Rapid Screening of Transformed Data Leaks with Efficient Algorithms and Parallel Computing[*]

Xiaokui Shu, Jing Zhang, Danfeng (Daphne) Yao, Wu-Chun Feng
Department of Computer Science
Virginia Tech
Blacksburg, VA, USA
{subx, zjing14, danfeng, feng}@cs.vt.edu

ABSTRACT

The leak of sensitive data on computer systems poses a serious threat to organizational security. Organizations need to identify the exposure of sensitive data by screening the content in storage and transmission, i.e., to detect sensitive information being stored or transmitted in the clear. However, detecting the exposure of sensitive information is challenging due to data transformation in the content. Transformations (such as insertion, deletion) result in highly unpredictable leak patterns. Existing automata-based string matching algorithms are impractical for detecting transformed data leaks, because of its formidable complexity when modeling the required regular expressions. We design two new algorithms for detecting long and transformed data leaks. Our system achieves high detection accuracy in recognizing transformed leaks compared to the state-of-the-art inspection methods. We parallelize our prototype on graphics processing unit and demonstrate the strong scalability of our detection solution required by a sizable organization.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*security and protection*; D.1.3 [**Programming Techniques**]: Concurrent Programming—*parallel programming*

Keywords

Data leak detection; content inspection; algorithm; sampling; alignment; dynamic programming; parallelism

1. INTRODUCTION

The number of leaked records on personal computers and organization networks increases dramatically in the last years from 95 million in 2010 to 822 million in 2013 [3]. A typical approach to minimize the exposure of sensitive data is to identify all occurrences of cleartext sensitive data in storages or communications. The detection system alerts administrators of any sensitive data exposure discovered in file systems or supervised network channels. Leaks can then be identified according to the sensitive data storage and sharing policy. Different from pattern matching techniques employed in anti-virus and intrusion detection systems, data leak detection imposes new security requirements and algorithmic challenges:

1. *Data transformation.* The exposed data in the content may be transformed or modified by users or applications, so 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. Thus, the detection algorithm needs to recognize variations of sensitive data patterns.
2. *Scalability.* The heavy workload of data leak screening is due to *long sensitive data patterns* and the *large amount of content.* Sensitive data (e.g., documents, source code) can be of arbitrary length (e.g., megabytes). Some types of contents may need to be scanned in a timely manner (e.g., traffic scanning).

Automata-based string matching are widely used in anti-virus and network intrusion detection systems (NIDS) where the patterns to search for are static or can be characterized by regular expressions [1]. 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. Therefore, automata approach cannot be used for detecting long and transformed data leaks.

Existing data leak detection approaches are largely based on set intersection. Set intersection between the content fragment set and sensitive data fragment set tells the amount of sensitive data fragments appearing in the content[1]. However, set intersection is *orderless*, i.e., the order information of fragments is abandoned. Thus, set-based detection generates undesirable false alerts. In addition, set intersection cannot effectively measure the likelihood of a leak when partial data is leaked. Therefore, none of the existing techniques is adequate for detecting transformed data leaks.

The key of our solution to the detection of transformed data leaks is a new sequence alignment algorithm. *The alignment is between the sampled sensitive data sequence and the sampled content being inspected.* The alignment produces a

[*]This work has been supported in part by Security and Software Engineering Research Center (S^{2}ERC), a NSF sponsored multi-university Industry/University Cooperative Research Center (I/UCRC).

CODASPY'15, March 2–4, 2015, San Antonio, Texas, USA.
ACM 978-1-4503-3191-3/15/03.
http://dx.doi.org/10.1145/2699026.2699130.

[1]Typical units in a set are n-grams of a string, which preserves local features of a string and tolerates discrepancies.

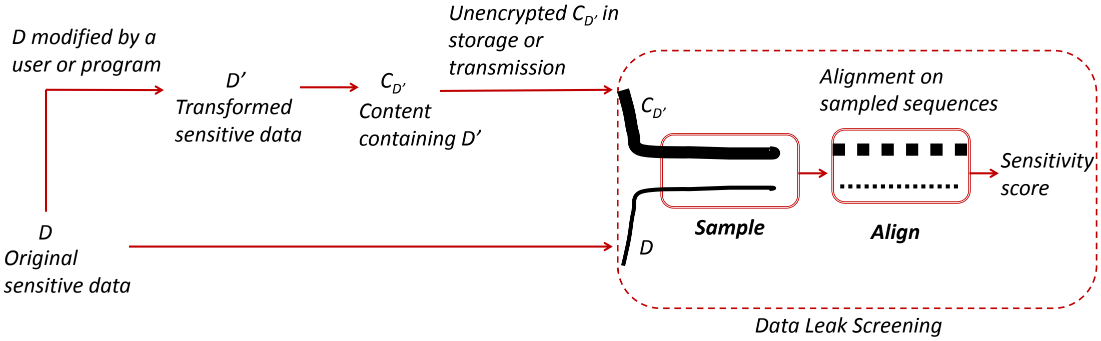

D modified by a user or program
D′
Transformed sensitive data
$C_{D'}$
Content containing D′
Unencrypted $C_{D'}$ in storage or transmission
$C_{D'}$
Alignment on sampled sequences
Sensitivity score
Sample
Align
D
Original sensitive data
D
Data Leak Screening

Figure 1: A schematic drawing showing the two types of sequences in our model, their relations, and the workflow of our detection.

score indicating the amount of sensitive data contained in the content. Our alignment-based solution measures the order of n-grams and handles arbitrary variations of patterns without an explicit specification of all possible variations.

In order to deal with the scalability issue, we design a pair of algorithms to perform alignment. Our solution consists of a *comparable* sampling algorithm and a *sampling oblivious* alignment algorithm. Our sampling algorithm samples both content and sensitive data sequences. It satisfies the *comparable sampling* property that the similarity of two sequences is preserved through sampling, and the samples are meaningful to be aligned. Our solution aligns sampled sequences to infer the similarity between the original sequences before sampling. The special *sampling oblivious* property differentiates our algorithm from existing ones. Experiments show that our solution achieves accurate detection with low false positive and false negative rates. It substantially outperforms set-based methods in terms of detection accuracy.

We design the pair of algorithms to be efficiently parallelized. We parallelize our prototype on a GPU, which achieves nearly 50 times of speedup over the CPU version. Our prototype reaches 400Mbps analysis throughput. This performance can support the rapid security scanning of storage and communication required by a sizable organization.

2. MODELS AND OVERVIEW

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

- *Content sequence* is the sequence to be examined for leaks. The content may be data extracted from file systems on workstations and servers, or payloads extracted from supervised network channels[2].
- *Sensitive data sequence* is the information, e.g., proprietary documents, that needs to be protected and cannot be exposed to unauthorized parties.

Both the content and the sensitive data sequences are known to the analysis system. A data leak is detected when the detection system finds a piece of sensitive data in the content sequence (Fig. 1), but the appearance is not allowed in the sensitive data storage and sharing policy. We assume that the analysis system is secure and trustworthy. Thus, the sensitive data sequence is secure during the data leak analysis. The two assumptions can be removed when our alignment is performed utilizing secure multi-party computation or other privacy-preserving techniques [2, 4]. We do not aim at detecting stealthy data leaks that an attacker encrypt the sensitive data by herself before leaking it.

2.1 Technical Challenges

High detection specificity. In our data-leak detection model, high specificity refers to the ability to distinguish true leaks from coincidental matches, which can cause false alarms. Existing set-based detection is orderless, where the order of matched patterns (n-grams) is ignored. Orderless detection can result in false positives as shown below.

Sensitive data	`abcdefg`
3-grams of the sensitive data	`abc`, `bcd`, `cde`, `def`, `efg`
Content stream (false positive)	`...efg...cde...abc...`

Pervasive and localized modification. Sensitive data could be modified before it is leaked out. The modification can occur throughout a sequence (pervasive modification). The modification can also only affect a local region (local modification). We describe some modification examples:

- Character replacement, e.g., `WordPress` replaces every space character with a `+` in HTTP POST requests.
- String insertion: HTML tags inserted throughout a document for formatting or embedding objects.
- Data truncation or partial data leak, e.g., one page of a two-page sensitive document is transmitted.

2.2 Overview of Our Approach

Our work presents an efficient sequence comparison technique needed for analyzing a large amount of content for sensitive data exposure. We illustrate our workflow in Fig. 1. Our detection approach consists of a special sampling algorithm and a corresponding alignment algorithm working on preprocessed n-grams of sequences. The pair of algorithms computes a quantitative similarity score between sensitive data and content. Local alignment, as opposed to global alignment, is used to identify similar sequence segments, enabling the detection of partial data leaks.

Our workflow includes EXTRACTION, PREPROCESSING, SAMPLING, ALIGNMENT, and DECISION operations. The EXTRACTION operation collects content. The PREPROCESSING operation prepares the sequences of n-grams for both the content and sensitive data. The SAMPLING operation generates samples from both sensitive data and content sequences. The ALIGNMENT operation performs local alignment between the two sampled sequences to compute their

[2] Such channels are widely used for advanced NIDS where MITM (man-in-the-middle) SSL sessions are employed to handle encryption.

similarity. Finally, the DECISION operation confirms and reports leaks according to the sensitive data sharing policy.

3. IMPLEMENTATION AND EVALUATION

We evaluate the accuracy of our solution with several large datasets under real-world data leak scenarios[3]. We implement a single-threaded prototype (referred to as *AlignDLD* system) and a collection intersection method as a baseline. Both systems are written in `C++`, compiled using `g++` 4.7.1 with flag `-O3`. We also provide two parallel versions of our prototype for performance demonstration.

- *AlignDLD:* our sample-and-align data leak detection method with sampling parameters $N = 10$ and $|w| = 100$. 3-grams and 32-bit Rabin's fingerprints are used.
- *Coll-Inter:* a data leak detection system based on collection intersection[4], which is widely adopted by commercial tools such as GlobalVelocity and GoCloudDLP. 8-grams and 64-bit Rabin's fingerprints are used, which is standard with collection intersection.

We conduct all experiments in a virtualized network using VirtualBox. The detection system is deployed on the gateway that connects the virtual local network and the Internet. Simple leaks are performed using web and FTP file transmission. Publishing services such as `WordPress` modifies sensitive data when it is leaked. We test our detection system using two dataset: *A. Enron dataset* consisting of 2.6GB emails, and *B. MiscNet* consisting of 500MB Internet traffic dump with various kinds of Internet traffic.

We define the *sensitivity* $\mathbb{S} \in [0, 1]$ of the content sequence in Formula 1. It indicates the similarity of sensitive data D and content $C_{D'}$ with respect to their sequences $\mathcal{S}^a$ and $\mathcal{S}^b$ after PREPROCESS. ξ is the maximum score in the alignment, and r is the reward for one-unit match in the alignment.

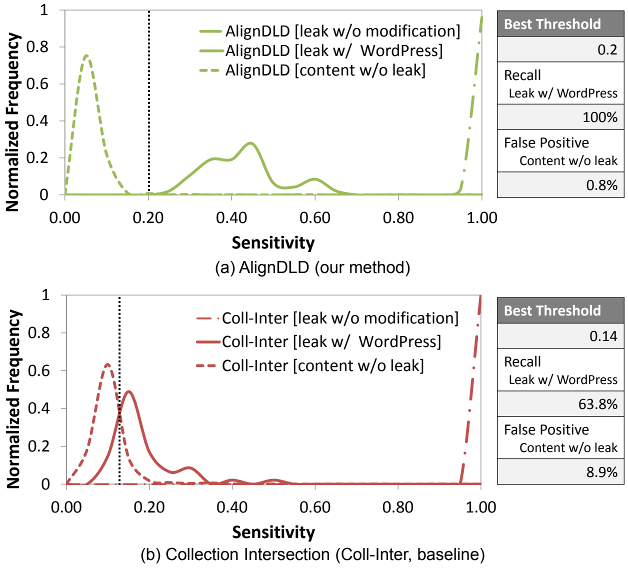
$$\mathbb{S} = \frac{\xi}{r \times \min\left(|\mathcal{S}^a|, |\mathcal{S}^b|\right)} \tag{1}$$

3.1 Detecting Modified Leaks

We conduct detection accuracy experiments on three types of data leaks listed below.

1. Content without any leak, i.e., the content does not contain any sensitive data.
2. Content with unmodified leak, i.e., sensitive data appearing in the content is not modified.
3. Content with modified leaks caused by `WordPress`, which substitutes every space with "`+`" in the content.

We evaluate and compare AlignDLD and Coll-Inter. We present the distributions of all sensitivity values in Fig. 2. Both methods perform as expected in the scenarios of no-leak and unmodified leak. The solid lines in Figure 2 represent the detection results of leaks with `WordPress` modifications. Our AlignDLD method in Fig. 2 (a) gives much higher sensitivity scores to the transformed data leak than the Coll-Inter method. With a threshold of 0.2, **all the email messages with transformed leaks are detected and reported.** In contrast, the collection intersection method in Fig. 2 (b) has a significant overlap between messages with no leak and messages with transformed leaks. Its accuracy is much lower than that of our method, e.g., 63.8% recall and a 10 times higher false positive rate.



(a) AlignDLD (our method)

(b) Collection Intersection (Coll-Inter, baseline)

Figure 2: Detection comparison of leak through `WordPress` using AlignDLD (a) and Coll-Inter (b).

3.2 Parallelization and Scalability

In order to achieve high analysis throughput, we parallelize our algorithms on CPU as well as on general-purpose GPU platforms. The multithreaded CPU version is written in C, compiled using gcc 4.4.5 with `flag -O2`. The GPU version is written in CUDA compiled using CUDA 4.2 with `flag -O2 -arch sm 20` and NVIDIA driver v295.41. We deploy our prototypes on a hybrid CPU-GPU machine equipped with an Intel Core i5 2400 and an NVIDIA Tesla C2050 GPU (Fermi architecture with 448 GPU cores).

Our GPU detection prototype achieves over 40 times of speedup over the CPU version on both content datasets. The prototype achieves a throughput of over 400Mbps against dataset *B*. This throughput is comparable to that of a moderate commercial firewall.

4. CONCLUSIONS

Despite the commercial success of data leak software and appliances, existing solutions based on set intersection have serious security drawbacks. We present new and sophisticated alignment-based algorithms to improve the accuracy detecting data leaks, e.g., high specificity (i.e., low false alarm rate). Our extensive experimental evaluations with real-world data and leak scenarios confirm that our method has much higher precision in detecting transformed data leaks than the state-of-the-art set intersection method.

5. REFERENCES

[1] A. V. Aho and M. J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, 1975.

[2] F. Liu, X. Shu, D. Yao, and A. R. Butt. Privacy-preserving scanning of big content for sensitive data exposure with MapReduce. In *Proceedings of ACM CODASPY*, 2015.

[3] RiskBasedSecurity. Data breach quickview: An executive's guide to 2013 data breach trends, Feb 2014.

[4] X. Shu and D. Yao. Data leak detection as a service. In *Proceedings of SecureComm*, 2012.

[3] We only present the most important experiments due to the limited space.

[4] Set and collection intersections are used interchangeably.