A Scalable Spam Filtering Architecture

Nuno Ferreira[1], Gracinda Carvalho[1], and Paulo Rogério Pereira[2]

[1] Universidade Aberta, Portugal
[2] INESC-ID, Instituto Superior Técnico, Technical University of Lisbon, Portugal
ze@nuno.pro, Gracinda.Carvalho@uab.pt, prbp@inesc.pt

Abstract. The proposed spam filtering architecture for MTA[1] servers is a component based architecture that allows distributed processing and centralized knowledge. This architecture allows heterogeneous systems to coexist and benefit from a centralized knowledge source and filtering rules. MTA servers in the infrastructure contribute to a common knowledge, allowing for a more rational resource usage. The architecture is fully scalable, ranging from all-in-one system with minimal components instances, to multiple components instances distributed across multiple systems. Filtering rules can be implemented as independent modules that can be added, removed or modified without impact on MTA servers operation. A proof-of-concept solution was developed. Most of spam is filtered due to a grey-listing effect from the architecture itself. Using simple filters as Domain Name System black and white lists, and Sender Policy Framework validation, it is possible to guarantee a spam filtering effective, efficient and virtually without false positives.

Keywords: spam filtering, distributed architecture, component based, centralized knowledge, heterogeneous system, scalable deployment, dynamic rules, modular implementation.

1 Introduction

Internet mail spam[2] is a problem for most organizations and individuals. Receiving spam on mobile devices, and on other connected appliances, is yet a bigger problem, as these platforms are not the most appropriate for spam filtering.

Spam can be seen as belonging to one of two major categories: **Fraud** and **Commercial**. In the *fraud* category we include phishing, scams, malware, counterfeit products, and any other criminal activities perpetrated or assisted through Internet mail. In the *commercial* category we include promotional messages – such as newsletters – that we do not want to receive, either being sent legally or illegally[3] from legitimate organizations. We can also classify these two categories as per threat and per volume [1], as shown in Table 1.

[1] MTA – Mail Transfer Agent, commonly referred as mail server.
[2] For this paper, spam is defined as every message that most people do not want to receive.
[3] At the European Community countries all commercial mail messages are opt-in, that is to say that it is illegal to send commercial messages without prior consent from the receiver.

L.M. Camarinha-Matos, S. Tomic, and P. Graça (Eds.): DoCEIS 2013, IFIP AICT 394, pp. 107–114, 2013.

Table 1. Category classification

Category	Threat	Volume
Fraud	*High*	*High*
Commercial	*Low*	*Low*

So, we should primarily address the spam messages in the *fraud* category.

What is an efficient – and still effective – solution for this spam filtering?

Can an architecture, that enables filtering based on the source of spam, be both efficient and effective? Designing and implementing a proof-of-concept for such an architecture, should provide us with enough statistical data to find an answer.

2 Relationship to Internet of Things

The usefulness of information platforms, specially of mobile devices and connected appliances, depends largely on the relevance of the information they provide. The spam received in these platforms wastes resources[4], and reduces the overall relevancy of the information provided.

The proposed spam filtering solution is a scalable and distributed architecture that allows the construction of an ecosystem composed by connected heterogeneous systems to collaborate for a common knowledge. This common knowledge provides a more rational use of resources, as it allows a simple and fast decision to be taken at each MTA and, at the same time, it prevents the duplication of complex decisions as these are based on a common source of information. This solution reduces waste and improves the relevancy of the information provided.

3 Traditional Solutions

Traditional and common solutions for spam filtering can be found in two flavours: **client** and **server** side.

Client-side solutions – that reject or obfuscate messages – must be avoided, as in these cases the sender cannot be properly informed, becoming a major problem for any false positives that might occur. And it is inefficient, as the server uses resources to process and store spam, and the client to receive it.

Most common server-side solutions, such as the Spamassassin [2], are monolithic and run a synchronous filtering process. This is highly inefficient, as each connection is kept active until a complex decision process is complete. On busy servers this might end in a Denial of Service (DoS). It is even more inefficient when an incoming connection, from a previously blocked source, triggers the same process again in any MTA of the infrastructure, due to the lack of a common knowledge.

Approaches have been made to spam filtering architectures, yet they focus more on effectiveness of spam identification, rather than on filtering efficiency. Ma, et al. [3]

[4] Such as CPU, bandwidth, connections, storage, and others related to energy consumption.

propose an architecture for content normalization, to improve content analysis. Cottereau [4] proposes a collaborative architecture for client-side filtering. Yih, et al. [5] also propose a method for increasing the effectiveness of spam identification.

4 The Architecture

In order to save resources[5], we need to make a decision as soon as possible, and this means to start our filtering process as soon as a connection is established to an MTA.

Also, we must avoid client-side bounces, as the mail addresses from senders may be forged, and we would be assisting the spammer by spamming innocent victims. In this respect, server rejections are safe, as these are reported to the MTA of the sender and not to any eventually forged mail address. It is vital to inform the sender of a rejection, so that false positives do not go unnoticed, thus giving the chance to a legitimate sender to find a way to bypass any false positive rejection that might occur.

So, the present architectural solution is designed to operate in mail servers (MTA).

4.1 Working Principle

The working principle of the architecture is a quite simple and straightforward one. It is roughly divided into three main areas and five components, as shown in Table 2.

Table 2. Areas and working Components

Areas	Components
Decision	*Adapter* + *Proxy*
Information	*Knowledge*
Assessment	*Consultant* + *Agent*

A *decision* process is initiated by the MTA, and its goal is to process each SMTP [6] command and decide to either accept or reject the SMTP session or transaction based on available *information*. When the *information* is insufficient to make a reliable *decision*, the MTA returns a temporary fail, and that fact gets registered.

The *information* is a set of data classified in order to allow a quick and reliable *decision*. Unclassified data is processed by an *assessment* to obtain the proper classification. The type of data to consider should provide reliable identification, such as an IP address of an MTA, or a mail domain. Mail addresses can also be considered for a type of data for identification purposes, although they can easily be forged, as long as this fact is kept in mind. The classification should be easy to interpret as a simple rule to reject (*dark*, *black*), accept (*light*, *white*), or as insufficient to make a decision and thus issue a temporary fail (*grey*).

An *assessment* process runs at regular intervals, checking if insufficient *information* was found by a *decision* process. If there is such an event, an *assessment*

[5] Including human resources, whose time is spent with irrelevant information (spam).

is made, in order to obtain a reliable classification (*dark* or *light*) for what was previously insufficient *information* (*grey*).

4.2 Architecture Components

The components of the architecture allow a distributed environment with a centralized source of information. Fig. 1 shows a possible deployment for the components of the architecture, where the dashed lines denote a tight coupling between components, and full lines denote a loose coupling (network connection).

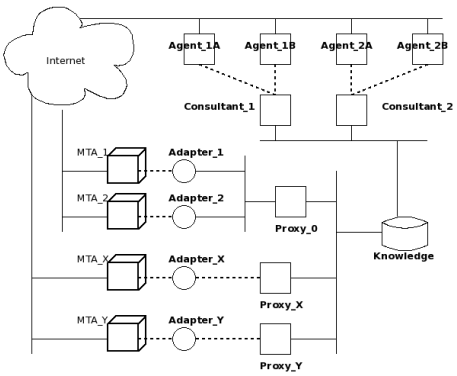


Fig. 1. Diagram of the proposed Architecture

The *Adapter* component allows heterogeneous MTA systems to coexist, and to benefit from a common and central source of information, to which every MTA contributes. This enhances the performance of the system and the rationalization of the resources, as it avoids duplicate assessments. It accesses the solution services through the *Proxy* component. A simple API is provided to enable this access.

The *Proxy* component is the decision centre, based on the *Knowledge*.

The *Knowledge* component gathers data into a database that can be used for informed decision making – by the *Proxy* – and for statistical purposes.

The *Consultant* component is responsible for regularly checking the *Knowledge* for information that requires classification, and to invoke the appropriate *Agents* that are able to properly classify that information.

The *Agent* component acts upon advice from a *Consultant*, and provides the proper classification as a response to an information query asked by the *Consultant*.

5 Results

A proof-of-concept solution was developed to assess the architecture efficiency and effectiveness. The developed *Adapter* was for the Sendmail MTA implementation. The *Proxy* – and associated API – were developed to allow communication only through IPC (Inter-Process Communication), instead of networked communications.

Yet, the developed solution allows for a full analysis and assessment of the architecture on both effectiveness and efficiency.

The whole solution was developed using only Open Source technologies. The test environment was a single CentOS 5 system running a Sendmail MTA server. The *Adapter* and *Proxy* components – as the *Proxy* API – were developed in C. A MySQL relational database was used for the *Knowledge* component. The PERL language was used for the *Consultant* and *Agent* components. Hardware wise, as it is important to assess the efficiency, the Internet connection was assured by an ADSL link of roughly 10 Mbps downstream and 700 Kbps upstream, and the whole software services (plus others like HTTP and DNS) ran on an ATOM D525@1.8GHz processor machine with 2 GB of memory.

Data was collected for two months. For this period of over sixty days[6], at the test server, it was recorded:

- 6,793 distinct source IP addresses
- 14,282 distinct mail addresses (both sender and receiver)
- 23,194 connections, of which:
  - 17,564 reached the envelope phase (sender identification)
  - 2,544 reached the recipient phase

5.1 Effectiveness

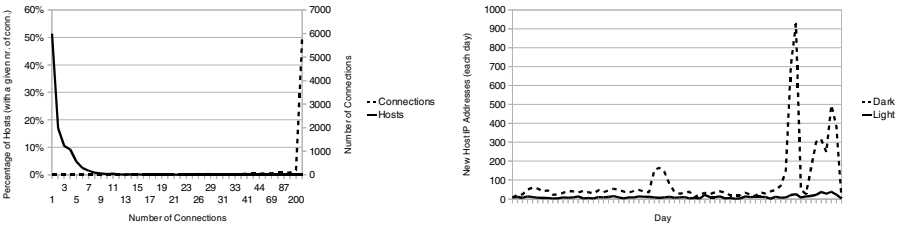
Being an implementation of a proof-of-concept, only one filtering *Agent* was developed for the *Consultant*. This developed *Agent* checks DNS lists – both black and white – for an MTA host IP address.

Yet, the architecture has a grey-listing effect, which acts as an additional filter. This happens because, when a source MTA is not yet classified (as to reject or to accept) at the central common *Knowledge* component, the *Proxy* instructs the MTA to return a temporary failure, to delay the reception until the *Consultant* assesses and classifies the required information. The delay time depends on the frequency set for the *Consultant* to run, plus the assessment duration, and the settings of the sender MTA for retry attempts.

In Fig. 2a we can understand why this grey-listing effect is effective to filter *fraud* spam, as most connections are from IP addresses that do not host real MTA servers and so they do not retry, ending with just one single connection established from that source IP address. We can also conclude, from Fig. 2b, that most legitimate messages (*light*) come from a low amount of new IP addresses[7], and that spam (*dark*) come from a large number of different IP addresses. This is consistent with spam being sent from zombie computers of spam botnets.

[6] The first couple of days, and the last day of the recorded 66 days of data are incomplete.

[7] Should be even less than the amount shown, as these numbers include – most certainly – IP addresses that are in fact source of spam but were not in any of the DNS lists used.



(a) Connections by Host (b) New Hosts per Day

Fig. 2. Distribution of Host Connections

As it can be seen in Fig. 3, one of the DNS black lists (Spamhaus), used by the single implemented *Agent*, was able to identify 5,909 IP addresses as a source of spam from the 6,793 distinct source IP addresses from which connections were established to the testing MTA used. This accounts for over 85% of the sources.

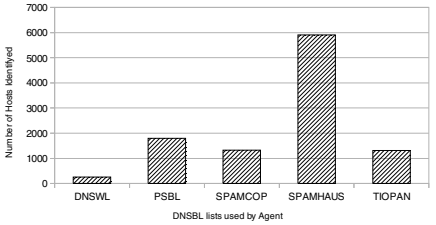


Fig. 3. DNS Lists Effectiveness

As most spam in the *fraud* category has faked mail addresses, adding an *Agent* for SPF [7] validation to the solution would guarantee an even higher effectiveness of the solution.

Considering that over 50% of *fraud* spam is caught by the grey-listing effect, that a single DNS black list can detect 85% of the remaining messages, and that the others DNS black lists can increase this detection ratio, then it is possible to expect a virtual full detection of *fraud* spam just by adding an SPF filtering *Agent*.

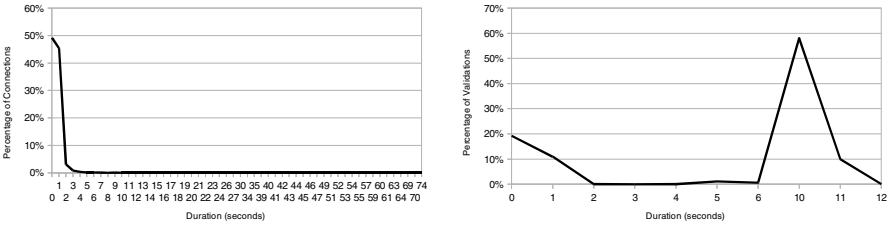
False positives (*FP*) are not an issue, up to this point, even with SPF filtering added. Those *FP* that might result from DNS black lists can be reversed by the usage of white lists. The *FP* resulting from SPF and grey-listing are due to poor server configurations, and those systems administrators should properly configure their systems following the RFC directives. Besides, *FP* are always properly addressed, meaning that the sender will always be notified of a rejection, allowing for an alternative form of communication, or by correcting the system configuration.

5.2 Efficiency

The architecture allows spam filtering to occur asynchronously, and this is important to the solution efficiency. Not only it relieves the MTA server from a longer waiting

period of validations – spam filtering rules and associated procedures – as it only validates one single instance of each data across all MTA in the ecosystem, being it from several instances at the same MTA or from different MTA.

As the *Proxy* component only makes a decision to accept, reject or temporarily fail a connection based on a simple common local knowledge (as the source MTA IP address, or senders domain and IP as in SPF), the MTA server takes little time and effort to make a decision. This can be seen in Fig. 4a, where almost 50% of the connections took less than one second to process, and almost 95% took less than two seconds.



(a) Proxy Performance (b) Consultant Performance

Fig. 4. Performance

On the other hand, the *Consultant* component, as it uses *Agents* that can have complex filtering procedures, or that check external services subject to delays, may take a lot longer to classify a source as spam or ham[8], as it can be seen in Fig. 4b, where almost 60% of the DNS list verifications took 10 seconds to conclude.

So, considering a monolithic solution with a synchronous decision process, where the MTA would need to wait for the filtering process, just to consult DNS black lists would increase each MTA connection duration from an average below one second to one almost 8 seconds long, as it can be seen in Table 3.

Table 3. Processing Times (in seconds)

Component	Minimum	Average	Maximum
Proxy	0	0.7473	74
Consultant	0	7.1089	12

6 Conclusions and Further Developments

It came as a bit of a good surprise that this architecture alone, without any help from a filtering *Agent*, could be responsible for filtering most messages in the *fraud* category. The temporary fail, imposed when no data is available to make an informed decision,

[8] Ham is said to be the opposite of spam, which is to say a wanted message.

has a grey-listing effect that prevents most spam originated from botnets of zombies computers, as most only tries once to send the spam.

The IP addresses DNS lists (both black and white) filtering *Agent*, the only one implemented in the proof-of-concept, was able to identify over 85% of the IP source addresses as being a source of spam. An implementation of an SPF filtering *Agent* would most certainly fill the gap for messages in the *fraud* category.

On the downside, we have a more complex solution due to its many disperse components. Thus, faults due to communication failures are more prone to occur, and security constitutes a major concern. These issues must be addressed with extreme care, as the reliability of the solution depends largely on them.

The final conclusion is that the architecture fulfilled the objectives. It greatly improves spam filtering processing performance, and has an impressive effectiveness record, even with so scarce development of *Agent* components, dubbed as filtering rules and procedures. With a good set of filtering *Agents*, it can achieve an excellent record at spam identification. On top of this, we can run the solution in heterogeneous systems, sharing efforts and information, and scale the distributed system to match the traffic volume of mail messages.

Further developments most certainly will include the development of new *Agents*, such as SPF filtering and trap mail addresses. Yet, the logs recorded will allow a more detailed spam profiling, and this analysis should be used to develop other *Agents* that will increase effectiveness without any loss of efficiency. Also, this server-side filtering can be combined with a client-side filtering that can contribute to the common central *Knowledge*, and with other forms of filtering.

Acknowledgements. I would like express my sincere thanks to professor Gracinda Carvalho and professor Paulo Pereira for all the help in reviewing, guidance, suggestions and incentive provided, whose supervision made this project possible. I also would like to thank my wife Graça, and my parents, for the support, patience and comprehension on my absence of mind and spirit for the time this project lasted.

References

1. Symantec: Internet Security Threat Report, 2011 Trends, vol. 17, p. 31. Symantec Corporation (April 2012)
2. Apache: Spamassassin, `http://en.wikipedia.org/wiki/SpamAssassin`
3. Ma, W., Tran, D., Sharma, D.: On extendable software architecture for spam email filtering. IAENG International Journal of Computer Science 34(1) (2007)
4. Cottereau, L.: A peer-to-peer architecture for collaborative spam filtering. Master's thesis, University of Dublin (2002)
5. Yih, W., McCann, R., Kocz, A.: Improving spam filtering by detecting gray mail. In: CEAS 2007 - Fourth Conference on Email and Anti-Spam. Microsoft (2007)
6. Klensin, J.: Simple mail transfer protocol. RFC 5321 (October 2008)
7. Kitterman, S.: Sender policy framework (SPF) authentication failure reporting using the abuse reporting format. RFC 6652 (June 2012)