IRILD: an Information Retrieval based method for Information Leak Detection

Eleni Gessiou [1], Quang Hieu Vu [2], Sotiris Ioannidis [1]
[1] *Institute of Computer Science, FORTH, Greece*
[2] *Institute for Infocomm Research, Singapore*

Abstract

A popular approach for detecting information leaks of sensitive data is to partition them into multiple parts and use fingerprints generated from these parts to check against fingerprints from parts of outgoing documents. The weakness of this approach is that it incurs a high cost in both fingerprint generation and leak detection because every part of the document needs to be checked. Additionally, it is prone to false positives if a lot of fingerprints contain common phrases, since they often exist in many documents. In this paper, we propose a solution based on information retrieval to address these problems. The core idea of our solution is to identify common phrases as well as non-sensitive phrases to eliminate them from the fingerprinting process. While non-sensitive phrases are identified by looking at available public documents of the organization that we want to protect from information leaks, common phrases are identified with the help of search engine such as Google. In this way, our solution not only reduces the cost of performing leak detection but also increases the accuracy of the result. Furthermore we conducted an experimental evaluation of the the efficiency and effectiveness of our proposed solution.

1 Introduction

Information leaks are a critical problem of computer systems. The leak of confidential data, be it accidental or intentional, may cause huge losses to the owner of data. According to a study [9] in 2006 conducted by the Ponemon Institute at 31 organizations that lost confidential information, the average cost of a case of information leak was approximately 4.8 million USD.

There are two primary solutions for information leak detection. The first one is to use specific expressions, keywords or phrases to identify confidential information. For example, a leak of a Master card number can be detected by searching expressions of 16 digits starting with two digits in the range from 51 to 55. The disadvantage of this solution is that it cannot be employed if confidential information cannot be expressed by expressions, keywords, or phrases. In this case, the alternative solution is to generate fingerprints of confidential information, which can appear in any structure, and check the generated fingerprints against fingerprints obtained from outgoing traffic for information leak detection.

To protect a confidential document from information leak, a simple approach is to generate a fingerprint of the whole document and check this fingerprint against fingerprints of all outgoing documents for leak detection. Since this approach cannot deal with the case where the leaked information is only in part of the document, the popular approach is to employ cyclical hashing to generate a series of fingerprints for the document and use this series of fingerprints to check against the series of fingerprints of outgoing documents. Nevertheless, there are still two weaknesses of the popular approach. First, the approach incurs a high cost in both fingerprint generation and leak detection since every part of a document needs to be checked. Second, it is prone to false positives if a lot of common phrases are used in confidential documents, and hence a lot of fingerprints of these phrases are generated and checked against fingerprints of outgoing documents. The reason is because these phrases often exist in all documents.

To address the problems of the popular approach, in this paper, we propose a solution based on information retrieval to identify only phrases containing sensitive information for fingerprinting. The basic idea of our solution is to check the popularity of phrases before fingerprinting by two ways. On the one hand, we look at available public documents of the company or organization that we want to protect from information leak. If the phrase exists in these documents, it does not convey any secrete information, and hence it is not sensitive phrase. On the other hand, we submit the phrase to a search engine such as Google and measure the number of returned results. Intuitively, the higher the number of returned re-

sults is, the more the popularity of the phrase is. What this means is that if a phrase has a large number of returned results, it is a common phrase. In summary, our work makes the following major contributions:

- We propose an information retrieval based method to identify common phrases as well as non-sensitive phrases to eliminate them from the fingerprinting process of confidential documents. In this way, our method improves both the processing speed and the accuracy of information leak detection.
- To evaluate the popularity of a long combined phrase which has no returned result from search engines, we suggest a simple technique to split the phrase into shorter phrases and identify the popularity of the phrase based on the popularity of its divided phrases.
- Extensive experiments have been done to evaluate the effectiveness and efficiency of our proposed method.

The rest of the paper is organized as follows. Sections 2 and 3, we discuss prior work and introduce background knowledge. In Sections 4 and 5, we present and evaluate our solution. Finally, we conclude in Section 6.

2 Related Work

In the past few years, a number of white papers have been written discussing different aspects of information leak prevention in general, and how to detect information leak in particular. These include [8], which introduces basic solutions to detect and prevent information leaks, [11], which presents testing and evaluation standards for information leak prevention products and [9], which studies the cost incurred by information leaks in practice.

In general, there are two main approaches to information leak detection. One is based on defining sensitive expressions, keywords or phrases. This way, information leaks are detected if the outgoing traffic contains the specified expressions, keywords or phrases. The other is based on fingerprints of information. For example, a popular approach for information leak detection in documents is to divide them into multiple parts and generate fingerprints of these parts. These fingerprints are checked against fingerprints of similar divided parts of outgoing traffic for leak detection.

A special type of information leak is information leak from applications. To deal with this type of leak, [10] proposes Privacy Oracle, a solution that tests an application with different inputs and maps input perturbations to output perturbations to detect potential information leaks. Alternatively, [4] introduces the use of shadow execution that runs two copies of an application at the same time in which, the one containing personal information is kept away from accessing the network while the other with non-confidential data is used to communicate over the network. The response from the network is then shared for both copies. These solutions are, in fact, complementary to these basic solutions as well as the solution presented in this paper.

In addition to information leak detection, there exists a class of techniques that address access control to prevent information leak. Access control is required in cases where the information is available to someone but should be restricted from others. With respect to this aspect, [5] introduces a solution to avoid information leak caused by accidentally sending emails to unintended recipients. The basic idea of this solution is to measure the similarity between the current outgoing email and previous outgoing emails of the same recipient. If there is a big difference between them, the current outgoing email may contain information leak. On the other hand, [13] presents CLAMP, an architecture that protects confidential information in web servers by enforcing strong access control on user data while isolating code running on behalf of different users. These access control techniques are orthogonal to the basic solutions as well as the solution presented in this paper.

The use of search engines to detect information leaks has been introduced in [6]. However, in our work, the purpose of using search engines is simply to detect inferences between keywords. The purpose of finding inferences between keywords is to discover sensitive documents that do not contain specified sensitive keywords but contain closely associated keywords. It is because with high probability, these sensitive documents also contain confidential data. With respect to inference detection, prior to this work, web based inference has been significantly studied in a number of papers such as [7], [12], [14].

3 Background

A popular approach to detect information leak from a confidential document is to employ cyclical hashing to split the document into multiple parts and generate fingerprints for these parts. In particular, given a document, the method repeatedly creates fingerprints for strings of C characters from the start to the end of the document, offset by O characters each time (C and O are predefined parameters). An example is shown in Figure 1, where C and O are set to 30 and 10 respectively.

Given an outgoing traffic (e.g., an outgoing email or a file uploading to an outside server), cyclical hashing is also used to generate a set of fingerprints for the traffic. These fingerprints are then checked against those extracted from confidential documents to detect information leak. As in Figure 1, since the first three fingerprints of the outgoing document match the first three finger-

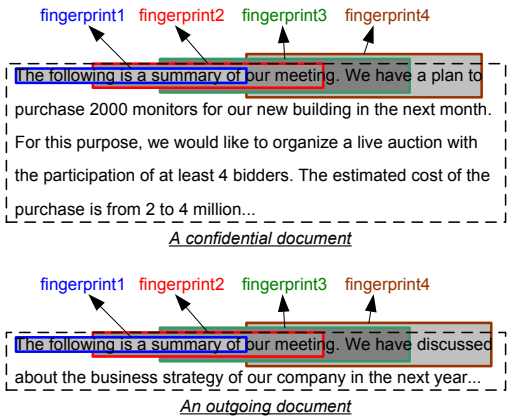


Figure 1: An example of using cyclical hashing

prints of the confidential document, it is considered to have partial information leak.

A problem with this approach, illustrated in the example of Figure 1, is that it causes false positives when common phrases or sentences are used. In this example, even though the common sentence "the following is a summary of our meeting" appears in both the confidential document and the outgoing document, since it does not convey sensitive information, there is actually no information leak.

4 IRILD

To avoid false positives involving common phrases as in the example of Figure 1 and also reduce the unnecessary cost of generating and checking fingerprints of the common phrases, we propose IRILD, an information retrieval based method that is able to identify common phrases and eliminate them from the fingerprinting process. Our main idea is to evaluate the popularity of phrases by submitting them to a search engine such as Google and measure the number of returned results. Basically, the higher the number of returned results of a phrase is, the more common the phrase is. For example, since there are approximately 15,300 results returned from Google by searching the sentence "the following is a summary of our meeting", this sentence is considered as a common sentence and no fingerprint should be generated for it. Besides, assume that the organization, which employs IRILD, maintains public documents (e.g., documents in public folders of the company's web site), we also suggest to eliminate phrases that can be found in public documents from the fingerprinting process because these phrases contain already known information.

In general, IRILD generates fingerprints for confidential documents in three steps. In the first step, similar to the popular approach introduced in Section 3, IRILD employs cyclical hashing on each confidential document to generate a set of candidate strings for the fingerprinting process. Note that by using a fix number of characters to generate strings, a candidate string may not be a complete phrase (e.g., as in the example of Figure 1, the second candidate string, "ing is a summary of our meetin" is not a complete phrase). Thus, from this set of candidate strings, IRILD needs to generate a set of candidate, complete phrases. Each phrase corresponds to a candidate string and is the shortest phrase that totally covers the string. For example, the candidate phrase corresponding to the second candidate string in the example of Figure 1 is "following is a summary of our meeting". In the second step, IRILD identifies public phrases and common phrases and remove them from the set of candidate phrases. Finally, candidate strings associated with remaining sensitive candidate phrases are used to generate fingerprints. The overview of IRILD's processing steps to generate fingerprints for a confidential document is illustrated in Figure 2. In the subsequent sections, we respectively introduce the techniques to identify public phrases and common phrases in details. Note that while the fingerprinting process of IRILD is different from the popular approach, the information leak detection is still the same, i.e., fingerprints of confidential documents in the database are used to check against fingerprints of outgoing documents for information leak detection.

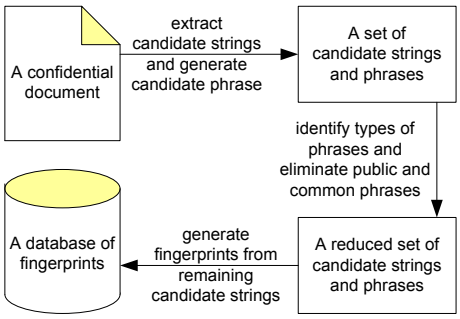


Figure 2: An overview of fingerprint generation

4.1 Public phrase identification

The task of identifying public phrases from a set of candidate phrases is simply a search of these phrases from available public documents. To fulfill the task, a solution is to employ the basic information retrieval technique to create document indices for public documents and inverted indices for words in public documents. Basically, each document index records words that appear in a public document. On the other hand, each inverted index records positions of a word in all documents the word appears. For example, the structure of a document index can be $\{doc_x$: $word_1$, $word_2$, $word_3$, $\cdots\}$ while the structure of an inverted index can be $\{word_y$: $[doc_1$: pos_{11}, pos_{12}, pos_{13}, $\cdots]$, $[doc_2$: pos_{21}, pos_{22}, pos_{23}, $\cdots]$, $\cdots\}$. To detect whether a phrase is a pub-

lic phrase, we first parse the phrase into a list of words. After that, we search document indices to see if there is any document that contains all words in the list. If such a document exists, we then retrieve inverted indices of the words to see their positions in the document. If the words appear at adjacent positions, it forms a phrase in the document. In this case, we conclude the checking phrase is a public phrase since the checking phrase matches a phrase in a public document. While this solution always generates exact results without false positives, it incurs a high cost in search. As a result, this technique can only be used if the number of public documents is not very large. An alternative solution is to also employ cyclical hashing to generate fingerprints for public documents and compare these fingerprints to the fingerprints of the candidate strings to check if the candidate strings exist in public documents. This technique is often used if the number of public documents is large.

4.2 Common phrase identification

As previously discussed , to check whether a phrase is a common phrase, we submit the phrase to Google and measure the number of returned results. Basically, a phrase is a common phrase if there are a lot of returned results. That is because if the phrase can be found easily by the search engine, with high probability, it does not contain secrete information. On the other hand, if the search engine returns only a few results for the search, the phrase is considered as a sensitive phrase. To make easily evaluate the popularity of a phrase, we propose a formula to calculate the popularity score of the phrase from its number of returned results by the search engine as follows:

$$Score(P) = \log_{10}(N(P) + 1) \quad (1)$$

where P is the evaluating phrase and $N(P)$ is the number of results returned from searching P. A common phrase is a phrase whose popularity score is greater than a predefined threshold K. For example, if we set K to 4 and the search of a phrase P has 20,000 number of results, P is considered as a common phrase because $Score(P) = \log_{10}(20,000 + 1) > 4$.

The above technique to identify common phrases usually works if candidate phrases are not long (e.g., common phrases are single phrases). However, in cases where candidate phrases are long (e.g., when C is set to a large value, the candidate phrases that cover candidate strings are also long), a problem comes. In these cases, candidate phrases can be a combined phrase, a sentence, or even some sentences or a paragraph. While it is easy to identify the popularity of a single phrase by the search engine, it is more difficult to do the same thing for a combined long phrase or sentence. It is because with high probability, the long phrase cannot be found by the search engine. To deal with this problem, we suggest a simple way to parse the combined phrase or sentence into smaller single phrases and calculate the popularity of the combined phrase from the popularity of the split single phrases. In particular, let P_1, P_2, $\cdots$, P_n be n single phrases that are split from a combined phrase P and $Score(P_1)$, $Score(P_2)$, $\cdots$, $Score(P_n)$ be their popularity scores. $Score(P)$ is calculated as:

$$Score(P) = \min_{i=1..n}\{Score(P_i)\} \quad (2)$$

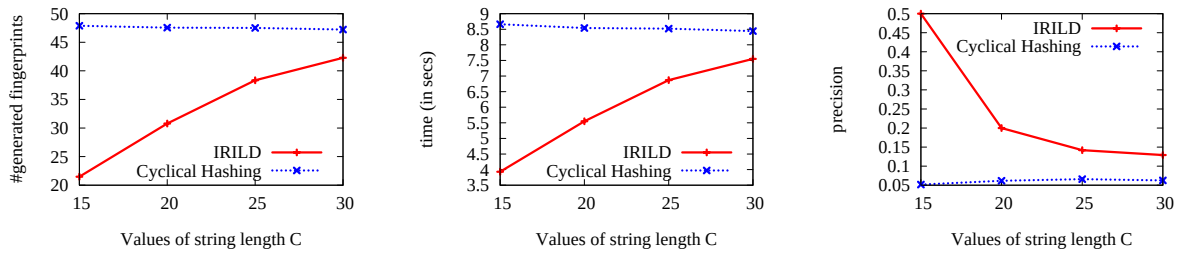
The rational behind the above formula is that the score of the combined phrase should be equal to the minimum popularity score of its members. The intuition of the formula is straightforward. A combined phrase is a common phrase if all of its sub-phrases are common. On the other hand, a combined phrase is a sensitive phrase if at least one of its split phrases is a sensitive phrase.

In addition to submitting a phrase to Google to check its popularity, we observe that if a phrase contains numbers, it is often a sensitive phrase. From our experience, we believe that in most cases, numbers stand for sensitive information, such as telephone numbers, dates of birth, amounts of money, etc. As a result, we decided to consider all phrases containing numbers as sensitive phrases. In this way, we save time by not querying the Google for these phrases. In general, we suggest that prior to submitting a phrase to Google, we check if there is any existing number in the phrase. If there is, the phrase is considered as a sensitive phrase without the need of searching the phrase in Google. The false positives of this choice are tolerable in the total accuracy.

5 Experimental Study

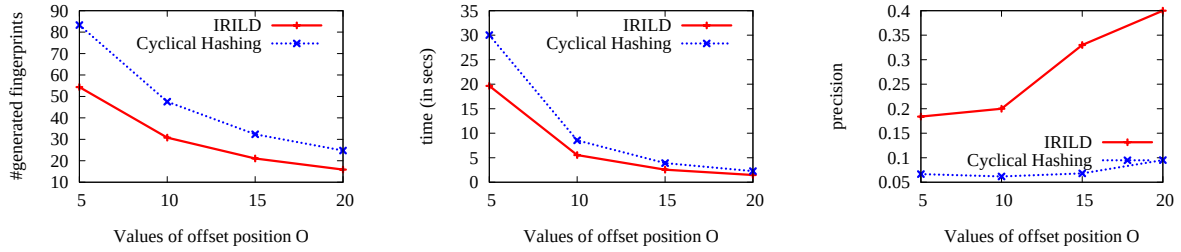
To evaluate the efficiency and effectiveness of IRILD, we implemented it in Python 2.5 and conducted experiments with the Enron Email Dataset [1], where we randomly chose 50 emails from the "Inbox" of 10 employees as confidential documents, and used 200 random emails from their "Sent_Items" to test the accuracy of the method. We considered textual information in the Enron's website (an instance of January 2001 [2]) as public information. For comparison purpose, we compared IRILD to the popular approach that employed only cyclical hashing without the removal of public and common phrases in fingerprint generation. In the experiments, the default values of C, O, and K are respectively set to 20, 10, and 4.

We evaluated the performance of IRILD and the popular approach in three aspects: (i) the cost of fingerprint generation in terms of processing time and the number of fingerprints, (ii) the cost of information leak detection in terms of processing time, and (iii) the accuracy of the



(a) on the number of fingerprints (b) on the leak detection time (c) on the precision

Figure 3: Effect of varying the string length C, keeping O = 10



(a) on the number of fingerprints (b) on the leak detection time (c) on the precision

Figure 4: Effect of varying the offset position O, keeping C = 20

results in terms of *recall* and *precision*. We define *recall* as the ratio of the total number of leaked cases that are correctly detected over the total number of leaked cases that exist in the testing data set and *precision* as the ratio of the total number of leaked cases that are correctly detected over the total number of leaked cases that are returned (i.e., including incorrectly detected cases – false positive cases). Note that to compute *recall* and *precision*, we manually looked at testing documents to extract all possible leaked cases.

5.1 Cost of fingerprint generation

We first evaluated the cost of fingerprint generation for confidential documents (emails in our case). As expected, IRILD incurred a processing time around 2 to 3 times longer than the basic approach, due to the extra time of submitting queries to Google to determine the popularity of candidate phrases. While this extra time can be reduced by employing PlanetLab [3] to submit queries concurrently, since fingerprint generation is done offline whenever a confidential document is submitted to the system, it does not affect the efficiency and effectiveness of IRILD.

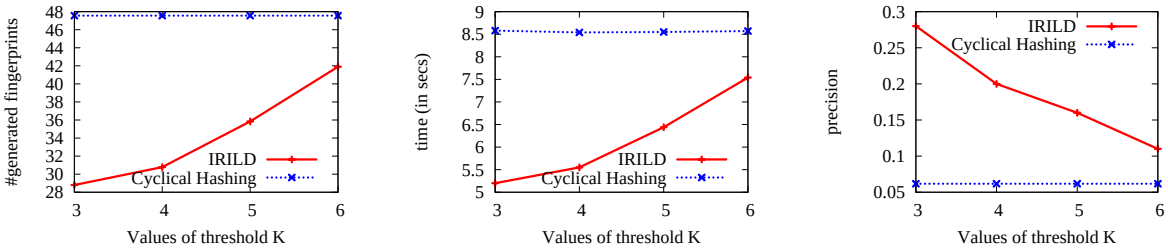
The average number of fingerprints generated by IRILD and the basic approach when varying C and O are shown in Figure 3(a) and Figure 4(a) respectively. The results show that by removing public and common phrases from fingerprint generation, IRLID significantly reduced the number of generated fingerprints compared to the basic approach. In particular, as shown in Figure 3(a), the percentage difference in the number of generated fingerprints between cyclical hashing and IRILD varied from 11% when C = 30 to 55% when C = 15. On the other hand, as in Figure 4(a), IRILD generated 35% less fingerprints in all cases of varying O. It is interesting to observe that while the number of generated fingerprints in cyclical hashing does not depend on the value of C, in case of IRILD, the bigger the value of C is, the more the number of generated fingerprints is.

5.2 Cost of information leak detection

In this experiment, we measured the processing time required to detect information leak from testing documents. Figure 3(b) and Figure 4(b) show the processing time of information leak detection when varying C and O. As expected, since the number of fingerprints in IRILD was less than those in the basic approach, IRILD took a smaller number of comparisons, and hence it incurred a faster processing time. Actually, if we make a comparison between Figure 3(a) and Figure 3(b) as well as Figure 4(a) and Figure 4(b), the similarity between them show a strong correlation between the number of the generated fingerprints and the information leak detection time.

5.3 Accuracy

As discussed earlier, we evaluated the accuracy of IRILD by measuring *recall* and *precision*. It is interesting to notice that the *recalls* of IRILD and the basic approach were always the same. That is because both approaches



(a) on the number of fingerprints (b) on the leak detection time (c) on the precision

Figure 5: Effect of varying the threshold K, keeping $C = 20$ and $O = 10$

employ the same techinque for indexing sensitive information and searching for information leaks. On the other hand, since the advantage of IRILD compared to the basic approach is in the capability of reducing false positives caused by common phrases, as shown in Figure 3(c) and Figure 4(c), IRILD achieved a much better precision compared to the basic approach.

5.4 Effect of varying the threshold K

So far we had K fixed at 4. In this experiment, we evaluated the effect of varying K from 3 to 6 on IRILD (note that the change of K does not affect the popular approach). The experimental results displayed in Figure 5 show that with the increasing of K, the number of generated fingerprints as well as the processing cost increased. It is because when K increased, we put a higher boundary for phrases to be considered common phrases, and hence less common phrases were identified and removed. The consequence of having less identified common phrases was a decrease in the precision of the method.

6 Conclusion

In this paper, we introduced IRILD, a novel, information retrieval based method for detecting information leaks. The core idea of IRILD is to identify and remove public phrases (found in public documents) and common phrases (identified by checking the number of results returned from Google when querying the phrases). Our experiments proved that compared to the popular cyclical hashing approach, our method is both quicker in detecting information leak and more accurate.

References

[1] *Enron Email Dataset*. `http://www.cs.cmu.edu/~enron/`.

[2] *Internet Archive*. `http://www.archive.org/`.

[3] *Planetlab: An open platform for developing, deploying, and accessing planetary-scale services*. `www.planet-lab.org`.

[4] CAPIZZI, R., LONGO, A., VENKATAKRISHNAN, V. N., AND SISTLA, A. P. Preventing information leaks through shadow executions. In *ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 322–331.

[5] CARVALHO, V. R., AND COHEN, W. W. Preventing information leaks in email. In *SDM* (2007).

[6] CHOW, R., GOLLE, P., AND STADDON, J. Detecting privacy leaks using corpus-based association rules. In *KDD* (2008), pp. 893–901.

[7] DAVISON, B. D., DESCHENES, D. G., AND LEWANDA, D. B. Finding relevant website queries. In *In Proc. of WWW* (2003).

[8] INSTITUTE, S. *Understanding and Selecting a Data Loss Prevention Solution*, 2009. White Paper.

[9] INSTITUTE, T. P. *2006 Annual Study: Cost of a Data Breach*, 2006. White Paper.

[10] JUNG, J., SHETH, A., GREENSTEIN, B., WETHERALL, D., MAGANIS, G., AND KOHNO, T. Privacy oracle: a system for finding application leaks with black box differential testing. In *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security* (New York, NY, USA, 2008), ACM, pp. 279–288.

[11] LABS, P. T. *Information Leak Prevention Accuracy and Security Tests*, 2006. White Paper.

[12] NAKOV, P., AND HEARST, M. Using the web as an implicit training set: application to structural ambiguity resolution. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (Morristown, NJ, USA, 2005), Association for Computational Linguistics, pp. 835–842.

[13] PARNO, B., MCCUNE, J. M., WENDLANDT, D., ANDERSEN, D. G., AND PERRIG, A. Clamp: Practical prevention of large-scale data leaks. In *SP '09: Proceedings of the 2009 30th IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2009), IEEE Computer Society, pp. 154–169.

[14] STADDON, J., GOLLE, P., AND ZIMNY, B. Web-based inference detection. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium* (Berkeley, CA, USA, 2007), USENIX Association, pp. 1–16.