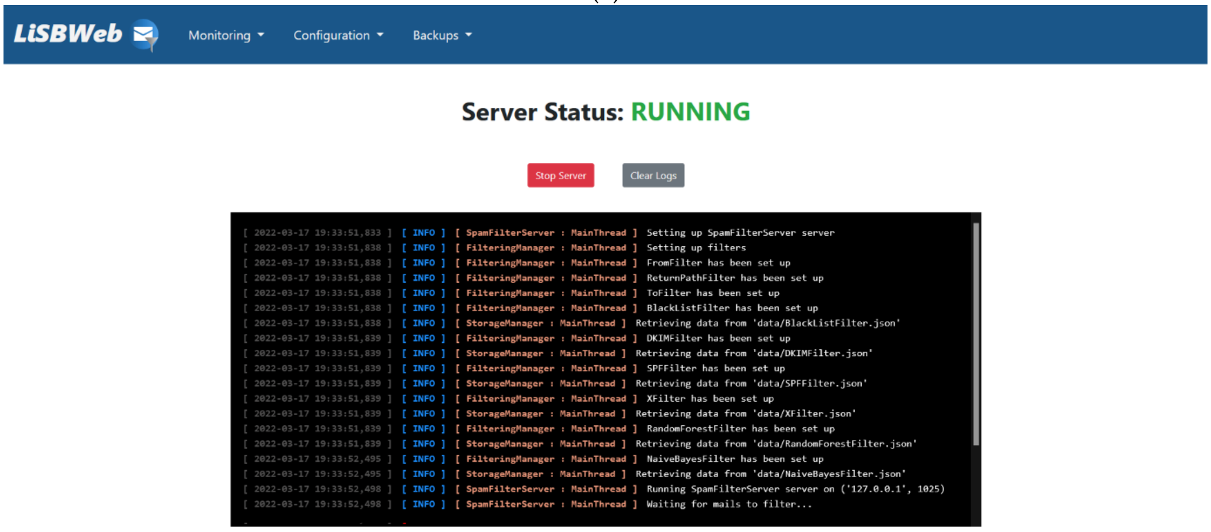
Our framework incorporates a *time_limit* parameter that prevents excessively slow email classifications. To this end, when the classification of a specific message takes longer than the time defined in the *time_limit* parameter, the classification process is interrupted and the message is classified as ham.

Messages classified as ham by the filtering framework are queued for subsequent forwarding. Different threads work on this queue to forward the verified emails to the destination SMTP server. The framework also includes an administration website (see Figure 2), which is used as the user interface of the system and gives access to all implemented functionalities.

As shown in Figure 2a, the administration website includes a wide range of administration tasks including monitoring, configuration, and backup. In detail, it allows the administrator to monitor system events in real time, access past events, enable and disable the SMTP proxy server, create and restore local and S3 backups, and edit all system configuration files. Figure 2b shows the server status console accessed through the "Monitoring/Server status" menu item.



(**a**)



```
[ 2022-03-17 19:33:51,833 ] [ INFO ] [ SpamFilterServer : MainThread ] Setting up SpamFilterServer server
[ 2022-03-17 19:33:51,838 ] [ INFO ] [ FilteringManager : MainThread ] Setting up filters
[ 2022-03-17 19:33:51,838 ] [ INFO ] [ FilteringManager : MainThread ] FromFilter has been set up
[ 2022-03-17 19:33:51,838 ] [ INFO ] [ FilteringManager : MainThread ] ReturnPathFilter has been set up
[ 2022-03-17 19:33:51,838 ] [ INFO ] [ FilteringManager : MainThread ] ToFilter has been set up
[ 2022-03-17 19:33:51,838 ] [ INFO ] [ FilteringManager : MainThread ] BlackListFilter has been set up
[ 2022-03-17 19:33:51,838 ] [ INFO ] [ StorageManager : MainThread ] Retrieving data from 'data/BlackListFilter.json'
[ 2022-03-17 19:33:51,839 ] [ INFO ] [ FilteringManager : MainThread ] DKIMFilter has been set up
[ 2022-03-17 19:33:51,839 ] [ INFO ] [ StorageManager : MainThread ] Retrieving data from 'data/DKIMFilter.json'
[ 2022-03-17 19:33:51,839 ] [ INFO ] [ FilteringManager : MainThread ] SPFFilter has been set up
[ 2022-03-17 19:33:51,839 ] [ INFO ] [ StorageManager : MainThread ] Retrieving data from 'data/SPFFilter.json'
[ 2022-03-17 19:33:51,839 ] [ INFO ] [ FilteringManager : MainThread ] XFilter has been set up
[ 2022-03-17 19:33:51,839 ] [ INFO ] [ StorageManager : MainThread ] Retrieving data from 'data/XFilter.json'
[ 2022-03-17 19:33:51,839 ] [ INFO ] [ FilteringManager : MainThread ] RandomForestFilter has been set up
[ 2022-03-17 19:33:51,839 ] [ INFO ] [ StorageManager : MainThread ] Retrieving data from 'data/RandomForestFilter.json'
[ 2022-03-17 19:33:52,495 ] [ INFO ] [ FilteringManager : MainThread ] NaiveBayesFilter has been set up
[ 2022-03-17 19:33:52,495 ] [ INFO ] [ StorageManager : MainThread ] Retrieving data from 'data/NaiveBayesFilter.json'
[ 2022-03-17 19:33:52,498 ] [ INFO ] [ SpamFilterServer : MainThread ] Running SpamFilterServer server on ('127.0.0.1', 1025)
[ 2022-03-17 19:33:52,498 ] [ INFO ] [ SpamFilterServer : MainThread ] Waiting for mails to filter...
```

(**b**)

**Figure 2.** LiSB administration website: (**a**) LiSBWeb Menu; (**b**) LiSBWeb admin console.

### *3.2. Using ML Schemes during SMTP Transactions*

An important contribution of this study is the experimentation on the use of ML techniques during the SMTP transaction. The use of such techniques in this context seems totally unfeasible due to the execution time required. However, if the model has been previously built and only a few features of the message are used, it is possible to apply some ML techniques to classify the message during the delivery transaction.

This study involved an experiment in which ML techniques were used during SMTP transactions taking advantage of the following features for message representation: (*i*) number of images, (*ii*) number of URLs, (*iii*) content type represented as multiple binary features, (*iv*) extensions of attachments represented as multiple binary features, (*v*) a numeric value that identifies the sender (zero if unknown), and (*vi*) a feature that indicates if *ReturnPath* and *From* headers have different values.

Initially, we evaluated three ML techniques: Support Vector Machines (SVM), Naïve Bayes (NB), and Random Forest (RF). The aim of using different learning algorithms is to run them and compare their performance. For experimental purposes, we created a database of 26,598 emails, 13,299 ham (legitimate), and 13,299 spam. The experimental dataset was built by collecting all messages included in SpamAssassin (available at https://spamassassin.apache.org/old/publiccorpus/, accessed on 1 September 2022) public corpus, the entire Enron dataset shared by Professor Ion Androutsopoulos of the University of Athens (available at https://spamassassin.apache.org/old/publiccorpus/, accessed on 1 September 2022) and part of Bruce Guenter Spam Archive (available at http://untroubled.org/spam/, accessed on 1 September 2022) (messages received from 1 January 2000 to 31 August 2001). Despite their age, the original corpora used in the construction of the experimental dataset have been widely used in recent spam filtering studies [17,19]. These 26,598,074 emails have been processed and as a result, a feature vector has been obtained for each of them. The feature vector, containing 26,598,074 rows and 8 columns (7 features and 1 target attribute), has been used to build 5 ML models (Naïve Bayes Multinomial, Random Forest, SVM with RBF kernel, SVM with Sigmoid kernel, and SVM with Polynomial Kernel). Table 1 shows the time required for building each model in a computer with an Intel(R) Core(TM) i7-8565U microprocessor and 8 GB of RAM memory.

**Table 1.** Time required to build ML models.

| Model | Training Time (Seconds) |
|---|---|
| Naïve Bayes Multinomial | 0.02 s |
| Random Forest | 0.94 s |
| SVM with RBF kernel | 96.87 s |
| SVM with Sigmoid kernel | 56.22 s |
| SVM with Polynomial kernel | 82.50 s |

As shown in Table 1, NB Multinomial and RF algorithms are built quickly. However, these times are not particularly relevant for selecting an ML technique since the models can be built on a stand-alone server. Therefore, the performance of the models is what we need to evaluate in order to select the most suitable one to use for the SMTP transaction. To do so, we run a training/testing experiment using 80% of instances for building the model and 20% for testing. Table 2 presents *precision*, *recall,* and *f-score* evaluations obtained by each algorithm.

**Table 2.** Performance evaluation of analyzed models.

| Model | Precision | Recall | F-Score |
|---|---|---|---|
| Naïve Bayes Multinomial | 0.95 | 0.49 | 0.65 |
| Random Forest | 0.95 | 0.47 | 0.63 |
| SVM with RBF kernel | 0.72 | 0.71 | 0.72 |
| SVM with Sigmoid Kernel | 0.72 | 0.53 | 0.61 |
| SVM with Polynomial Kernel | 0.45 | 0.73 | 0.56 |

As shown in Table 2, the lowest false positive (*precision*) values correspond to the NB Multinomial and RF models. Therefore, these models are the most suitable for filtering spam messages. Furthermore, considering *recall* (and *f-score*, which combines *recall* and *precision* measures), the use of Multinomial Naïve Bayes seems to be the best choice. Taking into account the performance scores, we measured the average time required to classify a message. The result obtained for each model is included in Table 3. This experiment was run on a computer with an Intel(R) Core(TM) i7-8565U microprocessor and 8 GB of RAM memory.

**Table 3.** Time required for classification.

| Model | Classification Time Per Message (Milliseconds) |
|---|---|
| Naïve Bayes Multinomial | 9.749 ms |
| Random Forest | 17.692 ms |
| SVM with RBF kernel | 12.823 ms |
| SVM with Sigmoid Kernel | 11.132 ms |
| SVM with Polynomial Kernel | 12.432 ms |

As shown in Table 3, models can be successfully used during SMTP transactions as long as the time required for classifying an email is always under 18 milliseconds. This would not mean an appreciable delay in the email delivery process since it would take a maximum time of 18 ms to process each email. The next section presents the conclusions drawn from the execution of this study and outlines future research directions

## 4. Conclusions and Future Work

In this study, we introduce the LiSB Framework (source code shared on https://github.com/sing-group/LiSB, accessed on 1 September 2022), which allows network administrators to execute a wide variety of spam filtering techniques during an SMTP transaction. The techniques implemented are based on SMTP anomaly detection, server/user profiling, and content-based ML. These techniques have been tested in a real environment and we have publicly shared the generated source code in a GitHub repository (https://github.com/sing-group/MLClassifers4LiSB, accessed on 1 September 2022). Moreover, we have developed a simple process to install the software on Amazon AWS services (available at https://aws.amazon.com, accessed on 1 September 2022) using Ansible (available at https://www.ansible.com, accessed on 1 September 2022) and Terraform (available at https://www.terraform.io, accessed on 1 September 2022). The Ansible playbook is available on a GitHub repository (https://github.com/sing-group/LiSBSetup, accessed on 1 September 2022).

The LiSB framework includes modules that check for possible anomalies in SMTP and MIME headers. It also includes some modules that are able to build profiles for servers and senders and check for unexpected changes in the messages they sent. LiSB also uses ML to classify messages by examining some properties such as the presence of attachments, and the number of images and URLS. Furthermore, we have proven that the developed modules can successfully run during SMTP transactions with small delays. Furthermore, the modularity of the framework enables the maximum duration of the filtering process to be adapted to the time limits imposed by SMTP transactions.

In the context of ML, we have analyzed the use of three ML algorithms (SVM, Naïve Bayes, and Random Forest) during SMTP transactions. Considering that the ML techniques are applied in real time, our work has focused on reaching a trade-off between the time needed to run the algorithms and their accuracy. The effectiveness of each ML strategy was measured to identify the most suitable one for a real environment and the conclusion was that the NB Multinomial and Random Forest algorithms showed the best performance by returning the lowest number of false positives (FP or ham emails incorrectly classified). Reducing the number of FP errors prevents important emails from being discarded because they are identified as spam. Both algorithms are also remarkable for their low computational time for training and classifying messages.

The modular design of LiSB allows for the easy addition of new filters, as new filtering techniques can be simply incorporated to be executed during the SMTP transaction. On the other hand, being able to include new filtering modules that work under different criteria provides greater control over the email traffic during SMTP transactions, beyond simple blacklists.

Future work includes the development of new modules to extend the functionality of the LiSB framework and to improve the mechanisms used to combine the output of the modules. Moreover, it would be useful to identify new features that can be analyzed to increase the performance of the ML models. Finally, we are aware that the mechanism currently used to combine the output of filtering modules and make a single decision is quite simple and does not provide enough flexibility compared to the mechanisms incorporated in other reactive spam filtering frameworks (e.g., SpamAssassin). We plan to improve the flexibility of this mechanism taking advantage of some optimizations introduced in previous studies [20] to achieve a powerful framework that can be used during SMTP time.

**Author Contributions:** Conceptualization, S.G.-M. and J.R.M.; Methodology, S.G.-M. and J.R.M.; Software, C.G.M. and E.P.A.-C.; Validation, C.G.M., E.P.A.-C. and J.R.M.; Formal analysis, C.G.M. and E.P.A.-C.; Investigation, S.G.-M., C.G.M., E.P.A.-C. and J.R.M.; Writing—original draft preparation, S.G.-M.; Writing—review and editing, S.G.-M. and J.R.M.; Visualization, C.G.M., E.P.A.-C. and J.R.M.; Supervision S.G.-M. and J.R.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the Conselleria de Cultura, Educación e Universidade (Xunta de Galicia) under the scope of the strategic funding of Competitive Reference Group [grant number ED431C 2022/03-GRC].

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Datasets used for the experimentation included in this study are publicly available online and have been cited in the text.

**Acknowledgments:** SING group thanks CITI (Centro de Investigación, Transferencia e Innovación) from the University of Vigo for hosting its IT infrastructure.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Damiani, E.; di Vimercati, S.D.C.; Paraboschi, S.; Samarati, P. An Open Digest-Based Technique for Spam Detection. In Proceedings of the ISCA 17th International Conference on Parallel and Distributed Computing Systems, San Francisco, CA, USA, 15–17 September 2004; Citeseer: San Francisco, CA, USA, 2004; Volume 2004, pp. 559–564.
2. Levine, J. *Email Authentication for Internationalized Mail*; Request for Comments Published by IETF Number 7669; IETF: Fremont, CA, USA, 2019. [CrossRef]
3. Herzberg, A. DNS-Based Email Sender Authentication Mechanisms: A Critical Review. *Comput. Secur.* **2009**, *28*, 731–742. [CrossRef]
4. Blanzieri, E.; Bryl, A. A Survey of Learning-Based Techniques of Email Spam Filtering. *Artif. Intell. Rev.* **2008**, *29*, 63–92. [CrossRef]
5. Bratko, A.; Filipič, B.; Cormack, G.V.; Lynam, T.R.; Zupan, B. Spam Filtering Using Statistical Data Compression Models. *J. Mach. Learn. Res.* **2006**, *7*, 2673–2698.
6. Ruano-Ordás, D.; Fdez-Riverola, F.; Méndez, R.J. Using Evolutionary Computation for Discovering Spam Patterns from E-Mail Samples. *Inf. Process. Manag.* **2018**, *54*, 303–317. [CrossRef]
7. Herrera Silva, J.A.; Barona López, L.I.; Valdivieso Caraguay, Á.L.; Hernández-Álvarez, M. A Survey on Situational Awareness of Ransomware Attacks—Detection and Prevention Parameters. *Remote Sens.* **2019**, *11*, 1168. [CrossRef]
8. Wu, D.; Shi, W.; Ma, X. A Novel Real-Time Anti-Spam Framework. *ACM Trans. Internet Technol.* **2021**, *21*, 1–27. [CrossRef]
9. Chen, J.; Paxson, V.; Jiang, J. Composition Kills: A Case Study of Email Sender Authentication. In Proceedings of the USENIX Security Symposium, San Diego, CA, USA, 12–14 August 2020.
10. Dada, E.; Joseph, S. *Random Forests Machine Learning Technique for Email Spam Filtering*; University of Maiduguri Faculty of Engineering Seminar Series; University of Maiduguri: Maiduguri, Nigeria, 2018; Volume 9, pp. 29–36.
11. Karim, A.; Azam, S.; Shanmugam, B.; Kannoorpatti, K.; Alazab, M. A Comprehensive Survey for Intelligent Spam Email Detection. *IEEE Access* **2019**, *7*, 168261–168295. [CrossRef]
12. Kulkarni, P.; Jatinderkumar, R.; Acharya, H. Effect of Header-Based Features on Accuracy of Classifiers for Spam Email Classification. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 396–401. [CrossRef]
13. Bin Abd Razak, S.; Bin Mohamad, A.F. Identification of Spam Email Based on Information from Email Header. In Proceedings of the 2013 13th International Conference on Intellient Systems Design and Applications, Salangor, Malaysia, 8–10 December 2013; IEEE: Salangor, Malaysia, 2013; pp. 347–353.
14. Dada, E.G.; Bassi, J.S.; Chiroma, H.; Abdulhamid, S.M.; Adetunmbi, A.O.; Ajibuwa, O.E. Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems. *Heliyon* **2019**, *5*, e01802. [CrossRef] [PubMed]
15. Crawford, M.; Khoshgoftaar, T.M.; Prusa, J.D.; Richter, A.N.; Al Najada, H. Survey of Review Spam Detection Using Machine Learning Techniques. *J. Big Data* **2015**, *2*, 23. [CrossRef]
16. Crocker, D.; Hansen, T.; Kucherawy, M. *Domain Keys Identified Mail (DKIM) Signatures*; Request for Comments Published by IETF Number 6376; IETF: Fremont, CA, USA, 2011. [CrossRef]
17. Novo-Lourés, M.; Ruano-Ordás, D.; Pavón, R.; Laza, R.; Gómez-Meire, S.; Méndez, J.R. Enhancing Representation in the Context of Multiple-Channel Spam Filtering. *Inf. Process. Manag.* **2022**, *59*, 102812. [CrossRef]
18. Kitterman, S. *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*; Request for Comments Published by IETF Number 7208; IETF: Fremont, CA, USA, 2014. [CrossRef]

19. Khan, S.A.; Iqbal, K.; Mohammad, N.; Akbar, R.; Ali, S.S.A.; Siddiqui, A.A. A Novel Fuzzy-Logic-Based Multi-Criteria Metric for Performance Evaluation of Spam Email Detection Algorithms. *Appl. Sci.* **2022**, *12*, 7043. [CrossRef]
20. Pérez-Díaz, N.; Ruano-Ordas, D.; Fdez-Riverola, F.; Méndez, J.R. Wirebrush4SPAM: A Novel Framework for Improving Efficiency on Spam Filtering Services: Wirebrush4spam: A novel framework for spam filtering. *Softw. Pract. Exp.* **2013**, *43*, 1299–1318. [CrossRef]