LETTER *Special Section on Security, Privacy and Anonymity in Computation, Communication and Storage Systems*

Doc-Trace: Tracing Secret Documents in Cloud Computing via Steganographic Marking

Sang-Hoon CHOI[†], **Joobeom YUN**[†], *and* **Ki-Woong PARK**[†a)], ***Members***

SUMMARY The secret document leakage incidents have raised awareness for the need to better security mechanisms. A leading cause of the incidents has been due to accidental disclosure through via removable storage devices. As a remedy to the issue, many organizations have been employing private cloud platform or virtual desktop infrastructure (VDI) to prevent the leakage of the secret documents. In spite of the various security benefits of cloud-based infrastructure, there are still challenges to prevent the secret document leakage incidents. In this paper, we present a novel scheme, called *Doc-Trace*, to provide an end-to-end traceability for the secret documents by inserting steganographic pattern into unused regions of the secret documents on private cloud and VDI platforms. We devise a computationally efficient storage scanning mechanism for providing end-to-end traceability for the storage scanning can be performed in an event-driven manner since a steganographic mark are encoded into a well-regulated offset address of the storage, which decrease the computation overhead drastically. To evaluate the feasibility of the proposed scheme, this work has been undertaken on a real cloud platform based on OpenStack.

key words: *data traceability, cloud computing*

1. Introduction

The secret document leakage incidents is defined as the accidental or unintentional distribution of sensitive data to an unauthorized parties [1]. A leading cause of the incidents has been through network or via removable storage devices. As a countermeasure, many organizations are employing virtual desktop infrastructure (VDI) to prevent the leakage of the secret documents [2], [3]. In spite of the various security benefits of VDI, there is still challenges to prevent the secret document leakage incidents [4].

In this paper, we propose a novel scheme, called *Doc-Trace*, to provide a traceability framework for the secret documents in cloud computing environment. The works of inserting steganography into file have been carried out to send/receive secret data [5]–[7]. Our proposal is utilized for providing traceability of the secret document rather than for hiding the data. The steganography technology we proposed acts as a core not only to allow a host-level traceability but also to provide tamper-resistant trace mechanism for the secret documents. Overall contributions of this work can be summarized as follow:

First, we devised a steganography-based document trace mechanism for cloud computing environment, termed Doc-Trace. It injects a steganographic mark into the secret documents for providing end-to-end traceability. Therefore, steganography technology is utilized for providing traceability of the secret document rather than for hiding the data. In result, the steganography technology acts as a core not only to allow a host-level traceability but also to provide tamper-resistant trace mechanism for the secret documents.

Second, we devised a computationally efficient storage scanning mechanism for providing end-to-end traceability for the secret documents in a real cloud computing environment. More specifically, the storage scanning can be performed in an event-driven manner since a steganographic mark are encoded into a well-regulated offset address of the storage, which decrease the computation overhead drastically. This mechanism is implemented as a driver module in hypervisor layer. The driver is triggered whenever the steganographic marks inside the secret documents is modified so that the secret documents containing the mark are traced by the event-driven monitoring mechanism.

Last but not least, this work has been undertaken on an OpenStack-based real cloud computing testbed.

We devised mechanism to maximize the efficiency of file tracking documents in cloud platform. To exmaple it, we investigate thoroughly Compound File Binary Format (CFBF) [8] used in Microsoft Office as a standard file format. We then identified spare or unused regions of CFBF-based files. Based on the investigation, we devised an encoding scheme for injecting steganographic mark and usage logs into the secret document files in cloud, which allows trace to be identified by the centralized security center.In this paper, we proposed scanning method for CFBF (such as PPT, XLS, DOC, MSI). But if the position of the inserted pattern can be fixed, regardless of private cloud or VDI environment, tracking file and leakage detection could be possible. Our proposed scheme is fully compatible and independent with existing computing environment: the steganographic mark and usage logs are embedded into the secret documents without any violation of standard; *Doc-Trace* inspects the secret documents from outside for providing an end-to-end visibility for file transactions.

Doc-Trace, which we proposed, can be applied in environment as Fig. 1. Doc-Trace can also be used in cloud platforms where multiple VMs are running. When the document is edited in private cloud, our system monitors it in hypervisor layerl which manages block-storage [9], [10]. So, tracing document can be possible even without installing

Manuscript received December 15, 2016.
Manuscript revised May 21, 2017.
Manuscript publicized July 21, 2017.

[†]The authors are with Department of Computer and Information Security, Sejong University, Korea.
a) E-mail: woongbak@sejong.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2016INL0002

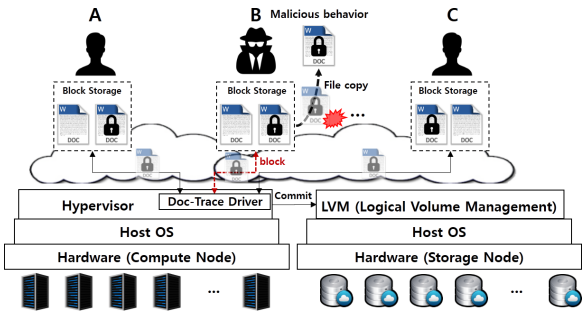

Fig. 1 Overview of Doc-Trace on cloud platform

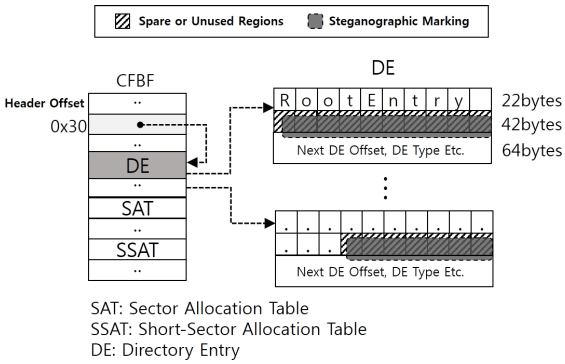

Fig. 2 Overview of compound file binary format architecture

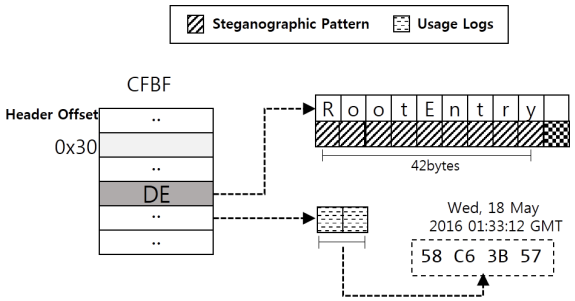

Fig. 3 Steganographic marking into compound file binary format (CFBF)

agent inside VM. Therefore, malicious user will not be able to evade the trace.

Composition of this paper is as follows. In Sect. 2, we discuss the details about analyzed contents of filesystem to utilize Doc-Trace. In Sect. 3, we describe the scenario and algorithm of Doc-Trace. In Sect. 4, we explain result of Doc-Trace experiment. Finally, at Sect. 5, conclusion.

2. Prototyping *Doc-Trace* in Cloud

2.1 Identifying Regions for Steganographic Marking

CFBF is a compound document file format by Microsoft for storing numerous files and streams within a single file on a storage. The format manages to several blocks that have 512 bytes and has a structure that can be saved as a directory and file like a file system. As shown in Fig. 2, CFBF is divided into three sections: Sector Allocation Table (SAT) that holds the size of 512 bytes; Short-SAT (SSAT) that holds smaller size than SAT's; and Directory Entry (DE) that holds the size of 128 bytes. In order to identify the proper regions to insert steganographic marking, we extracted spare or unused regions by means of referring to the sector table on each section and tracing the file stream. The sections of SAT and SSAT are identified to inappropriate regions to insert the steganographic marking since the section may not be ensured depending on the size of the corresponding file. In contrast, the DE is considered to an appropriate region since one CFBF contains multiple DE, and the DE is an essential elements of CFBF. Therefore, we focused on searching the spare regions in DEs and we found the start position of the first DE by referring to the CFBF header located in offset 0x30. Each DE (size of 128 bytes) consists of two parts. The first part (64 bytes) is for storing the entry name. The remaining part (64 bytes) contains the offset of the next DE. It allows to traverse the whole DE of the CFBF, looking up the offset of each successive DE as a chain until the end of the DE is reached. Among the DEs, the first DE is a special DE, called *RootEntry* that marks the beginning of the DE, used by every CFBF. The size of *RootEntry* is static (22 bytes), so that the remaining regions (42 bytes) can be used as the first spare region to insert a steganographic marking into the corresponding file.

2.2 Inserting Steganographic Mark into Secret Documents

Figure 2 shows how the steganographic mark and usage logs can be encoded into the identified spare region of CFBF. The first rule is that the steganographic mark is inserted into the first spare region located in the first DE. Since the size of the first spare region is static (42 bytes), the first 38 bytes is used to store the steganographic mark to prevent the secret documents from occurring false positive. The remaining 4 bytes are used to record the offset for the spare region of the next DE. The second rule is that the next spare regions are used for storing usage logs. Consequently, by referring to the two offsets (for the next DE and for the next spare region of the next DE), whole spare region can be accessed as a linked list of spare regions. This two-rules-based encoding scheme allows *Doc-Trace* not only to comply the file format of CFBF, but also to utilize a large size of spare region for storing usage logs of the file. The usage logs contains following information: access users, access times, modified time, last saved time, access IP, file location, etc.

2.3 Applying *Doc-Trace* into Cloud Platform

To gain end-to-end visibility for the transactions of the secret documents, we present our prototype applying *Doc-Trace* scheme on an OpenStack-based cloud platform. *Doc-Trace Driver* is implemented as a driver module in hypervisor layer. The driver is triggered whenever the embedded marks inside the secret documents are modified so that

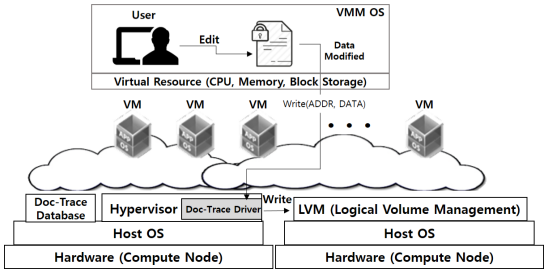

Fig. 4 Deploying Doc-Trace in cloud platform

the secret documents containing the mark are traced by the event-driven monitoring mechanism. When write request is valid on storage, the request is passed to LVM and applied to block storage. If the request is invalid, it is blocked in Host-Level. Therefore, the system we proposed can detect malicious user trying to copy the file or delete steganography by tracking files. This mechanism has been undertaken on an OpenStack-based real cloud computing testbed. The details of algorithm are followed.

2.3.1 Detection of Steganographic Mark Change for Secret Document Tracking

The monitoring operations based on Doc-Trace are performed as a driver module on the host layer, which is tightly coupled with LVM (Logical Volume Manager). LVM acts as a device mapper that provides block storage to users on private cloud or VDI platforms. LVM is divided into PV (Physical Volume) which is typically a hard disk, PE (Physical Extent) which has the same size as the logical extents for the volume group, and LE (Logical Extent) which is split into chunks of data. Users who use virtual block storage are allocated to LV (Logical Volume) which is visible as a standard block device. Because LV acts as block device in host, direct access to virtual disk can be possible. That is, files injected steganography can be scanned directly in host. Doc-Trace utilizes such features of LVM to trace secret documents efficiently.

Algorithm 1 shows the pseudo code of detection algorithm. When the user requests write operations on storage, *addr* and *data* are delivered to Doc-Trace function. If $17_{(16)}$ is the value of the delivered *addr* modulo 512 bytes, size of a sector, steganography could be embedded in *addr* (Line 9). If the above condition is satisfied, *addr*-$29_{(16)}$ can be used to derive the last written *time* of the corresponding secret document (Line 10). Then if *addr*is not included in a set of addresses storing steganographic mark and *data* matches with signature stored in *addr*, the requested operation is considered as file-copying operation (Line 11-12). If *addr* is included in a set of addresses storing steganographic mark and *data* does not match with signature stored in *addr*, the requested operation is considered as operation to modify signature (Line 13-14). Otherwise, if *addr* is included in a set of addresses storing steganographic mark and *time* does not match with last written *time* stored in *addr*, we update the trace log of *d.addr* into database (Line 15-17).

Algorithm 1 Detection of steganographic mark change for secret document tracking

```
1:  procedure Doc-TRACE(addr, data)                                  ▷
2:      addr: access address, data: data to be written into storage
3:      S_block: size of a sector (512 bytes)
4:      D: a set of elements in log database for secret document tracking
5:      · D.addr: a set of addresses storing steganographic mark
6:      · D.addr.sig: signature corresponding to the element stored in addr
7:      · D.addr.time: last written time of the element stored in addr
8:      · D.addr.log: log data corresponding to the element stored in addr
9:      if (addr mod S_block) = 17_(16) then
10:         time ← time data stored on (addr − 29_(16))
11:         if (addr ∉ D.addr) ∧ (data = D.addr.sig) then
12:             Operation ← Filecopy
13:         else if (addr ∈ D.addr) ∧ (data ≠ D.addr.sig) then
14:             Operation ← Malicious behavior
15:         else if (addr ∈ D.addr) ∧ (time ∉ D.addr.time) then
16:             Operation ← Update for the trace log of D.addr
17:             D.addr.log ← TraceLog
18:         end if
19:     end if
20: end procedure
```

3. Experminet

3.1 Measuring Operation Overhead Occurs while Monitoring

To retain traceability of secret documents, it is essential to scan storage. To compare Doc-trace scanning and agent based scanning, we measured each operation overhead. Scanning storage makes overhead inevitably. For details, we conducted the following experiments. We used ubuntu 16.04, 16GB of memory and normal HDD as storage server and used Windows 7 SP1 64bit, 4GB of memory and 50GB of block storage as VM. We measured CPU utilization and memory utilization while scanning storages. Details of the performance results are as follows. CPU utilization took 6% share and memory utilization took 11% share for agent-based scanning. On the other hand, CPU utilization took 0% share and memory utilization took 0% for Doc-trace scanning we proposed. Because Doc-trace we designed does not require agent on VM, the scheme does not make overhead at all.

3.2 Efficiency of Storage Scanning Mechanism Based on Doc-Trace

In order to evaluate the efficiency of the proposed storage scanning scheme, we compared agent-based and host-based storage scanning with storage scanning scheme based on Doc-Trace. We measured the storage scanning latency of each method with varying the size of storage. Figure 5 shows the performance comparison. In the storage of 1GB, storage scanning based on agent took 1.5s, storage scanning based on host took 0.98s while the mechanism we proposed took 0.002s. In the storage of 10GB, agent-based scanning took 117.8s, host-based scanning took 76.97s while the pro-

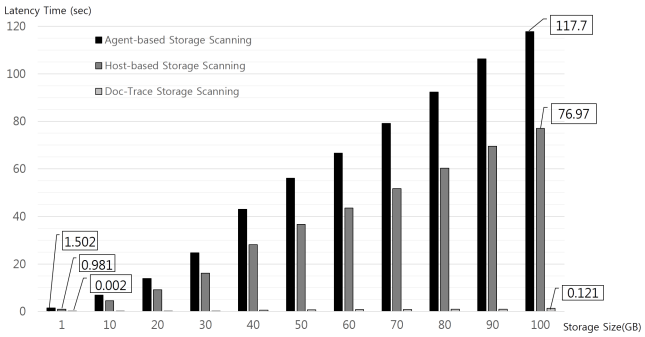

Fig. 5 Measuring storage scanning latency with varying storage size

posed scheme took only 0.12s. The results show the mechanism we designed is 975 times faster in performance than agent-based scanning, and 633 times faster than host-based scanning. Doc-trace outperforms other scanning methods. This is clear given that other scanning methods require more latency times to find steganography, as shown in Fig. 1. In addition, this mechanism is implemented as a driver module in hypervisor layer. The driver is triggered whenever the embedded marks inside the secret documents is modified so that the secret documents containing the mark are traced by the event-driven monitoring mechanism.

4. Conclusion

Our task has been to provide an end-to-end visibility for secret documents by inserting steganographic pattern in cloud computing environment such as VDI. To accomplish this task, we proposed *Doc-Trace* to gain end-to-end visibility for the transactions of the secret documents by inserting steganographic pattern into spare or unused regions of the secret documents. To evaluate the feasibility of the proposed scheme, we applied the proposed *Doc-Trace* scheme into a real cloud platform based on OpenStack. To track the document efficiently, we proposed the searching-friendly mechanism. The results showed the mechanism we designed is 975 times faster in performance than agent-based scanning, and 633 times faster than host-based scanning. Last but not least, the operational resilience of *Doc-Trace* should be guaranteed for providing sustainability. Our next step is to consider the trustworthiness for the steganographic marking. Currently, we are investigating *Doc-Trace* from the perspectives of traceability without the consideration of operational resilience. We believe that putting trusted platform mechanism and trusted marking schemes is an appropriate way forward. We are working towards an advanced *Doc-Trace*-based leakage prevention system with more trustworthiness against anti-trace attacks.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2017R1C1B2003957, NRF-2016R1A4A1011761)

References

[1] I.A. Center, "Global data leakage report, h1 2015," https://infowatch.com/, p.5, 2015.
[2] F. Bellard, "Qemu, a fast and portable dynamic translator," USENIX Annual Technical Conference, FREENIX Track, pp.41–46, 2005.
[3] A. Velte and T. Velte, Microsoft virtualization with Hyper-V, McGraw-Hill, Inc., 2009.
[4] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L.P. Cox, J. Jung, P. McDaniel, and A.N. Sheth, "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," ACM Transactions on Computer Systems (TOCS), vol.32, no.2, p.5, 2014.
[5] A.M.S. Rahma, B. AbdulWahab, and A. Al-Noori, "Proposed steganographic method for data hiding in microsoft word documents structure," Al-Mansour Journal, no.15, pp.1–29, 2011.
[6] M.A. Mohamed, O.G. Altrafi, M.O. Ismail, and M.O. Elobied, "A novel method to protect content of microsoft word document using cryptography and steganography," International Journal of Computer Theory and Engineering, vol.7, no.4, p.292, 2015.
[7] M.M. Amin, M. Salleh, S. Ibrahim, M.R. Katmin, and M.Z.I. Shamsuddin, "Information hiding using steganography," Telecommunication Technology, 2003. NCTT 2003 Proceedings. 4th National Conference on, pp.21–25, IEEE, 2003.
[8] V. Roussev and S.L. Garfinkel, "File fragment classification-the case for specialized approaches," Systematic Approaches to Digital Forensic Engineering, 2009. SADFE'09. Fourth International IEEE Workshop on, pp.3–14, IEEE, 2009.
[9] L.D. Baranovsky, L.F. Cabrera, C. Chin, and R. Rees, "Logical volume manager and method having enhanced update capability with dynamic allocation of storage and minimal storage of metadata information," 1999. US Patent 5,897,661.
[10] M.A. Grubbs, G.F. McBrearty, and G.H. Neuman, "File system backup in a logical volume management data storage environment," 2004. US Patent 6,829,688.