Mailspike

Henrique Aparício

1 Introduction

For many years now, email has become a tool of great importance as a means of communication. Its growing use led inevitably to its exploitation by entities that use it to send publicity, virus and phishing, to the most possible number of recipients. This is a problem that can greatly affect productivity in organizations.

This theses, made in AnubisNetworks, an organization specialized in email security, intends to considerably improve some systems that were previously developed by the senior collaborators of the organization. These systems' (in the scope of this project, fingerprinting and IP reputation) objective, is to avoid that the majority of received messages don't have to be subjected to the heavy analysis of traditional content filters, by rejecting the messages as they enter the destination mail server.

Mailspike (or just Spike) is a system by AnubisNetworks, which among others, uses the fingerprinting and sender reputation techniques to filter spam. The fingerprinting technique consists of generating a signature (or hash) from the body of every received email. These signatures are stored in a database which contains a catalogue of signatures generated from emails that were considered as spam. This way, every time an email generates a signatures belonging to the signatures catalogue, the email is automatically discarded without being subjected to the analysis of the content filters. The sender reputation technique uses an heuristic function, which essencially bases itself on the email sending history of the senders, to calculate a reputation value that is attributed to every sender's IP address, also doing botnet detection in the process. The IP's reputation value and the fact that it belong or not to a botnet, are factors that allow us to decide if an email sent by that IP should be rejected.

The first objective of this project, is to completely refactor the fingerprinting system previously developed by AnubisNetworks, by organizing the architecture of the solution in a different way. The new solution should correct the current Spike issues, but maintaining the current protocol, and above all, it must have better performance, since every email will be subjected to this filter.

The second objective is to redesign the reputation system's architecture, by creating a new data storing model and a new model for reputation calculation. With this, it is expected to gain more flexibility, more performance, and computation of fairer values of reputation.

2 Related work

2.1 Types of spam filters

In this theses we identify essencially three types of spam filters to better contextualize its focus: Whitelists/Blacklists, Content based filters and Identifier based filters.

Whitelists and blacklists are a very simple type of filter which basically just lists email or IP addresses which are known to be typical ham or spam senders. In the case of the whitelists every email is rejected except for those that come from sources listed in the whitelist. As opposed to this, the blacklists accept every email except for those that come from sources listed in the blacklist. In spite of their simplicity, these filters are widely used. By themselves these filters are too restrictive (whitelists) or too permissive (blacklists), but when used in conjunction with other types of filters, they can help a lot to reduce the number of false negatives and false positives.[4]

Content based filters are still the most used type of filter in email classification. In this category we can include text analysis filters and statistic filters. The text analysis filters search the messages for words that are typically used by spammers, and classify the message as spam in case these words are found.[4] Statistic filters, or Bayesian filters, use the theory developed by the mathematician Thomas Bayes, which allows us to calculate the probability of occurrence of an event, based on the probability of two or more independent events. These filters verify the frequency of occurrence of words in ham and spam messages, and as they process more messages, they learn to better classify every new email that is received.[4, 7]

Finally there are the identifier based filters. To understand this type of filters, we must first understand the notion of identifier in this context. An identifier is an aspect inherent to the sent message, that is in some way strongly correlated to the message itself.[6] These identifiers can be used not only to classify emails, but also to know previously attributed classifications to the emails. The identifiers that are used in this kind of filters, can be essencially of two types: address identifiers and content identifiers. The address identifiers can be for example the sender's IP address, the sender's email address or the sender's domain. In email classification, a technique that makes use of these identifiers is the calculation of the sender's reputation. Content identifiers consist of aspects that are normally inherent to the email's body, like for example URLs present in the message, or a hash signature of the message body itself, which can uniquely identify the email. In this case, the most frequently used technique, is the fingerprinting technique. The identifier based filters are the scope of this theses.

2.2 Fingerprinting systems

2.2.1 Vipul's Razor

Vipul's Razor (or just Razor) is a distributed and collaborative network of spam detection and filtering which uses the message fingerprinting technique.[5] This system uses a database that is collaboratively populated by their users where, among other information, the hash signatures of messages considered to be spam, are stored, so that a user that receives spam previously received by others, can automatically discard the email.

Razor uses a special kind of signature, which the creator of the system denominated as ephemeral signatures. In order to generate these signatures, instead of applying the hashing algorithm to the mes-

sage's whole body, only a portion of the text is used as input. The portion of text is chosen based on a random number which is constantly altered and collaboratively generated by the Razor users. Doing so, means that at different time instants, an email will generate different signatures. This is mainly a security measure, to keep spammers from knowing exactly what signature an email will generate, because with this they have no way of knowing which text parts will be used to generate the signature.[5]

Other Razor features include text preprocessors, which preform necessary decoding and message body handling, four filtering engines, each one with a different hashing technique, and a reputation and confidence system, which attributes a confidence level to the Razor users.[5]

2.2.2 Pyzor

Pyzor is a fingerprinting system that started as being entirely based on Vipul's Razor, the main difference being the programming language (Razor is in Perl, Pyzor is in Python). The hash signature generation method is similar to Razor's, in the sense that not all the text in the email is used as the hash algorithm's input. Aside from the confidence system and a mecanism that Razor uses to avoid false positives, Pyzor includes the same features as Razor.

2.2.3 DCC

Like in the previous systems, DCC users' email clients are configured to verify if every received email resulting signature, belongs to the signature catalogue. The main difference is that DCC stores all the signatures that are checked against the database, and counts the number of times each of them was checked.[2] Note that there isn't the notion of reporting signatures that a user considers to be from spam emails. Instead, every time a user checks a signature, he is given the total number of times he and other users checked that same signature. With this, the email client decides to reject the email, in case this count is above a certain threshold defined by the user. This means that DCC only looks at the fact that an email is bulk, and ignores if it was solicited or not.[2] For a DCC user to be able to receive newsletters, mailing lists and other bulk mail services, each user has to keep a whitelist of bulk mail senders, whose emails are actually solicited.[2]

The process used to generate the hash signatures is somewhat similar to Razor's, since the hash function's input isn't the text of the whole body, but only some parts of it. This is mainly done so that the signatures can be "fuzzy", in other words, so that there is a possibility of similar emails to generate the same signature.

2.3 Sender Reputation

2.3.1 Gmail Sender Reputation

Among the various mechanisms Google uses to filter email, there is the Gmail sender reputation system.

In order to approach this problem, Google created their own notion of spam. Instead of regarding emails as unsolicited bulk messages, Google decided to consider spam simply as unwanted emails[1].[8] This is valid in Gmail, because the information that is used to calculate a sender's reputation, consists mainly of the number of times every user marks emails from that sender as "spam" or "not spam". In other words, from Gmail reputation system's point of view, users

[1] Unwanted email is different than unsolicited email, because it is possible that a receiving user is interested in an unsolicited email.

are the ones that decide if emails are wanted or unwanted.[8]

Reputation values are associated with the senders' address domains[2]. If we take SMTP limitation into account, this wouldn't be possible, since there is no guarantee that an email was sent from the address that is declared in the email header. In order to solve this problem, Google tries to authenticate every domain using SPF and DKIM/DomainKeys.[8] The SPF used by Gmail is an altered version of the original, with some additional characteristics:

1. First it tries to authenticate with regular SPF, by checking in a DNS record if the sender's IP address is one of the IPs that is allowed to send email from the used domain;

2. If the sender doesn't use SPF, then Gmail uses a "Best-guess SPF" which assumes a domain as being authenticated if the sending IP comes from the same range of IPs as the A or MX records, or if the sending IP's reverse DNS name matches the domain claimed in the email;

3. If both SPF and best-guess SPF fail to authenticate the domain, the domain can still be authenticated if the sender is a subdomain of the DNS PTR's zone;

DomainKeys are also used in Gmail to authenticate domains. Every email the uses DomainKey based authentication, includes a field in its header which is ciphered using the sender's private key. This header field is then deciphered by the receiver using the sender's public key, confirming so that the sender is part of the domain that he claims to be.

[2]Example: The sender *someone@domain.tld* is part of domain *domain.tld*

2.3.2 DCC Reputations

DCC Reputations is a component of DCC (*Distributed Checksum Clearinghouses*), which was mentioned previously, and it is responsible for computing and storing reputation values for email senders. In order to understand DCC Reputation, it is important to remember that DCC's notion of spam is limited to emails being bulk or not, and ignoring if emails were solicited. Sender reputation is calculated using a very simple heuristic, which consists of the ratio between the sender's number of sent bulk emails and his total number of sent emails. This means that the reputation value of a sender is a percentage of bulk mails sent by him.[1]

Reputation values are assotiated with the sender's IP address. Since IP addresses aren't always assotiated with the same sender, reputations are expired one week to thirty days after the last bulk email was reported by a DCC Reputations' client.[1]

3 Motivations and Contributions

When AnubisNetworks first initiated its activity, only traditional content filters were used in its product. However, soon it was realized that these filters weren't able to process every email fast enough, from which arose the necessity of creating Mailspike. So Mailspike was built with few planning, but resulting in a solution which very successfully managed to solve the problem of insuffitient processing capacity traditional filters had. Being this a distributed solution for clients that opt to share received email signatures with other clients, and the fact that there are clients with more and more emails to process, the amount

of emails Spike has to process and the need to add more functionality to the solution, are factors that demand substantial improvements in the current solution. Therefore, Spike is being completely redesigned to solve these problems in an organization wide effort, being the scope of this theses complete refactoring and implementation of Spike's new fingerprinting and reputation systems.

We started by thoroughly analyze the original version of Spike in order to understand which are the existing functionalities and how they work. On one hand this process was difficulted by the lack of documentation, but on the other hand, the extensive analysis of the source code, gave a detailed understanding of the way Spike works. After carefully studying Spike's original version, we proceeded to the redesign and re-implementation of the various components of Spike.

This theses contributions can be enumerated as follows:

1. Introduction of two new abstraction layers in the fingerprinting system, making it more flexible and efficient: reusable communication layer which sends signatures generated by *libspike* (email fingerprinting library) to the server that stores them, and the layer that integrated *libspike* with the MTA (*libmilter*)[3];

2. New algorithm that pre-processes the email in order for the body to be ready for signature generation;

3. New reputation database model, simpler and more flexible: implementation of a relational model that helps to reduce the number of tables and the system's complexity;

4. Total refactoring of the reputation system's architecture, with more flexibility and performance: use of a cache system which allowed significant performance gains, a more logical organization of the reputation library, and use of different *daemons* to execute different operations;

5. Slight alteration in the reputation heuristic, by using exponential moving averages in the computation of reputation values;

6. New botnet and *zombie* detection algorithm: because signatures are made of four parts, we use this feature to do a more effective botnet detection;

After doing some tests to the new solution, we were able to extract promising results. The boost in performance that was desired for the new fingerprinting system was achieved, with increased flexibility. An improved performance and flexibility was also obtained for the new reputation system, along with more fairness in reputation values, increased capacity to process more senders and the possibility to detect much more botnets.

Referências

[1] Distributed checksum clearinghouses reputations web page. http://www.rhyolite.com/dcc/reputations.html.

[2] Distributed checksum clearinghouses web page. http://www.rhyolite.com/dcc/.

[3] Milter web page. https://www.milter.org/.

[4] Spam filter reviews, anti spam tips, advice and spam filter ratings web page. http://www.whichspamfilter.com/TypesOfFilters.htm.

[5] Vipul's razor collaborative spam filtering network web page. http://razor.sourceforge.net/docs/.

[6] D. Alperovitch, P. Judge, and S. Krasser. Taxonomy of email reputation systems. pages 27 –27, jun. 2007.

[7] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, Konstantinos V. Ch, George Paliouras, and Constantine D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. pages 9–17, 2000.

[8] Bradley Taylor. Sender reputation in a large webmail service, 2007.