4th National Conference On Emerging Trends In Computer Science & Engineering (NCETCSE-2018)

Spamdoop: A Privacy-Preserving Big Data Platform for Collaborative Spam Detection

[1]Nayana K.M, [2]Rakshitha B.S, [3]Suhas K.B, [4]Ujwal S, [5]Shashikala S.V

[1234]UG student, [5]Professor and Head,

[12345]Department of information science and technology

Abstract-Spam has become the platform of choice used by cyber-criminals to spread malicious payloads such as viruses and trojans. In this project, we consider the problem of early detection of spam campaigns. Collaborative spam detection techniques can deal with large scale e-mail data contributed by multiple sources; however, they have the well-known problem of requiring disclosure of e-mail content. Distance-preserving hashes are one of the common solutions used for preserving the privacy of e-mail content while enabling message classification for spam detection. However, distance preserving hashes are not scalable, thus making large- scale collaborative solutions difficult to implement. As a solution, through this project, we propose Spamdoop, a Big Data privacy-preserving collaborative spam detection platform built on top of a standard Map Reduce facility.

Keywords: Spam, E-mail, Big data, Detection.

I. INTRODUCTION

The word spam was originally used to describe unsolicited emails sent in bulk. It is hard to define the term spam more accurately. Some argue spam is about the lack of consent on the part of the recipient, while others believe it is about unsolicited e-mail quantity or scale. Other definitions also stressed the commercial nature of spam; for example, the US SPAM act of 2003 established stringent requirements for sending commercial e-mails. Later, spam got closely associated with cyber-crime [1]. Spam emails often try to lure the recipient to click on a fake or infected URL that links to a malicious Website (phishing) or downloads a malicious attachment containing a zero-day exploit (spear-phishing). Spam in all of its forms is still considered as one of the hardest challenges of the connected generation.

This in return drove a massive amount of research towards countering it. The effort led to the gradual decline of spamming activities which lead many to believe that spam was no longer a threat. However, recent thorough studies and statistics have concluded otherwise. In fact, about 66.34% of all e-mails sent worldwide are considered spam e-mails according to Kaspersky's Spam and Phishing Statistics for the first quarter of 2014 which leads us to conclude that it is still an evolving phenomenon and is still an active cyber threat.

While some spam campaigns advertise products and services, others serve more malicious and sinister purposes such as advertising illegal goods and terrorism which was the main topic of spam in the first quarter of 2016. Furthermore, evidence has shown that spam serves as a platform for many other cyber-criminal activities. A major case of link between spam and criminal activities goes back to November 11th 2008, when two Internet upstream providers blocked the network access of McColo, a U.S. based web hosting service provider, reporting that the firm's servers were being used for illegal activities. The Washington Post later reported that McColo was used by organized crime as a host for e-mail sales of counterfeit pharmaceuticals, fake security products and child pornography. Following McColo's shutdown, security agencies noticed a decrease of 75% percent in unsolicited e-mail sent worldwide1.

In the last ten years, multiple sources have mentioned spam being used for distributing malicious software, phishing, and delivering other social engineering related attacks [2]. In the 1990s, the average PC user received one or two spam messages a day. Some years ago, the amount of spam grew to an estimated 190 billion messages sent per day.

Spammers collect gross worldwide revenues of the order of $200 million per year. Today, the huge quantity of spam generated and distributed on a daily basis makes fighting spam a tall order in terms of processing power and bandwidth. Rather than being selective in their campaigns, spammers aim to reach as many users as possible in a short period of time.

Many specialized software tools for bulk mail delivery are available, including Shotmail, Batware, Bulk e-mail generator, and others. All these tools support bulk e-mail address collection, the creation of mailing lists and pushing large amounts of e-mails.

Recently, spam delivery has become integrated with cyber-crime toolkits such as Blackhole, Whitehole, Cool pack, Zeus and others. Furthermore, spammers have started using botnets to speed up the spread of spam. The suppression of spam involves the need to understand complex patterns of behavior and the capacity to detect emerging types of spam.

II. SYSTEM MODEL

Here, we will be implementing the Spam Detection Algorithm to classify the uploaded emails into spam mails and legitimate emails. The process will be comprising of three phases:

- The Obfuscator**:** The obfuscator Encodes e-mail content. The encoding allows for parallel spam processing without compromising the privacy of the original e-mail.
- The Parallel Classifier**:** The Parallel classifier Leverages the properties of the encoding in order to parallelize the process of routing digests corresponding to similar messages to the same bucket.
- The Anomaly Detector**:** The Anomaly detector Detects spams based on the size of the buckets and their rate of growth.

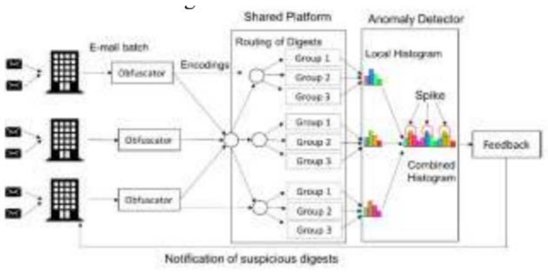


Fig. 2.1 System Architecture

Collaborative Privacy Aware Spam Detection Drawbacks:

- The security properties have not been thoroughly tested and thus are not recommended for private data sharing.

II. PROPOSED METHODOLOGY

Spamdoop is a platform that allows multiple entities to collaborate in early detection of bulk spam campaigns. Our platform also satisfies the privacy requirements of participants. An overview of Spamdoop architecture is shown in figure in the previous section, highlighting the three key components of the system:

- The Obfuscator
- The Parallel Classifier
- The Anomaly Detector

III. PRIVACY AND COMPRESSION

In this subsection we discuss the reversibility of our encoding. Accuracy, design choices and targeted attacks on then coding are also discussed.

Given that a common attacker model used in cloud platforms is a passive adversary, we remark that our proposed encoding ensures that a honest-but-curious platform hosting the Spamdoop service will not able to read any data post second stage hashing. Instead, an active adversary will try to compute an e-mail given the output of the two stages of encoding. That means that both stages of encoding would have to be reversed.

Distance-Preserving Hashing Drawbacks:

- This require the need for distance computation by ensuring that the output is not easily affected by minor modifications (An added computational overhead)
- Data owners choose not to trust distance-preserving hash techniques for personal data sharing because these techniques have not been scrutinized

Collaborative Spam Detection

Drawbacks:

- No Explicit control of parallelization.
- These approaches do not consider privacy of emails.
- Authors of these approaches have reported that their computation may take a long time.

Let us start analyzing reversibility from the second stage. By using a cryptographic hash function such as SHA, Spamdoop encoding inherits its non-reversibility properties, meaning that it is computationally unfeasible to revert the output of stage two back to stage one. However, in some cases where security is not an issue and performance is prioritized, a short representation such as CRC, which is easily reversible, could be favored. While the first stage of obfuscation offer no privacy guarantees, it offers basic anonymization of e-mails because of three factors:

- The output contains the entire language model regardless of the trigram count where the appearance of the trigram is masked.
- The output of the first stage replaces trigram counts with channels that represent a range of possible occurrences.
- The position of the occurring trigrams are masked. The output does not show the order in which the trigrams appeared in the e-mail.

Having neither an exact count of the trigrams nor the set of occurring trigrams (we recall the entire language model is included and set to channel one) makes it nowhere near trivial to reverse the first stage to the original e-mail. However, we stress that the first stage's output is not suitable for data sharing because it does not offer any privacy guarantees.

Furthermore, the representation obtained from the first stage is very large, thus the cryptographic hash in second stage provides both a more compact and more secure output. Given the output of the first stage, an attacker can try to guess if a word is present in an e-mail by looking for improbable trigrams. If an unlikely or rarely occurring trigram (in the English language) is found in the first stage output, then the choice of words generating that trigram can be narrowed down. However, would find difficulty because the entire language model (all possible trigrams) are included in the representations and are set to channel one, thus no educated guess can be made on which trigram is in the e-mail unless it occurs often enough to put it into the next channel which can be unlikely for rarely occurring trigrams. On the other hand, an attacker can guess with a probability if a trigram does not occur at all or often enough. However, that information is seldom useful to attackers.

A more sophisticated attack on our encoding is to specifically alter the messages so that a certain trigram is increased or decreased enough to change the output of the first stage, thus changing the final hash output. To achieve this, the spammer has to inject a specific trigram enough to move it from its channel to the neigh bouring channel and half of the neighbouring channel by doing this, the similarity between both e-mails cannot be detected because of the effects of the cryptographic hash. However, this would substantially increase the time needed to generate the spam messages from the template because the attacker is also bound by computational constraints and needs to push a large number of e-mails quickly.

One of the most well documented trade-offs of any spam detection platform regards achieving performance, accuracy and privacy at the same time. This remains true in our proposal. Indeed, we sacrifice a small degree of accuracy for performance by introducing the first stage. For the second stage, however, the user can choose to prioritize performance over privacy depending on the chosen hash. Due to the progress of in-memory computation for Map Reduce, populating the entire possible trigrams combinations of a character set is a computationally inexpensive.

Finally, we would like to discuss compressing an email using our digest. The language model created in the first stage contains every possible trigram combination. However, the value associated with a trigram can range from a single digit to multiple digits. Since a cryptographic hash function with a fixed output length is used for our final representation, the large size of the language model is always compressed to a fixed size regardless of the trigram count. However, the choice of final representation is up to the developer to decide.

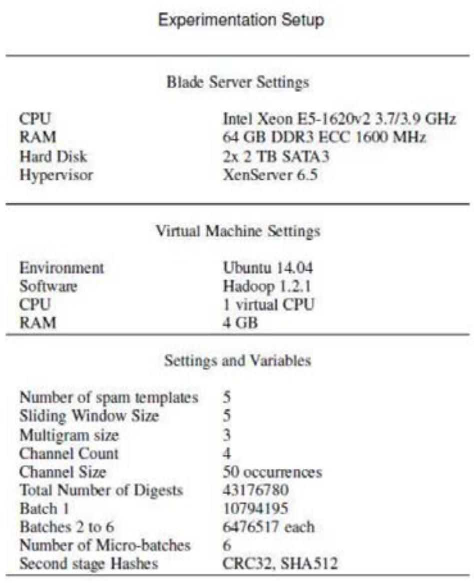
IV. FUNCTIONALITY AND SCALE OF TEST

The results of the first set of functionality tests are reported Table 3 where the third column shows the results obtained by our encoding and the fourth column are the results obtained using Nilsimsa. The goal of this test is to verify if our encoding is able to group messages generated from the parent templates and place them under the same encoding.

Thus, a smaller group number means less variety of encodings generated for a spam campaign which translates into better detection of spam campaigns. Our technique managed to group the child templates under the same encoding except for the third template which was an e-mail that is repeated three times making the trigram count skewed before the spamming software began processing it. Thus, a targeted attack aiming to inject a significant number of a trigram is the most efficient way to force an encoding change. However, this process is computationally expensive and would makes the spam e-mail very unnatural and suspicious.

Such attacks can be controlled using more intelligent channel sizes which is out of the scope of this work. Since Nilsimsa and ssdeep do not readily group similar e-mails under the same hash, an extra step had to be performed. Each hash value was compared with all of the other hashes where scores over 56 (recommended by the original paper) and 103 were considered a match for Nilsimsa, while scores over 54 and 80 were considered a match for ssdeep. Nilsimsa scored perfectly in our tests using both thresholds since it was originally designed for such applications.

SSdeep also scored perfectly using the lower threshold with minor misses on the higher one. To evaluate if Spamdoop can scale up, we devised a test in which the performance of the system is measured based on the time spent by working nodes to group 43; 176; 780 digests. The effects of adding working nodes and the choice of second stage hashing on the time are also reported. Initially, the digests were set to be grouped by a single virtual machine, then we gradually increased the number of working machines.


Experimentation Setup

Blade Server Settings	
CPU	Intel Xeon E5-1620v2 3.7/3.9 GHz
RAM	64 GB DDR3 ECC 1600 MHz
Hard Disk	2x 2 TB SATA3
Hypervisor	XenServer 6.5
Virtual Machine Settings	
Environment	Ubuntu 14.04
Software	Hadoop 1.2.1
CPU	1 virtual CPU
RAM	4 GB
Settings and Variables	
Number of spam templates	5
Sliding Window Size	5
Multigram size	3
Channel Count	4
Channel Size	50 occurrences
Total Number of Digests	43176780
Batch 1	10794195
Batches 2 to 6	6476517 each
Number of Micro-batches	6
Second stage Hashes	CRC32, SHA512

V. CONCLUSION

Spamdoop aims to facilitate collaborative spam detection by taking into account the privacy of all the participants and the scale of collective data. A major innovation of our stage encoding based is representing and then hashing the entire language model. This allows us to group spam emails generated from the same parent template into one bucket. Our encoding scales well on Map Reduce platforms, outperforming distance-preserving hashing techniques. Also, an efficient bucketing technique was deployed to simplify grouping of digests. The histogram based anomaly detection we used to distinguish between ham and spam readily lends itself to Hadoop implementation; however, we remark that our framework is agnostic with respect to the specific anomaly detection technique.

We used an adversary platform mimicking real spamming platforms to test the effectiveness of our encoding and the performance of our parallel classifier against digests of more than 43 million synthetic e-mails. For a single large batch, our tests showed that it is possible to reduce the grouping time of digests by 53% when distributing the work across four nodes. Furthermore, the computation time was further decreased by 57% when using CRC32 on a single working node and 46% in the case of four nodes compared to SHA512. On the other hand, processing the six batches was 52% faster on four nodes compared to only one node and 13% when switching to CRC32. We believe these results to show clearly that Bigdata spam detection technique are ripe for in-production deployment.

The spam detection mechanism currently uses the email body only. However, the first stage of obfuscation can be applied to trigram techniques such as which can be used for grouping binary data such as images. Instead of using the entire language base to produce the first stage output, one can intelligently remove unlikely trigrams if the spammer has no way of knowing which. Finally, different strategies for choosing channel sizes can be employed where channels can be of different sizes.

VI. FUTURE SCOPES

The spam detection mechanism currently uses the email body only. However, the first stage of obfuscation can be applied to trigram techniques which can be used for grouping binary data such as images. Instead of using the entire language base to produce the first stage output, one can intelligently remove unlikely trigrams if the spammer has no way of knowing which. Finally, different strategies for choosing channel sizes can be employed where channels can be of different sizes

VII. REFERENCES

[1] S. Heron. Technologies for spam detection. Network Security, Vol. 2009: Pages 11 - 15, 2009.

[2] D. Wang, D. Irani, and C. Pu. A study on evolution of email spam over fifteen years. In 9th International Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), pages 1 - 10. IEEE, 2013.

[3] Kaspersky Lab, “Spam and Phishing Statistics Report Q1- 2014”. [Online]. Available:

[4] https://usa.kasperskv.com/internetsecuritvcenter/threats/ spam -statistics-report-q1-2014/. Accessed: 2016-06-2.

[5] Kaspersky Lab, “Spam and Phishing Statistics for 2016”. [Online] .Available:https://www.kaspersky.com/about/pressreleases/2016 kaspersky-lab-reports-significant-increase inmalicious- spam-emails-in-q1-2016. Accessed: 2016-06-2.

[6] P. Wood, B. Nahorney, K. Chandrasekar, S. Wallace, and K. Haley.Symantec internet security threat report. Vol. 21: pages 27-36, 2016.

[7] G. Cormack. Email spam filtering: A systematic review. Foundations and Trends in Information Retrieval, Vol. 1: Pages 335-455.