ACHIEVING SECURITY ON SECURED DATA IN NETWORK SECURITY

Sumitha. N [1],Kanimozhi.G [2],

[1] Assistant Professor, Department of IT,
Idhaya Engineering College for Women, Chinnasalem,
Sumithabtech91@gmail.com

[2]PG Scholar, Department of CSE,
Idhaya Engineering College for Women, Chinnasalem,
Kanimozhicse4@gmail.com

ABSTRACT

The leak of sensitive data on computer systems poses a serious threat to organizational security. Statistics show that the lack of proper encryption on files and communications due to human errors is one of the leading causes of data loss. Organizations need tools to identify the exposure of sensitive data by screening the content in storage and transmission, i.e., to detect sensitive information being stored or transmit-ted in the clear. However, detecting the exposure of sensitive information is challenging due to data transformation in the content. Transformations (such as insertion and deletion) result in highly unpredictable leak patterns. In this paper, we utilize sequence alignment techniques for detecting complex data-leak patterns. Our algorithm is designed for detecting long and inexact sensitive data patterns. This detection is paired with a comparable sampling algorithm, which allows one to compare the similarity of two separately sampled sequences. Our system achieves good detection accuracy in recognizing transformed leaks. We implement a parallelized version of our algorithms in graphics processing unit that achieves high analysis throughput. We demonstrate the high multithreading scalability of our data leak detection method required by a sizable organization.

Index Terms— Data leak detection, content inspection, sampling, alignment, dynamic programming, parallelism.

I. INTRODUCTION

REPORTS show that the number of leaked sensitive data records has grown 10 times in the last 4 years, and it reached a record high of 1.1 billion in 2014 [3]. A significant portion of the data leak incidents are due to human errors, for example, a lost or stolen laptop containing unencrypted sensitive files, or transmitting sensitive data without using end-to-end encryption such as PGP. A recent Kaspersky Lab survey shows that accidental leak by staff is the leading cause for internal data leaks in corporates [4]. The data-leak risks posed tool searches for the occurrences of *plaintext sensitive data* in the *content of files or network traffic*. It alerts users and administrators of the identified data exposure vulnerabilities. For example, an organization's mail server can inspect the content of outbound email messages searching for sensitive data appearing in unencrypted messages.

Data leak detection differs from the anti-virus (AV) scanning (e.g., scanning file systems for malware signatures) or the network intrusion detection systems (NIDS) (e.g., scanning traffic payload for malicious patterns) [5]. AV and NIDS typically employ automata-based string matching (e.g., Aho-Corasick [6], Boyer-Moore [7]), which match static or regular patterns. However, data leak detection imposes new security requirements and algorithmic challenges:

1) *Data Transformation:* The exposed data in the content may be unpredictably transformed or modified by users or applications, and it may no longer be identical to the original sensitive data, e.g., insertions of metadata or formatting tags, substitutions of characters, and data truncation (partial data leak). Thus, the detection algo-rithm needs to recognize different kinds of sensitive data variations.

2) *Scalability:* The heavy workload of data leak screening is due to two reasons.

   Long Sensitive Data Patterns: The sensitive data (e.g., customer information, documents, source code) can be of arbitrary length (e.g., megabytes).

   Large Amount of Content: The detection needs to rapidly screen content (e.g., gigabytes to

terabytes). Traffic scanning is more time sensitive than storage scanning, because the leak needs to be discovered

before the message is transmitted.

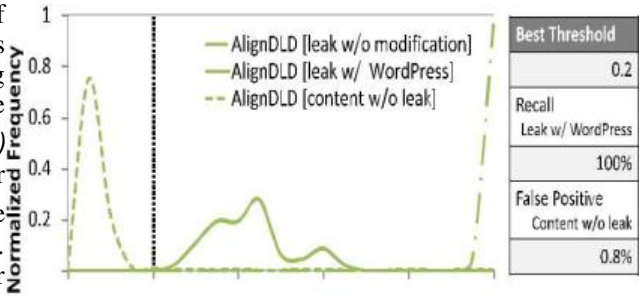
Directy applying automata-based string matching (e.g., [6], [8], [9]) to data leak detection is inappropriate

and inefficient, because automata are not designed to support unpredictable and arbitrary pattern variations. In data leak detection scenarios, the transformation of leaked data (in the description of regular expression) is unknown to the detection method. Creating comprehensive automata models covering all possible variations of a pattern is infeasible, which leads to $O(2^n)$ space complexity (for deterministic finite automata) or $O(2^n)$ time complexity (for nondeterministic finite automata) where n is the number of automaton states. Therefore, automata approaches cannot be used for detecting long and transformed data leaks.

Existing data leak detection approaches are based on set intersection. Set intersection is performed on two sets of n-grams, one from the content and one from sensitive data. The set intersection gives the amount of sensitive n-grams appearing in the content. The method has been used to detect

similar documents on the web [10], shared malicious traffic patterns [11], malware [12], as well as email spam [13]. The advantage of n-grams is the extraction of local features of a string, enabling the comparison to tolerate discrepancies. Some advanced versions of the set intersection method utilize Bloom filter, e.g., [14], which trades accuracy for space complexity and speed. Shu and Yao extended the standard use of n-grams and introduced data-leak detection as a service. They proposed the first solution for detecting accidental data leak with semi-honest providers [15].

However, set intersection is *orderless*, i.e., the ordering of shared n-grams is not analyzed. Thus, set-based detection generates undesirable false alerts, especially when n is set to a small value to tolerant data transformation. In addition, set intersection cannot effectively characterize the scenario when partial data is leaked, which results in false negatives. Therefore, none of the existing techniques is adequate for detecting transformed data leaks. Our solution to the detection of transformed data leaks is a sequence alignment algorithm, executed on the sampled sen-sitive data sequence and the sampled content being inspected. The alignment produces scores indicating the amount of sensitive data contained in the content. Our alignment-based solution measures the order of n-grams. It also handles arbi-trary variations of patterns without an explicit specification of all possible variation patterns. Experiments show that our alignment method substantially outperforms the set intersec-tion method in terms of detection accuracy in a multitude of transformed data leak scenarios.



II. Related Work

Existing commercial data leak detection/prevention solutions include Symantec DLP [14], IdentityFinder [18], GlobalVelocity [19], and GoCloudDLP [20]. GlobalVelocity uses FPGA to accelerate the system. All solutions are likely based on n-gram set intersection. IdentityFinder searches file systems for short patterns of numbers that may be sensitive (e.g., 16-digit numbers that might be credit card numbers). It does not provide any in-depth similarity tests. Symantec DLP is based on n-grams and Bloom filters. The advantage of Bloom filter is space saving. However, as explained in the introduction, Bloom filter membership testing is based on unordered n-grams, which generates coincidental matches and false alarms. Bloom filter configured with a small number of hash functions has collisions, which introduce additional unwanted false positives.

Network intrusion detection systems (NIDS) such as Snort [21] and Bro [22] use regular expression to perform string matching in deep packet inspection [23]. Nondetermin-istic finite automaton (NFA) with backtracking requires $O(2^n)$ time and $O(n)$ space, where n is the number of automaton states. Deterministic finite automaton (DFA) has a time com-plexity of $O(n)$ and a space complexity of $O(2^n)$ when used with quantification. Quantification is for expressing optional characters and multiple occurrences in a pattern. DFA's

space complexity can be reduced by grouping similar patterns into one automaton [24], reducing the number of edges [25], [26]. These improvements provide a coefficient level of speedup.

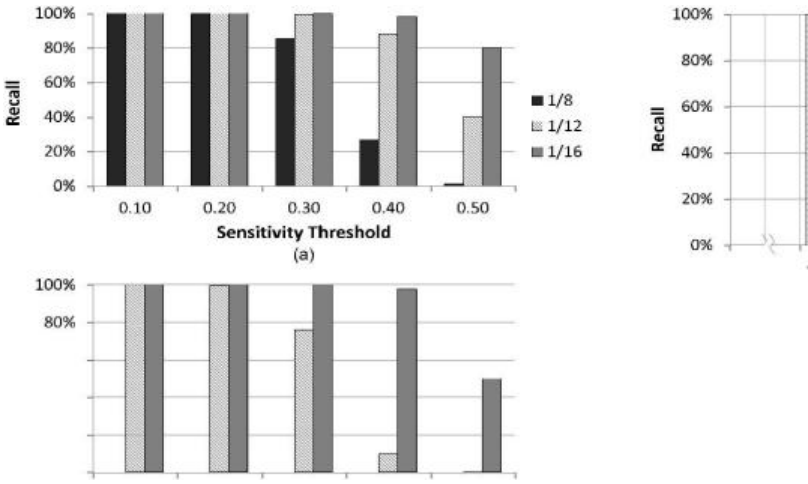
However, existing string matching approaches based on DFA or NFA cannot automatically match arbitrary and unpre-dictable pattern variations. Modified data leak instances cannot be matched or captured if the variation is not manually

specified as a matching pattern. Enumerating all potential vari-ation patterns takes exponential time and space with respect to the length of the pattern. Therefore, it is impractical.

In comparison, our sequence alignment solution covers all possible pattern variations in long sensitive data without explicitly specifying them. Another drawback of automata is that it yields binary results. In comparison, alignment pro-vides precise matching scores and allows customized weight functions. Our alignment gives more accurate detection than approximate string matching (e.g., [27], [28]).

Alignment algorithms have been widely used in computational biology applications, and features such as privacy-preserving sequence matching have been studied [29]. In security literature, NetDialign based on the well-known Dialign algorithms is proposed for network privacy [30]. It performs differential testing among multiple traffic flows. Kreibich and Crowcroft presented an alignment algorithm for traffic intrusion detection systems such as Bro [31]. It is a variant of Jacobson-Vo alignment that calculates the longest common subsequence with the minimum number of gaps. Researchers in [32] reported the use of dynamic programming for computing the similarity of network behaviors and presented a technique to handle behavioral sequences with differing sampling rates. Masquerade attacks in the context of user command sequences can be detected with semi-global sequence alignment techniques [33], [34].

Our data leak detection differs from the above network privacy and IDS problems, and it has new requirements as we have explained in the introduction. Our alignment performs complex inferences needed for aligning sampled sequences, and our solution is also different from fast non-sample align-ment in bioinformatics, e.g., BLAST [35].



III. Models and Overview

In our data leak detection model, we analyze two types of sequences: sensitive data sequence and content sequence.

[1] *Content sequence* is the sequence to be examined for leaks. The content may be data extracted from file sys-tems on personal computers, workstations, and servers; or payloads extracted from supervised network channels (details are discussed below).

[2] *Sensitive data sequence* contains the information (e.g., customers' records, proprietary documents) that needs to be protected and cannot be exposed to unauthorized parties. The sensitive data sequences are known to the

analysis system.

In this paper, we focus on detecting inadvertent data leaks, and we assume the content in file system or network traffic (over supervised network channels) is available to the inspec-tion system. A supervised network channel could be an unen-crypted channel or an encrypted channel where the content in it can be extracted and checked by an authority. Such a channel is widely used for advanced NIDS where MITM (man-in-the-middle) SSL sessions are established instead of normal SSL sessions [47]. We do not aim at detecting stealthy data leaks that an attacker encrypts the sensitive data secretly before leaking it. Preventing intentional or malicious data leak, especially encrypted leaks, requires

different approaches and remains an active research problem [48].

In our current security model, we assume that the analysis system is secure and trustworthy. Privacy-preserving data-leak detection can be achieved by leveraging special protocols and computation steps [49]. It is another functionality of a detection system, and the discussion is not within the scope of this paper.

IV. Comparable Sampling

In this section, we define the sampling requirement needed in data leak detection. Then we present our solution and its analysis.

A. Definitions

One great challenge in aligning sampled sequences is that the sensitive data segment can be exposed at an arbitrary position in a network traffic stream or a file system. The sampled sequence should be deterministic despite the starting and ending points of the sequence to be sampled. More-over, the leaked sensitive data could be inexact but similar to the original string due to unpredictable transformations. We first define substring and subsequence relations in Definition 1 and Definition 2. Then we define the capa-bility of giving comparable results from similar strings in Definition 3.

Definition 1 (Substring): A substring is a consecutive segment of the original string.

If x is a substring of y, one can find a prefix string

(denoted by y_p) and a suffix string (denoted by y_s) of y,

so that y equals to the concatenation of y_p, x, and y_s.

y_p and y_s could be empty.

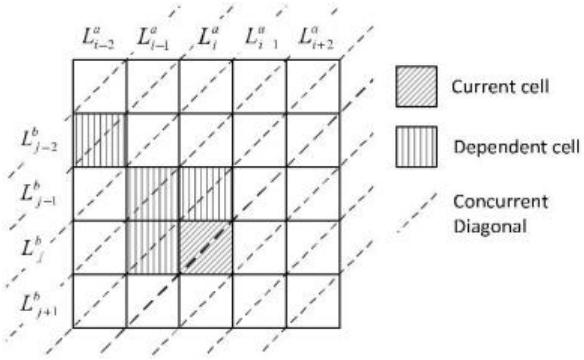
Definition 2 (Subsequence): Subsequence is a generaliza-tion of substring that a subsequence does not require its items to be consecutive in the original string.

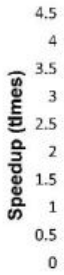
One can generate a subsequence of a string by removing items from the original string and keeping the order of the remaining items. The removed items can be denoted as gaps in the subsequence, e.g., `lo-e` is a subsequence of `flower` (– indicates a gap).

Definition 3 (Comparable Sampling): Given a string x and another string y that x is similar to a substring of y accord-ing to a similarity measure M, a comparable sampling on x and y yields two subsequences x (the sample of x) and y (the sample of y), so that x is similar to a substring of y according to M.

If we restrict the similarity measure M in Definition 3 to *identical relation*, we get a specific instance of comparable sampling in Definition 4.

Definition 4 (Subsequence-Preserving Sampling): Given x as a substring of y, a subsequence-preserving sampling on x and y yields two subsequences x (the sample of x) and y (the sample of y), so that x is a substring of y .





B. Scalability

In this experiment, we parallelize Sampling and Alignment in AlignDLD through various numbers of threads. The times of speedup in analyzing *A. Enron* dataset are reported in Figure 7. The results show the close-to-ideal scalability for Sampling when parallelized onto an increasing number of threads. Our unoptimized multithreaded CPU Alignment scales up less well in comparison, which we attribute to poor memory cache utilization. The score matrices are too large to fit into the cache for some alignments. The interaction between threads may evict reusable data from the cache. These operations in turn may cause cache misses. An optimized program should possess better data locality to minimize cache misses, and the optimization can be achieved in real-world detection products. [9] discussed about creating Obstacles to Screened networks. In today's technological world, millions of individuals are subject

to privacy threats. Companies are hired not only to watch what you visit online, but to infiltrate the information and send advertising based on your browsing history.

B. Detecting Modified Leaks

We evaluate three types of modifications: *i)* real-world pervasive substitution by `WordPress`, *ii)* random pervasive substitution, and *iii)* truncated data (localized modifications).

1) Pervasive Substitution: We test *AlignDLD* and *Coll-Inter* on content extracted from three kinds of network traffic.

Due to the limited bandwidth between CPU and GPU, data transfer is the bottleneck of our GPU implementation and dominates the execution time. A common strategy to solve the issue is to overlap data transfer and kernel execution or to batch the GPU input [61]. Another possible approach from the hardware perspective is to use a CPU-GPU integrated platform, such as AMD APU or Intel MIC, which benefits from the shared memory between CPU and GPU [62].

CONCLUSIONS AND FUTURE WORK

We presented a content inspection technique for detecting leaks of sensitive information in the content of files or network traffic. Our detection approach is based on aligning two sampled sequences for similarity comparison. Our experimen-tal results suggest that our alignment method is useful for detecting multiple common data leak scenarios. The parallel versions of our prototype provide substantial speedup and indicate high scalability of our design. For future work, we plan to explore data-movement tracking approaches for data leak prevention on a host.

REFERENCES

[1] X. Shu, J. Zhang, D. Yao, and W.-C. Feng, "Rapid and parallel content screening for detecting transformed data exposure," in *Proc. 3rd Int. Workshop Secur. Privacy Big Data (BigSecurity)*, Apr./May 2015, pp. 191–196.

[2] X. Shu, J. Zhang, D. Yao, and W.-C. Feng, "Rapid screening of transformed data leaks with efficient algorithms and parallel comput-ing," in *Proc. 5th ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, San Antonio, TX, USA, Mar. 2015, pp. 147–149.

[3] (Feb. 2015). *Data Breach QuickView: 2014 Data Breach Trends*. [Online]. Available: https://www.riskbasedsecurity.com/reports/2014-YEDataBreachQuickView.pdf, accessed Feb. 2015.

[4] Kaspersky Lab. (2014). *Global Corporate IT Security Risks*. [Online]. Available: http://media.kaspersky.com/en/business-security/Kaspersky_Global_IT_Security_Risks_Survey_report_Eng_final.pdf

[5] L. De Carli, R. Sommer, and S. Jha, "Beyond pattern matching: A concurrency model for stateful deep packet inspection," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1378–1390.

[6] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Commun. ACM*, vol. 18, no. 6, pp. 333–340, Jun. 1975.

[7] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Commun. ACM*, vol. 20, no. 10, pp. 762–772, Oct. 1977.

[8] S. Kumar, B. Chandrasekaran, J. Turner, and G. Varghese, "Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia," in *Proc. 3rd ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, 2007, pp. 155–164.

[9] Christo Ananth, P.Muppidathi, S.Muthuselvi, P.Mathumitha, M.Mohaideen Fathima, M.Muthulakshmi, "Creating Obstacles to Screened networks", International Journal of Advanced Research in Biology, Ecology, Science and Technology (IJARBEST), Volume 1,Issue 4,July 2015, pp:10-14

[10] A. Z. Broder, "Identifying and filtering near-duplicate documents," in *Proc. 11th Annu. Symp. Combinat. Pattern Matching*, 2000, pp. 1–10.

[11] M. Cai, K. Hwang, Y.-K. Kwok, S. Song, and Y. Chen, "Collaborative Internet worm containment," *IEEE Security Privacy*, vol. 3, no. 3, pp. 25–33, May/Jun. 2005.

[12] J. Jang, D. Brumley, and S. Venkataraman, "BitShred: Feature hashing malware for scalable triage and semantic analysis," in *Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS)*, 2011, pp. 309–320.

[13] K. Li, Z. Zhong, and L. Ramaswamy, "Privacy-aware collaborative spam filtering," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 5, pp. 725–739, May 2009.

[14] Symantec. (2015). *Symantec Data Loss Prevention*. [Online]. Available: http://www.symantec.com/data-loss-prevention, accessed Feb. 2015.

[15] X. Shu and D. Yao, "Data leak detection as a service," in *Proc. 8th Int. Conf. Secur. Privacy Commun. Netw. (SecureComm)*, Padua, Italy, Sep. 2012, pp. 222–240.