Flashlight: A Novel Monitoring Path Identification Schema for Securing Cloud Services

Heng Zhang, Jesus Luna, Neeraj Suri
Technischen Universität Darmstadt
Darmstadt, Germany
zhang,jluna,suri@deeds.informatik.tu-darmstadt.de

Ruben Trapero
Atos Research & Innovation
Madrid, Spain
ruben.trapero@atos.net

ABSTRACT

Cloud monitoring is an essential mechanism for helping secure cloud services. Thus, a plethora of monitoring schemas have been proposed in recent years. Particularly, a newly proposed indirect monitoring mechanism outperforms others with the unique merit of addressing scenarios where the information of the monitoring target is not directly accessible. To conduct indirect cloud security monitoring, a key prerequisite is to obtain a special set of monitoring data termed "*monitoring path*". However, how to ascertain the monitoring path is still an open issue.

In this paper, we propose *Flashlight* as a novel monitoring path identification mechanism to address the gap where the information of monitoring targets is inaccessible. For this purpose, *Flashlight* first introduces a novel data reduction technique to filter unnecessary monitoring information. Second, *Flashlight* develops a data association approach to identify the monitoring path by utilizing data relations and data attributes. Third, *Flashlight* devises a *monitoring property graph* to support fine-grain monitoring path identification as well as represent identified monitoring paths. In addition, the efficacy of our proposed approach is demonstrated by the case studies where *Flashlight* successfully identifies the monitoring paths for underpinning indirect cloud monitoring.

CCS CONCEPTS

• **Security and privacy → Security services**; **Software and application security**;

KEYWORDS

Cloud Security, Service Monitoring, Indirect Monitoring, Monitoring Path, Dependency.

ACM Reference Format:
Heng Zhang, Jesus Luna, Neeraj Suri and Ruben Trapero. 2018. Flashlight: A Novel Monitoring Path Identification Schema for Securing Cloud Services . In *Proceedings of International Conference on Availiability, Reliability and Security (ARES'18).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.475/123_4

ARES'18, August 2018, Hamburg, Germany

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

1 INTRODUCTION

Cloud monitoring is a critical mechanism to help secure cloud services, and at the same time validating the compliance of security requirements proposed by cloud service customers (CSCs). It utilizes collected monitoring information for ensuring data security, managing service performance, protecting user privacy, and other objectives. Consequently, a variety of cloud monitoring schemas have been proposed in recent years [4, 5, 9, 16, 29, 33, 34]. The proposed monitoring schemas are predominantly developed based on an assumption where the monitoring information is directly accessible. In reality, only a very few of monitoring information can be accessed by CSCs. For example, CloudWatch that targets monitoring applications and resources in Amazon Web Services (AWS) only provides the CSCs with a small number of monitoring metrics, such as CPU utilization, disk read/write behaviors, network traffic, and status check reports [3]. In many cloud services, some monitoring information is also restricted to be accessed by the CSCs. For instance, the version information of OpenSSL library, which is adopted by a Software-as-a-Service (SaaS) cloud service provider (CSP) for securing network communications, is important for monitoring the Heartbleed attack [21] but cannot be directly accessed for security reasons. Unfortunately, the proposed schemas cannot deal with this situation where particular monitoring information is inaccessible. The situation obstructs CSCs to check the security compliance of the subscribed cloud services.

The challenge of lacking direct access to particular monitoring information has recently been addressed by an indirect cloud monitoring mechanism [40]. This indirect monitoring mechanism is developed based on the observation that cloud services typically do not function on a standalone basis but could share system resources in some situation. Resource sharing could introduce data associations which can be exploited by attackers [14, 35, 41]. For example, a study [26] reports that the AES [20] and RSA [27] secret keys can be stolen by exploiting the data association introduced by memory resource sharing in virtualized environments, while such data associations can also be utilized to monitor cloud services. The proposed monitoring mechanism particularly takes advantage of the data association termed "*monitoring path*", which is formed by a set of accessible monitoring data and related data relations, to indirectly infer the inaccessible monitoring information. For example, the encryption overhead of the cloud service depends on the adopted encryption method and the size of the target file to encrypt. By aggregating the monitoring path that is formed by the information of the target file size, the adopted encryption method, and the dependency, the indirect mechanism can infer the information of encryption overhead which cannot be directly monitored. While the proposed indirect monitoring mechanism exhibits its

unique advantage of monitoring the particular information that is inaccessible for a variety of reasons (e.g., technical difficulties, intellectual property protection issues, or privacy concerns), the key problem of ascertaining monitoring paths to enable the indirect cloud monitoring has not been addressed.

The monitoring path is difficult to ascertain due to several reasons. First, it is hard to discern the valuable monitoring information that is relevant to help identify monitoring paths. Massive monitoring data is continuously collected by cloud security monitors, while only a small portion of monitoring data is useful for spotting potential data associations utilized as the monitoring paths for performing indirect cloud monitoring tasks. An effective technique for filtering the irrelevant monitoring data from the collected data is demanded. Second, there is a lack of quantitative techniques to associate useful monitoring data for generating the monitoring path. Given a set of monitoring data and a monitoring target, a valid quantitative technique is required to identify the data association that underpins the indirect monitoring on the inaccessible information of the monitoring target. Third, there is an absence of an effective schema that can not only represent the identified monitoring paths but also further support identifying monitoring paths in a complex scenario. It is difficult to properly represent the identified monitoring paths with existing representation mechanisms. Especially, the monitoring data involved complex data relations (e.g., service dependency, time causality, or data consistency) introduces additional challenges for identifying monitoring paths. Without properly ascertaining monitoring paths , the efficacy of the proposed indirect monitoring mechanism is substantially jeopardized.

In this paper, we follow the research direction of the emerging indirect cloud monitoring and thus propose *Flashlight* as a novel monitoring path identification mechanism to address the aforementioned gap. Given a set of collected monitoring data and the related information (i.e., data attributes and data relations), our proposed mechanism can select useful monitoring data from the collected data set, identify particular data associations as the monitoring paths, and represent the identified monitoring paths together with other useful information. To this end, *Flashlight* first makes use of the data attributes to obtain valuable monitoring data by filtering a significant amount of trivial monitoring data. Next, *Flashlight* takes advantage of the data relations to identify data associations (i.e., monitoring paths) by utilizing the statistical characteristics of the collected monitoring data. Finally, *Flashlight* proposes a monitoring property graph to represent the identified monitoring paths and other complex information of the collected monitoring data (i.e., values, attributes, relations, and associations) which underpins the fine-grain monitoring path identification. By employing *Flashlight*, security professionals can conveniently obtain monitoring paths for performing indirect cloud monitoring rather than manually conducting a cumbersome monitoring path analysis which is not only requiring a profound knowledge of all the collected monitoring data but also strictly subject to the scale of the collected data set.

To the best of our knowledge, our work is the first monitoring path identification mechanism for facilitating indirect cloud monitoring. In summary, we make the following contributions:

(1) We introduce a novel monitoring path identification mechanism that can discern valuable monitoring data and ascertain monitoring paths for performing indirect cloud monitoring.
(2) We propose a novel monitoring property graph with the advantage of representing complex information (i.e., data value, data attributes, data relations, data associations, and monitoring paths) in order to improve the efficacy of the indirect cloud monitoring methodology.
(3) We evaluate the efficacy of the proposed mechanism with identifying monitoring paths that help indirectly monitor different classes of practical security threats in cloud.

The remainder of this paper is organized as follows. Section 2 outlines the issues behind the monitoring path identification. Section 3 formulates the monitoring path identification problem. Section 4 details the design of the proposed mechanism. The effectiveness of the proposition is evaluated in Section 5 by performing case studies on real cloud security threats. Section 6 reviews related work.

2 BACKGROUND

To motivate the issues, we first present two prominently reported security threats to highlight the significance of monitoring path identification for achieving effective service security compliance monitoring by using the indirect cloud monitoring mechanism. Subsequently, we analyze the reported threats to provide insights on the monitoring data underpinning our design.

2.1 Existing Threats

Our work is motivated by the CSC's pragmatic requirement of monitoring the security compliance of cloud services. Effectively monitoring security compliance requires in-depth understanding of the links among monitoring data collected from cloud services whose security compliance might be violated by security threats.

2.1.1 The access-driven cache attack. A typical security requirement from the CSC is about the encryption key management which targets keeping the applied encryption key in safe. However, it is difficult to monitor the compliance of this security requirement without perceiving the valid monitoring path. For example, an access-driven cache attack (ADCA) aims to exploit the timing behaviors of cache accesses so as to compromise the confidentiality of cloud services. Gullasch et al. [11] present an access-driven cache attack which enables an unprivileged spying process to recover the secret key of a running encryption process (AES-128). To conduct such an attack, the attacker exploits the knowledge of the *cache specification*, namely the *cache access status (hits/misses)*, *CPU cycles*, and *AES*-128 *encryption protocol*. It is challenging to monitor the security compliance which might be probably violated by this particular attack, if the interlinked data associations across the data elements are not fully understood.

2.1.2 The resource-freeing attack. The CSC also puts emphasis on the business continuity management to assure the service availability varying within the specified range. Nonetheless, it is laborious to monitor the compliance of this security requirement without deeply understanding the monitoring data. For instance, a resource-free attack (RFA) targets the imperfect isolation mechanism of the virtual machine (VM) hypervisor. The attacker introduces crafted

interference on a victim VM to trigger a performance bottleneck which leads the victim VM to free up the system resource for the beneficiary VM. Varadarajan et al. [32] present the RFA case to exploit the hypervisor isolation policy (i.e., the fair-sharing policy) to free up the network resource from one web service to another web service, even though both web services should equally share the network resources. In this case, the attacker implements the attack by exploiting the knowledge of, *CPU utilization*, *high CPU overhead service request*, and *the crafted attack process*. Likewise, it is also critical to discover the compliance violation caused by the RFA attack with effectively understanding collective data associations.

2.2 Example Interpretation

These examples highlight the challenge of monitoring the security compliance of cloud services without gaining an insight on the collective monitoring data. The key point for addressing this challenge is to identify the "useful" associations across the monitoring data. A data association can be regarded as the *monitoring path*, which is formed by a sequence of monitoring data and data relations, used for monitoring the security compliance. To identify the monitoring path, two dominant issues need to be properly considered.

First, the monitoring data collected from the cloud carries much more information than just the data value. Considering to monitor Distributed Denial of Service (DDoS) attacks, a lot of monitoring data is collected by security monitors. The value of monitoring data (e.g., the amount of incoming user requests) can provide important information for monitoring DDoS attacks. Besides, some special types of monitoring data might have higher criticality than other collected data (e.g., the file accessing data or the process context data) in the context of the DDoS attack monitoring. Moreover, monitoring data might be subject to various relations such as time causality or service dependency. Without a collective understanding of the monitoring data, it is difficult to identify the monitoring path.

Second, an effective data association technique for identifying the monitoring path is missing. While a large number of cloud monitoring techniques have been proposed by researchers [4, 5, 18, 30], these techniques target monitoring the specific individual data (e.g., parameters, values, or formats). Given the paucity of data association mechanisms, these approaches cannot properly monitor the security compliance that may be violated by particular security threats (e.g., ADCA or RFA) mentioned in Section 2.1. Determining such data associations underlies our proposed approach.

2.3 Data Relation

Relations across the monitoring data is indispensable for identifying data associations in the raw monitoring data set [7]. Data relations encompass a wide range of aspects, such as causality, consistency, or dependency. For example, data dependency is a common type of data relation existing among many cloud services. Data dependency is observed between the Heartbleed attack and the OpenSSL library which is a widely used cryptography library for encrypting the network connection between a cloud server and its clients. Namely, if the OpenSSL library supporting the Heartbeat mechanism has the version number of "1.0.1" (excluding "1.0.1g") [21], the Heartbleed attack could take effect. Therefore, the indispensable knowledge on such dependency helps identify data associations.

Besides, the attribute of monitoring data is also the critical knowledge that can be used to facilitate identifying data associations. In this paper, we particularly consider the collected monitoring data with two different attributes, namely *reducibility* and *criticality*.

Reducibility is used to capture the necessity of the monitoring data. For example, when Dropbox conducts the directory scanning operation (by running the "pcscd" daemon), this operation can produce a lot of repetitive monitoring information (i.e., *reading/accessing* operations recursively conducted in the specified path) [36]. From a monitoring perspective, the repetitive data does not convey any additional useful information. Therefore, the repetitive data can be reasonably reduced by leveraging data reducibility.

Criticality is used to capture the significance of the monitoring data. For every monitoring target, the collected monitoring data has different levels of importance. As the ADCA example shows, the monitoring data of *cache access status (hits/misses)* has more importance than *file accessing status* with respect to monitoring the security compliance that may be violated by this particular security threats. Accordingly, the collected monitoring data can be weighted by taking advantage of the data criticality.

3 PROBLEM FORMULATION

On this background, we now present a formalization of the monitoring path identification problem.

In this paper, we consider the scenario that contains a set of monitoring data $\mathcal{D} = \{d_1, d_2, \cdots, d_k\}$ (for some $k \in \mathbb{N}$) collected by deployed security monitors and a set of data relations $\mathcal{R} = \{R_1, R_2, \cdots, R_l\}$ (for some $l \in \mathbb{N}$) existing among the collected data. The data relation R_i is a function $R_i : \{\mathcal{D}\} \rightarrow \mathcal{P}(\mathcal{D})$. The output of the function is a subset $D_i = \{d_{i1}, d_{i2}, \cdots, d_{ij}\}$ (for some $j \in \mathbb{N}, 1 \leqslant i \leqslant l$) where d_{ij} is collected monitoring data represented by a 3-tuple $d_{ij} = (v_{ij}, r_{ij}, c_{ij})$. For monitoring data d_{ij}, v_{ij} represents its value, $r_{ij} \in \{reducible, irreducible\}$ represents its data reducibility, and $c_{ij} \in [0, 1]$ represents its data criticality. That is to say, when the relation $\mathcal{R}_i$ functions on the data set $\mathcal{D}$, a corresponding subset of monitoring data D_i can be obtained.

This scenario targets the monitoring data collected by taking advantage of data relations (e.g., causality, consistency, or dependency) and data attributes (e.g., criticality and reducibility). Notably, data relations and attributes of monitoring data can be derived by utilizing the expert knowledge or applying existing approaches [10, 24, 39]. For the sake of simplicity, we assume that one collected monitoring data is subject to only one type of relation. As a matter of fact, even though the collected monitoring data may be subject to multiple types of relations, it is necessary to consider one specific type of relation for monitoring the information of a specific target.

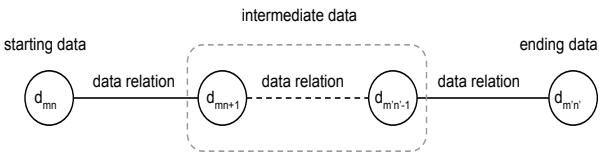

Figure 1: An Example of Monitoring Path

With the above definitions, we define a monitoring path p as a structure that begins from a monitoring data $d_{mn} \in \mathcal{D}(m, n \in \mathbb{N})$

via a sequence of intermediate monitoring data and ends at another monitoring data $d_{m'n'} \in \mathcal{D}(m', n' \in \mathbb{N})$. For each intermediate data of monitoring path p, it associates a predecessor with a successor. The association between adjacent monitoring data in the monitoring path reflects relations between the data. As depicted in Figure 1, the monitoring path p is composed of starting data d_{mn}, ending data $d_{m'n'}$, and a set of intermediate data (e.g., d_{mn+1} or $d_{m'n'-1}$). Two adjacent data d_{mn}, d_{mn+1} are required to belong to a common data relation: $d_{mn}, d_{mn+1} \in R_i(\mathcal{D})$ for some $1 \leqslant i \leqslant l$. By synthesizing such data associations, monitoring path p bridges the accessible monitoring data and the inaccessible monitoring target so as to enable the indirect monitoring task.

In this paper, we propose to identify the monitoring data association by utilizing the data attributes and the data relations. The monitoring path identification problem is defined as follows.

- Given a relation set $\mathcal{R}$,
- Given a monitoring data set $\mathcal{D}$, where the collected monitoring data d_{ij} is equipped with two data attributes : data reducibility r_{ij} and data criticality c_{ij},

▷ The monitoring path identification problem is to identify a set of monitoring paths $\mathcal{P} = \{p_s; s \in \mathbb{N}\}$ where p_s is the s^{th} monitoring path in $\mathcal{P}$.

4 PROPOSED METHODOLOGY

In this section, we detail the proposed methodology. We first provide an overview of the framework of the proposed mechanism. Based on the framework, we explain the specific design step by step.

4.1 Methodology Overview

The key point of the monitoring path identification is to identify the relevant monitoring data associations. We take advantage of data relations and data attributes as discussed in Section 2.3 to identify data associations. As a result, we propose a multi-step monitoring path identification mechanism as depicted in Figure 2. The proposed mechanism contains three steps as follows.

(1) *Reducible intra-relation data association* takes a set of monitoring data $\mathcal{D}$ and a set of data relations $\mathcal{R}$ as the input to execute the association process for deriving the data association with the same relation.
(2) *Weighted inter-relation data association* takes data associations derived from different relations (Step 1) as input to associate them as a complete monitoring path.
(3) *Monitoring path representation* takes the relevant information from the previous steps as the input for representing the identified monitoring paths in the proposed monitoring property graph where the represented monitoring information (i.e., data values, data attributes, data relations, and data associations) facilitates monitoring path identification when the monitoring data involves with more complex relations.

4.2 Methodology Design

We now detail the proposed multi-step methodology as demonstrated in the framework that is depicted in Figure 2.

4.2.1 Reducible intra-relation data association. To identify the data associations in the monitoring data set, *Flashlight* first reduces the input data set by utilizing data attributes (i.e., *reducibility*) and then ascertains data associations with the identical relation therein.

In practice, the collected monitoring data set is not necessarily completely required for conducting the specific monitoring task. Considering the Dropbox example in Section 2.3, we notice that even though a large amount of monitoring data is collected (i.e., the repetitive low-level file system accessing data), the meaningful monitoring information is about the scanning operation that triggered the "pcscd" process. In other words, the repetitive low-level data does not provide much meaningful monitoring information.

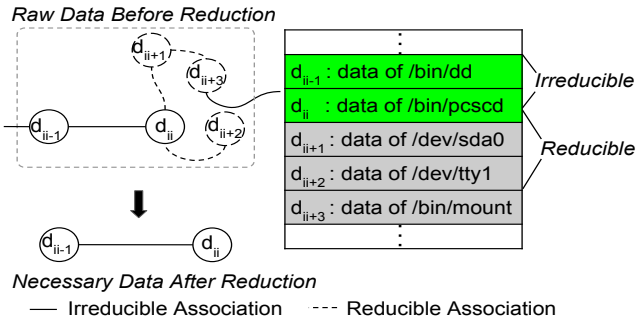

Figure 3: An Example of Data Reduction

Without loss of generality, we explain the data reduction process in Figure 3 which consists of five nodes (monitoring data) and six edges (data associations). In this figure, we use full circles to represent *irreducible monitoring data* (e.g., d_{ii} can denote the monitoring data of "pcscd" process), while dashed circles denote *reducible monitoring data* (e.g., d_{ii+1} is the monitoring data of the low level file system accessing operation occurred in hard disks). The solid edge represents the irreducible association between a pair of the irreducible monitoring data. In contrast to the solid edge, the dotted edge represents the reducible association which is introduced by the reducible monitoring data. In this example, the "pcscd" process is executed by calling $/bin/dd$ command. Then, the "pcscd" process repetitively performs a series low level of operations like mounting storage devices ($/bin/mount$), reading hard disks ($/dev/sda0$), or accessing TTY devices ($/dev/tty1$). From a monitoring perspective, monitoring information of those repetitive operations does not give more help and thus can be properly reduced. Therefore, by removing the reducible monitoring data, the original graph can be reduced to a simpler structure as a two-node association between d_{ii-1} and d_{ii}. This data reduction process is conducted by examining the value of r_{ij} in the triple d_{ij} as defined in Section 3. To be specific, if r_{ij} equals to "reducible", d_{ij} will be removed from the input data set. Otherwise, d_{ij} will be kept in the data set.

After filtering the unnecessary data, *Flashlight* proceeds to an intra-relation data association process. By running a relation-based monitoring process (e.g., a monitoring process collects several data in terms of the service dependency), certain monitoring data combinations frequently appear in the collected monitoring data set. For example, Transport Layer Security (TLS) handshake request and response messages frequently appear on Port 443 (i.e., the HTTPS port) during monitoring the Heartbleed attack. Such data

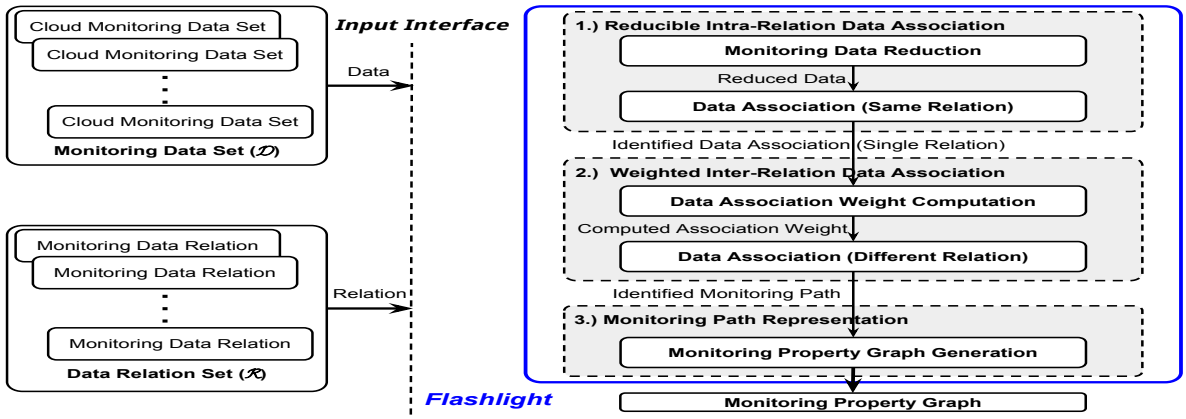

Figure 2: The Framework of the Monitoring Path Identification Methodology

co-appearance indicates the existence of an association among the monitoring data. The frequency of the co-appearance also reflects the association degree of the data. With the above observation, the intra-relation monitoring data association is defined as follows.

Definition 1. *Let* (d_{ij-1}, d_{ij}) *be a pair of monitoring data subject to the same relation* R_i. *The monitoring data association between* d_{ij-1} *and* d_{ij} *is denoted as* $\mathcal{A}|_{d_{ij}}^{d_{ij-1}} : d_{ij-1} \uplus d_{ij}$.

In this paper, $\uplus$ is the symbol used for representing the association between two different objects, such as two different monitoring data or two different monitoring data associations.

Flashlight adopts two metrics (i.e., the *support* and the *confidence*) to characterize the data association $\mathcal{A}|_{d_{ij}}^{d_{ij-1}}$ inspired by work [1] that utilizes the two metrics to quantify statistic properties of different objects to discover existing associations among the objects.

The *support (sup)* of the data association $\mathcal{A}|_{d_{ij}}^{d_{ij-1}}$ denotes the proportion of the collected monitoring data sets which include both d_{ij-1} and d_{ij}. It captures the co-appearance rate of two monitoring data by representing with the probability of the data co-appearance. Therefore, the formula for computing *support* is given as follows.

$$sup(\mathcal{A}|_{d_{ij}}^{d_{ij-1}}) = \sigma(d_{ij-1} \bigcap d_{ij}) \tag{1}$$

In Formula (1), $\sigma(d_{ij-1} \cap d_{ij})$ presents the proportion of the monitoring data sets containing both d_{ij-1} and d_{ij}. These data sets are collected by the monitoring process with utilizing relation R_i.

The *confidence (conf)* of the data association $\mathcal{A}|_{d_{ij}}^{d_{ij-1}}$ denotes the ratio of d_{ij} in the collected monitoring data sets which contain both d_{ij-1} and d_{ij}. It captures the conditional probability of d_{ij}'s appearance given d_{ij-1}'s appearance. Accordingly, the formula for computing *confidence* is given as follows.

$$conf(\mathcal{A}|_{d_{ij}}^{d_{ij-1}}) = \frac{\sigma(d_{ij-1} \cap d_{ij})}{\sigma(d_{ij-1})} \tag{2}$$

In Formula (2), $\sigma(d_{ij-1})$ presents the proportion of the monitoring data sets containing d_{ij-1}. Likewise, these data sets are collected by the relation R_i based monitoring process.

As the monitoring data appearance is regarded as a random variable, the correlation between different data also needs to be considered. A common method to quantify the random variables' correlation is to compute the Pearson correlation coefficient of the given variables. However, the Pearson correlation coefficient is generally hard to compute [31]. To effectively capture the correlation between monitoring data, *Flashlight* introduces the *ambiguity* which can not only represent monitoring data correlations but also be easier to determine by applying an optimized solution as follows.

$$amb(\mathcal{A}|_{d_{ij}}^{d_{ij-1}}) = \sqrt{\frac{p(d_{ij-1}, d_{ij})}{p(d_{ij-1}) \cdot p(d_{ij})} \cdot \sigma(d_{ij-1} \bigcap d_{ij})} \tag{3}$$

In Formula (3), $\sigma(d_{ij-1} \cap d_{ij})$ is $sup(\mathcal{A}|_{d_{ij}}^{d_{ij-1}})$, $p(d_{ij-1})$ is the probability of d_{ij-1}, and $p(d_{ij})$ is the probability of d_{ij}. $p(d_{ij-1}, d_{ij})$ is the joint probability of both data.

By applying Formulas (1)(2)(3), the intra-relation data associations can be identified over data set D_i and the identification results can be presented by an association vector $I(\mathcal{A}|_{d_{ij}}^{d_{ij-1}}) = (sup(\mathcal{A}|_{d_{ij}}^{d_{ij-1}}), conf(\mathcal{A}|_{d_{ij}}^{d_{ij-1}}), amb(\mathcal{A}|_{d_{ij}}^{d_{ij-1}}))$.

With this data association process, *Flashlight* can apply it in a stepwise mode for associating more data subject to the same relation. For example, to associate the next monitoring data d_{ij+1}, *Flashlight* regards the derived data association $\mathcal{A}|_{d_{ij}}^{d_{ij-1}}$ as a whole (i.e., like one "virtual data") and applies the intra-relation data association process to identify the association vector $I(\mathcal{A}|_{d_{ij+1}}^{d_{ij}})$. The data association process keeps executing until no more data can be associated according to the predefined threshold or baseline.

4.2.2 Weighted inter-relation data association. In many scenarios, the monitoring path is a combination of multiple data associations derived from different data sets subject to different types of relations. Therefore, the *inter-relation data association* technique is proposed based on the following idea: In Step 4.2.1, a set of data associations are derived by leveraging different data relations. To properly select and combine the derived data associations, the inter-relation data association can be achieved.

To perform the inter-relation data association, we consider two different components: the *precedent* and *subsequent*. The *precedent*

is the data association that is about to be combined with another data association. The *subsequent* is the data association selected for the combination. The combination of both components can be also regarded as a random event where both *precedent* and *subsequent* appear at the same time with respect to monitoring a specific target.

Different from the intra-relation data association where the data is subject to the same relation, the inter-relation data association needs to select the *subsequent* for the *precedent*. One common practice for making selections is to consider the criticality of the selection object. As a result, we propose a weighting method for selecting the *subsequent* based on considering the criticality.

As the *subsequent* is a data association that contains a set of monitoring data, we propose to adopt the mean value of the data criticality in the *subsequent* as its *weight*. Formally, the *weight* of the data association is defined as follows.

Definition 2. *Weight W of a data association is the mean value of the data criticality of the association which can be computed as*

$$W(\mathcal{A}|_{d_{io}}^{d_{ih}}) = W(d_{ih} \biguplus \cdots \biguplus d_{io}) = \frac{1}{o-h}\sum_{j=h}^{j=o} c_{ij} \qquad (4)$$

Where, $\mathcal{A}|_{d_{io}}^{d_{ih}} (h \leqslant o; h, o \in \mathbb{N})$ represents the data association of $d_{ih} \biguplus \cdots \biguplus d_{io}$ and c_{ij} is the data criticality of d_{ij}.

By comparing weight W, the data association of the maximum value W in all the identified intra-relation associations is selected as the *subsequent* for a given *precedent*. The association process of *precedent* and *subsequent* can be carried out similarly to the intra-relation data association by regarding the identified associations as random variables. As a result, the identified inter-relation data association is regarded as the identified monitoring path which contains a set of monitoring data subject to different data relations.

4.2.3 Monitoring result representation. After executing the data association processes, *Flashlight* represents the output of the monitoring path identification process with a graph. Moreover, the graph needs to be able to represent complex information, such as the collected monitoring data, the corresponding data value, the data attribute, the data relation, and the identified data association so as to support the proposed data association process to deal with more complex monitoring scenario. Unfortunately, common graphical methods cannot meet these requirements.

To address this problem, *Flashlight* introduces the *Monitoring Property Graph* inspired by [28]. Compared to other graphical representations, the proposed monitoring property graph is capable of representing a variety of data information as depicted in Figure 4. The definition of the monitoring property graph is given as follows.

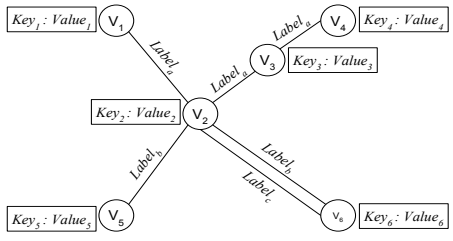

Figure 4: An Example of Monitoring Property Graph.

Definition 3. *A monitoring property graph $G = (V, E, \lambda, \mu)$ is an edge-attributed multi-graph, where V is a finite set of vertices which represent the monitoring data d_{ij}. $E \subseteq (V \times V)$ is a finite set of edges which refer to the data associations $\mathcal{A}|_{d_{ij}}^{d_{ij-1}}$. $\lambda : E \rightarrow \Sigma$ is a labeling function that specifies the edges E with the selected labels Σ for stating the association vector $I(\mathcal{A}|_{d_{io}}^{d_{ih}})$, and $\mu : V \rightarrow K \times S$ annotates vertices with key-value pairs, where $k \in K$ (K is the key set) represents the key (e.g., name) and $s \in S$ (S is the value set) represents the value.*

Figure 4 shows a monitoring property graph consisting of six vertices $\{V_1, V_2, V_3, V_4, V_5, V_6\}$ and six edges with three different kinds of labels $\{label_a, label_b, label_c\}$. Every vertex is annotated with the key-value pair (e.g., V_1 is annotated with Key_1 and $Value_1$). Notably, the monitoring property graph is a multi-graph that allows multiple edges between two vertices. For example, two edges exist between V_2 and V_6 while each edge has a different label (i.e., $Label_b$ and $Label_c$). Therefore, the monitoring property graph can address the challenge that monitoring data involves multiple relations.

The proposed monitoring property graph enhances the monitoring path identification, as it supports the data association process functioning in complex data relation scenario with fine granularity. Namely, the data association process can be conducted on selected monitoring information (i.e., data value, data attribute, and data relation) of the monitoring property graph. As an example, if monitoring data involves more than one data relation, it can be regarded as a compound of several "virtual data" where each "virtual data" only involves one single data relation. Therefore, the data association process can be conducted with finer granularity so as to effectively identify more monitoring paths.

5 EVALUATION

We evaluate the effectiveness of the proposed methodology for helping secure cloud services by performing case studies on two different classes of security threats. The evaluation tasks focus on addressing two critical questions as follows.

- Can *Flashlight* identify the monitoring path for supporting indirect cloud monitoring on the known security threats while some key monitoring information is inaccessible?
- Can *Flashlight* identify the monitoring path for supporting indirect monitoring on the unknown cloud security threats with existing monitoring information?

In the following parts, we present the details of the case studies as well as the related discussions.

5.1 Experimental Settings

We evaluate *Flashlight*'s efficacy of ascertaining monitoring paths to help monitor real security threats by simulated experiments. To this end, we set up an Apache web server for simulating a cloud service where the storage functionality is supported by the installed Samba server (version 4.5.9). Besides, we adopt the HTTPS protocol to secure the communication between the web server and simulated users. We adopt the vulnerable OpenSSL library (version 1.0.1) to implement the TLS protocol for the HTTPS connections. To simulate the situation that only limited amount of monitoring information is available in real cloud, we deploy a network monitor to capture the amount of incoming TCP requests (d_{11}), the amount of

port 443 requests (d_{12}), the amount of HTTPS responses (d_{13}), and the amount of odd-place logins (d_{21}). Additionally, we deploy a system activity monitor to capture the amount of privilege-required operations (d_{22}), process overhead (d_{31}), and network traffic overhead (d_{32}). Some relevant data is also required to be collected by leveraging expert knowledge or statistics analysis on security databases like CVSS[1], NVD[2], or exploit-DB[3]. In our experiments, the obtained relevant data consists of data relations (i.e., time causality R_1 and service dependency R_2), data reducibility (reducible / irreducible), data criticality, and monitoring data appearance probabilities. We list the collected information in Table 1 which contains three data relations, two data attributes, and four different kinds of normalized data appearance probabilities.

5.2 Case Study

5.2.1 ***Case I : The Heartbleed Attack***. For many cloud services, some security threats are difficult to monitor as the particular monitoring information for characterizing the threats is inaccessible. For example, the Heartbleed attack is a notorious security threat that can compromise the CSC's security requirements by exploiting the code flaw of OpenSSL cryptography library (version 1.0.1). By sending crafted packets, attackers are able to retrieve memory content that may contain confidential information (e.g., user passwords, credit card numbers, or other sensitive information) from the cloud service adopting the vulnerable OpenSSL library for encrypting service communications over port 443 [8]. However, the OpenSSL version information that can characterize if the cloud service prone to the Heartbleed attack cannot be directly accessed by the CSC, as it is restricted by the underlying CSP (i.e., a PaaS provider deploys the OpenSSL library to offer the encryption function). To highlight *Flashlight*'s virtue of ascertaining the monitoring path for indirectly monitoring security threats without the access to the key information, we perform a case study on identifying the monitoring path of the abstracted Heartbleed attack.

Flashlight performs the monitoring path identification process by utilizing the collected information as follows.

Step 1. Reducible intra-relation data association: *Flashlight* first starts the reducible intra-relation data association process over the monitoring data subject to same relation. Table 1 lists three different types of relations: $R_1(causuality)$, $R_2(dependency)$, and $R_3(abnormality)$. Accordingly, three different data sets can be derived : $D_1 = \{d_{11}, d_{12}, d_{13}\}$, $D_2 = \{d_{21}, d_{22}\}$, and $D_3 = \{d_{31}, d_{32}\}$. As the value of the collected monitoring data d_{ij} does not affect the identification process, Table 1 simply denotes the real data value as v_{ij}. The listed data is used for performing association processes.

Prior to running the association process, *Flashlight* filters the less valuable monitoring information from the raw data set. Namely, *Flashlight* starts with removing the reducible data which is listed in Table 1. In this case, the data reduction procedure is only applicable on D_1 where d_{11} is marked as reducible. As a result, the intra-relation data association can be applied on three different data pairs as $d_{12} \uplus d_{13}$, $d_{21} \uplus d_{22}$ and $d_{31} \uplus d_{32}$.

[1]http://www.cvedetails.com
[2]https://nvd.nist.gov
[3]https://www.exploit-db.com

By (1)(2)(3), the association vector of $\mathcal{A}|_{d_{13}}^{d_{12}}$ is computed as:

$$sup(\mathcal{A}|_{d_{13}}^{d_{12}}) = \sigma(d_{12} \bigcap d_{13}) = 0.83$$

$$conf(\mathcal{A}|_{d_{13}}^{d_{12}}) = \frac{\sigma(d_{12} \bigcap d_{13})}{\sigma(d_{12})} = \frac{0.45}{0.83} = 0.54$$

$$amb(\mathcal{A}|_{d_{13}}^{d_{12}}) = \sqrt{\frac{0.45}{0.83 \cdot 0.62}} \cdot 0.83 = 0.85$$

Therefore, the relation R_1-based data association can be identified as $I_1 = (0.83, 0.54, 0.85)$. Similarly, the association vector of $\mathcal{A}|_{d_{22}}^{d_{21}}$ is computed as: $sup(\mathcal{A}|_{d_{22}}^{d_{21}}) = 0.34$, $conf(\mathcal{A}|_{d_{22}}^{d_{21}}) = 0.65$, and $amb(\mathcal{A}|_{d_{22}}^{d_{21}}) = 0.90$. Hence, the relation R_2-based data association can be identified as $I_3 = (0.34, 0.65, 0.90)$. Additionally, $\mathcal{A}|_{d_{32}}^{d_{31}}$ is computed as: $sup(\mathcal{A}|_{d_{32}}^{d_{31}}) = 0.25$, $conf(\mathcal{A}|_{d_{32}}^{d_{31}}) = 0.68$, and $amb(\mathcal{A}|_{d_{32}}^{d_{31}}) = 0.97$. Thus, the relation R_3-based association can be identified as $I_5 = (0.25, 0.68, 0.97)$.

Step 2. Weighted inter-relation data association: Based on the identified intra-relation data associations, *Flashlight* proceeds to execute the weighted inter-relation data association. As discussed in Section 4.2.2, the key point for performing inter-relation data association is to select the proper *subsequent* by considering the data association weight which is decided by the mean value of the data criticality of a data association. As a result, the weight of association $\mathcal{A}|_{d_{13}}^{d_{12}}$ can be computed by applying Formula (4) as

$$W(\mathcal{A}|_{d_{13}}^{d_{12}}) = (c_{12} + c_{13})/2 = (0.53 + 0.41)/2 = 0.470$$

Similarly, the weights of association $\mathcal{A}|_{d_{22}}^{d_{21}}$ is $W(\mathcal{A}|_{d_{22}}^{d_{21}}) = 0.925$ and $W(\mathcal{A}|_{d_{32}}^{d_{31}}) = 0.775$ respectively. In terms of the computed weights, the selected *subsequent* for the *precedent* $\mathcal{A}|_{d_{13}}^{d_{12}}$ is $\mathcal{A}|_{d_{22}}^{d_{21}}$ rather than $\mathcal{A}|_{d_{32}}^{d_{31}}$, as $W(\mathcal{A}|_{d_{22}}^{d_{21}}) > W(\mathcal{A}|_{d_{32}}^{d_{31}})$. Therefore, the vector of the inter-relation data association of $\mathcal{A}|_{d_{13}}^{d_{12}} \uplus \mathcal{A}|_{d_{22}}^{d_{21}}$ can be computed as: $sup(\mathcal{A}|_{d_{13}}^{d_{12}} \uplus \mathcal{A}|_{d_{22}}^{d_{21}}) = 0.45$, $conf(\mathcal{A}|_{d_{13}}^{d_{12}} \uplus \mathcal{A}|_{d_{22}}^{d_{21}}) = 0.44$, and $amb(\mathcal{A}|_{d_{13}}^{d_{12}} \uplus \mathcal{A}|_{d_{22}}^{d_{21}}) = 0.95$. The inter-relation ($R_1$ and R_2) data association $\mathcal{A}|_{d_{13}}^{d_{12}} \uplus \mathcal{A}|_{d_{22}}^{d_{21}}$ is identified as $I_2 = (0.45, 0.44, 0.95)$ that is monitoring path p_1 for helping monitor the Heartbleed attack.

The obtained monitoring path p_1, which consists of four monitoring data as the Port 443 request amount (d_{12}), the HTTPS request amount (d_{13}), the odd-place login amount (d_{21}), and the privilege-required operation amount (d_{22}), can be interpreted as follows. By taking advantage of causality (i.e., relation R_1), the association between the HTTPS requests and the OpenSSL responses is identified as $I_1 = (0.83, 0.54, 0.95)$ where the high *sup* and *conf* value indicates the evident co-appearance between HTTPS requests and the corresponding responses observed on port 443 (the well-known port for HTTPS protocol). In addition, the *amb* value is 0.95 which refers to a very strong correlation between the two behaviors. By leveraging the service dependency (i.e., relation R_2), the association between the odd-place logins and the unexpected privilege escalations is identified as $I_3 = (0.34, 0.65, 0.90)$ that indicates a strong correlation ($amb = 0.90$) existing between the two behaviors at a noticeable rate ($sup = 0.34$, $conf = 0.65$). It implies the

Table 1: Case Study: An Excerpt of The Collected Monitoring Data Set for Conducting Monitoring Path Identification

The Information of the Collected Monitoring Data Set							
Data ID	d_{11}	d_{12}	d_{13}	d_{21}	d_{22}	d_{31}	d_{32}
Data Name	TCP Req.	Port 443 Reqst.	HTTPS Resp.	odd-place logins	Privilege ops.	Proc. Overhead	NW. Overhead
Relation	R_1	R_1	R_1	R_2	R_2	R_3	R_3
Data Value	v_{11}	v_{12}	v_{13}	v_{21}	v_{22}	v_{31}	v_{32}
Data Reducibility	reducible	irreducible	irreducible	irreducible	irreducible	irreducible	irreducible
Data Criticality	0.30	0.53	0.41	0.87	0.98	0.79	0.76
Data Appearance	N.A	0.83	0.62	0.34	0.27	0.25	0.18
Data Appearance	N.A	0.45		0.22		0.17	
Data Appearance	N.A	0.20				N.A	
Data Appearance	N.A	N.A		0.10			

privilege-required operations (e.g., service subscription alteration or user password modification) performed by the "CSC" logged in from odd places with correct password. Moreover, the inter-relation association of the two identified associations is computed as $I_2 = (0.45, 0.44, 0.95)$ which reveals that a portion (sup = 0.45, $conf$ = 0.44) of the user requests (i.e., the port 443 activities) are very likely to result in these privileged operations (amb=0.95).

5.2.2 ***Case II: The SambaCry Attack.*** In practice, many undisclosed threats that can stealthily undermine the security compliance of cloud services are hard to be monitored as no knowledge about the threats is available. For instance, the SambaCry attack is a recently exposed security threat that can violate the cloud service's security compliance [22]. The attack exploits the vulnerability of Samba services (version $3.5.x \sim 4.6.4$) that is widely deployed in cloud for offering file and printing services. By uploading a crafted library encapsulated with malicious codes to a writable shared folder, an attacker can remotely subvert the cloud systems and perform arbitrary operations (e.g., privilege escalations). Nevertheless, it is hard to monitor such a threat in cloud due to a lack of the specific attack details. To demonstrate *Flashlight*'s merit of identifying the monitoring path for helping to monitor unknown security threats with existing monitoring information, we conduct a case study on identifying the monitoring path regarding the abstracted SambaCry attack that simulates an unknown cloud security threat.

To support indirectly monitoring the threat, *Flashlight* identifies the monitoring path with the collected information as follows.

Step 1. Reducible intra-relation data association: With the obtained association information (i.e., I_1, I_3, and I_5) derived in the previous case study, *Flashlight* does not need to rerun the intra-relation association process and thus can directly proceed to carry out the inter-relation data association.

Step 2. Weighted inter-relation data association: *Flashlight* performs the weighted inter-relation data association by considering $\mathcal{A}|_{d_{32}}^{d_{31}}$ as the *precedent*, as a strong data correlation between the process overhead (d_{31}) and the network traffic (d_{32}) is indicated by a high $amb(\mathcal{A}|_{d_{32}}^{d_{31}})$ value (0.97) in the computed I_5.

By comparing the identified intra-relation association weights, *Flashlight* selects $\mathcal{A}|_{d_{22}}^{d_{21}}$ as the *subsequent* for the $\mathcal{A}|_{d_{32}}^{d_{31}}$ based on the inequality of $W(\mathcal{A}|_{d_{22}}^{d_{21}}) = 0.925 > W(\mathcal{A}|_{d_{13}}^{d_{12}}) = 0.470$. Therefore, the weighted inter-relation data association vector for $\mathcal{A}|_{d_{22}}^{d_{21}} \biguplus \mathcal{A}|_{d_{32}}^{d_{31}}$ is computed as: $sup(\mathcal{A}|_{d_{22}}^{d_{21}} \biguplus \mathcal{A}|_{d_{32}}^{d_{31}}) = 0.22$, $conf(\mathcal{A}|_{d_{22}}^{d_{21}} \biguplus \mathcal{A}|_{d_{32}}^{d_{31}}) = 0.45$, and $amb(\mathcal{A}|_{d_{22}}^{d_{21}} \biguplus \mathcal{A}|_{d_{32}}^{d_{31}}) = 0.77$. Thus, the inter-relation (R_2 and R_3) data association $\mathcal{A}|_{d_{22}}^{d_{21}} \biguplus \mathcal{A}|_{d_{32}}^{d_{31}}$ is identified as $I_4 = (0.22, 0.45, 0.77)$ which is regarded as the monitoring path p_2 for supporting indirectly monitoring the SambaCry attack.

The identified monitoring path p_2, which contains monitoring data as the odd-place login amount (d_{21}), the privilege-required operation amount (d_{22}), the process overhead (d_{31}), and the network traffic (d_{32}), can also be explained as follows. By examining the abnormality (i.e., relation R_3), the association between the high overhead of a particular process and the high network traffic is identified as $I_5 = (0.25, 0.68, 0.97)$ where the high amb value 0.97 refers to a very strong correlation between them at an unnoticeable rate (sup=0.25, $conf$=0.68). As aforementioned, the association between the odd-place logins and the privilege-required escalation is identified by leveraging the service dependency (i.e., relation R_2) as $I_3 = (0.34, 0.65, 0.90)$ that reveals the privilege-required operations caused by the users log in from odd places. The inter-relation association between the two data associations is identified as $I_4 = (0.22, 0.45, 0.77)$ which indicates the abnormal high traffic closely correlating with the odd-place login user's behavior (amb=0.77) of initiating the particular brute-force attack process secretly (sup=0.22, $conf$=0.65). Specifically, the attack is stealthily mounted by subverting the Samba service with uploading a crafted library file of attack codes to the user's writable shared folders.

5.2.3 Monitoring path representation. *Flashlight* introduces a novel schema termed monitoring property graph for representing the identified monitoring paths p_1 and p_2 together with related monitoring information over the given monitoring data set in Figure 5. With the monitoring property graph, related monitoring information can be comprehensively characterized in a quantitative way. Every collected monitoring data is represented by a vertex annotated with a key-value pair where the key is the data ID and the value is a vector representing the data value and data attributes (reducibility and criticality). For example, the monitoring data of an HTTPS response message is denoted by a key-value pair as $d_{11} : (v_{11}, reducible, 0.30)$, where d_{11} is the key and $(v_{11}, reducible, 0.30)$ is the value key representing the data value (v_{11}), the data attributes (reducibility = $reducible$ and criticality

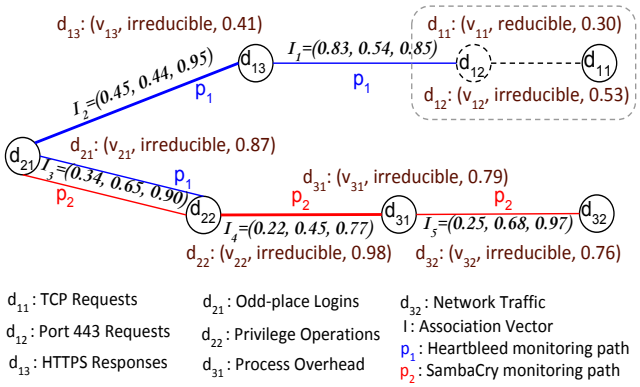

Figure 5: The Generated Monitoring Property Graph

= 0.30). Noticeably, the data reduction is shown as a dotted line representing the reducible data d_{11} and its association with d_{12} in Figure 5. Besides, every data association is represented by an edge annotated with an association vector. For instance, data association between d_{12} and d_{13} (i.e., $A|_{d_{13}}^{d_{12}}$) is denoted by an edge annotated by the identified association vector $I_1 = (0.83, 0.54, 0.85)$ where each value respectively represents the value of *support (sup), confidence (conf),* and *ambiguity (amb).* Moreover, the identified monitoring paths can also be represented in this graph, namely monitoring path $p_1 : A|_{d_{13}}^{d_{12}} \uplus \mathcal{A}|_{d_{22}}^{d_{21}}$ (highlighted with blue color) and $p_2 : A|_{d_{22}}^{d_{21}} \uplus \mathcal{A}|_{d_{32}}^{d_{31}}$ (highlighted with red color). Both monitoring paths consist of four different monitoring data respectively.

Discussions. The case studies demonstrate *Flashlight*'s advantages of identifying monitoring paths in different situations. The details of running the indirect cloud monitoring approach with the identified monitoring paths (p_1 and p_2) follow the descriptions in [40] and are not further detailed here. Besides, *Flashlight* also has several characteristics which are worth mentioning as follows.

Flashlight takes advantage of the basic knowledge (i.e., data relations and data attributes) to identify monitoring paths which are challenging to be achieved by pure manual analysis. As security experts require to have very in-depth knowledge for performing the manual analysis [19], it is very likely for the experts to overlook some important monitoring paths without sufficient expertise.

While machine learning approaches (like principal components analysis) can also be used for identifying data associations, the learning process is not only incurring tremendous computational overhead but also subject to the scale of the monitoring data set. As the scale/complexity increment of cloud systems, its applicability is substantially exacerbated. By contrast, *Flashlight* adopts a statistical-based approach for identifying monitoring paths to circumvent the aforementioned challenges. In fact, *Flashlight* supports executing the intra-relation association processes in a parallel mode which is particularly suitable for applying in cloud scenarios.

Besides, *Flashlight* can even support fine-grain monitoring path identifications by leveraging the monitoring property graph. Specifically, when monitoring data involves multiple data relations, the multi-graph property of monitoring property graph enables to replace the vertex representing that multi-relation data with several *virtual vertices* where each *virtual vertex* represents the monitoring data involving one single data relation. Then, the monitoring path identification process can be performed as usual.

Overall, *Flashlight* is proposed for working in a static scenario where all monitoring data stays unchanged during the path identification process. However, it can be adapted for managing the dynamic scenario as the static scenario can be regarded as a snapshot of the monitoring status in a dynamic cloud system at a specific time instance. The compiled set of such snapshots (at predefined intervals) can help capture transitional behaviors.

Threats to validity. While *Flashlight* can theoretically manage to identify monitoring paths at any length, proper thresholds on the *support*/*confidence*/*ambiguity* should be predefined for obtaining meaningful monitoring paths. As the increment of the path length, the co-appearance rate of monitoring data keeps dropping down. Without the appropriate threshold settings, the performance of *Flashlight* might be undermined as it costs excessive overhead to identify an unnecessary overlong monitoring path.

6 RELATED WORK

Monitoring path identification for securing cloud services is a novel topic with little specific coverage and with existing related work focused on general cloud monitoring and the generic path discovery.

The existing cloud monitoring approaches developed by taking advantage of different factors (e.g., data relations, timing characteristics, or topology properties) are applicable only when the monitoring parameter/data is directly accessible. For example, Deng et al. [5] proposed an access policy-based security tracing framework called *LACT* to monitor leaked access credentials with respect to cloud storage services. However, this framework requires all the relevant monitoring data to be accessible. Du et al. [9] proposed the *ROSIA* framework to monitor malicious service providers within a multi-tenant cloud system by examining the consistency of data flow. Nevertheless, *ROSIA* requires the whole data flow to be accessible. A large number of monitoring methodologies (e.g., *RAFT*[4], *PhisEye*[13], *MaaS*[16]) are proposed for monitoring different aspects of cloud services (e.g., service elasticity, transient violations, or data robustness), while these methodologies become invalid when losing the direct access to the related monitoring data. Many other monitoring methodologies (e.g., [2, 29, 34]) are developed by adopting a distributed model to monitor cloud services by running a series of local cloud monitoring tasks. Unfortunately, the proposed methodologies also require direct access to all needed monitoring information locally.

Even though not directly related to the topic, we found some research works which might be helpful for addressing the monitoring path identification question. A small number of approaches are proposed for dealing with the path discovery task on an ad-hoc basis. For instance, *CloRExPa* [6] is proposed to trace the malicious data modifications inside cloud systems by making use of the proposed virtual machine state/action graph. In a cloud scenario, it is challenging to construct the state/action graphs when necessary information is unavailable. Ravindranath et al. [25] proposed a discovery technique to capture the path of the mobile user

transaction by examining the time latency. Yet, this technique is developed for mobile systems rather than cloud systems. Several discovery approaches (e.g., [12, 15, 19]) are proposed by analyzing relevant statistical properties for locating obfuscated sequences, searching demanded content, or reconstructing attack scenarios, respectively. While, these approaches are inapplicable for the cloud scenario where the analysis process takes a lot of effort. Besides, some discovery techniques are developed by adopting taint-based methods. Naderi et al. [17] proposed a hybrid taint-based approach to discover SQL injections, while Yamaguchi et al. [38] proposed a taint-based approach to discover the patterns of unknown vulnerabilities. However, it is hard to perform the taint process on the monitoring data collected by security monitors from a monitoring path identification perspective. Various inference-based discovery methodologies are also proposed. Olivo et al. [23] proposed an inference methodology to infer a new type of Denial of Service (DoS) attacks by using particular database attributes. Unfortunately, the methodology is unsuitable for working in a complex cloud scenario, as it is hard to obtain all the required information. Additionally, Yadwadkar et al. [37] proposed an approach to infer victim cloud tasks by using support vector machine to study system features, but the approach incurs expensive system overhead for performing the machine learning process and the feature selection process.

7 CONCLUSION

Indirect cloud monitoring is an emerging and promising security approach to effectively monitor cloud services where particular monitoring information is inaccessible. To indirectly monitor cloud services, the prerequisite is to perform monitoring path identification which is a novel topic and still in an incipient stage in the research community. In this paper, *Flashlight* is proposed as a systematic approach for addressing the monitoring path identification problem by synthetically leveraging the complex relations and attributes of collected monitoring data. By performing the case studies on real security threats, *Flashlight* demonstrates its efficacy of identifying monitoring paths for security professionals indirectly monitoring cloud services without directly accessing to particular monitoring information.

The advantages of our proposed approach are the capability of managing different types of data relations and the possibility to deal with various path identification scenarios. In the future, we plan to automate *Flashlight*'s monitoring path identification process. Overall, this paper offers a good starting point for investigating the monitoring path identification question for securing cloud services.

8 ACKNOWLEDGMENTS

Research supported in part by EC H2020 CIPSEC GA #700378 and BMBF TUD-CRISP.

REFERENCES

[1] Agrawal R., et al. 1994. Fast algorithms for mining association rules. In *VLDB*, Vol. 1215. 487–499.
[2] Agrawal S., et al. 2007. Efficient detection of distributed constraint violations. In *ICDE*. IEEE, 1320–1324.
[3] Amazon. 2017. List the Available CloudWatch Metrics for Your Instances. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/viewing_metrics_with_cloudwatch.html. (2017). [Online].
[4] Bowers K., et al. 2011. How to tell if your cloud files are vulnerable to drive crashes. In *CCS*. ACM, 501–514.
[5] Deng H., et al. 2014. Who is touching my cloud. In *ESORICS*. Springer, 362–379.
[6] Di Pietro R., et al. 2014. CloRExPa: Cloud resilience via execution path analysis. *Future Generation Computer Systems* 32 (2014), 168–179.
[7] di Vimercati S., et al. 2014. Fragmentation in presence of data dependencies. *TDSC* 11, 6 (2014), 510–523.
[8] Dierks T and Allen C. 1999. The TLS Protocol, Version 1.0. https://www.ietf.org/rfc/rfc2246.txt. (1999). [Online].
[9] Du J., et al. 2010. On verifying stateful dataflow processing services in large-scale cloud systems. In *CCS*. ACM, 672–674.
[10] Giuseppe B., et al. 2010. Measurement data reduction through variation rate metering. In *INFOCOM*. IEEE, 1–9.
[11] Gullasch D., et al. 2011. Cache games–bringing access-based cache attacks on AES to practice. In *Security and Privacy*. IEEE, 490–505.
[12] Du H. and Yang S. 2014. Probabilistic inference for obfuscated network attack sequences. In *DSN*. IEEE, 57–67.
[13] Han X., et al. 2016. Phisheye: Live monitoring of sandboxed phishing kits. In *CCS*. ACM, 1402–1413.
[14] Hiller M., et al. 2001. An approach for analysing the propagation of data errors in software. In *DSN*. IEEE, 161–170.
[15] Liu C., et al. 2014. Path knowledge discovery: Association mining based on multi-category lexicons. In *BigData*. IEEE, 1049–1059.
[16] Meng S and Liu L. 2013. Enhanced monitoring-as-a-service for effective cloud management. *IEEE Trans. Comput.* 62, 9 (2013), 1705–1720.
[17] Naderi-Afooshteh A., et al. 2015. Joza: Hybrid taint inference for defeating web application sql injection attacks. In *DSN*. IEEE, 172–183.
[18] Ning J., et al. 2015. Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud. In *ESORICS*. Springer, 270–289.
[19] Ning P., et al. 2002. Constructing attack scenarios through correlation of intrusion alerts. In *CCS*. ACM, 245–254.
[20] NIST. 2001. Announcing the advanced encryption standard (AES). *Federal Information Processing Standards Publication* 197 (2001), 1–51.
[21] NVD. 2014. CVE-2014-0160. https://nvd.nist.gov/vuln/detail/CVE-2014-0160. (2014). [Online].
[22] NVD. 2017. CVE-2017-7494. https://nvd.nist.gov/vuln/detail/CVE-2017-7494. (2017). [Online].
[23] Olivo O., et al. 2015. Detecting and exploiting second order denial-of-service vulnerabilities in web applications. In *CCS*. ACM, 616–628.
[24] Papenbrock T., et al. 2015. Functional dependency discovery: An experimental evaluation of seven algorithms. *VLDB* 8, 10 (2015), 1082–1093.
[25] Ravindranath L., et al. 2012. AppInsight: Mobile App Performance Monitoring in the Wild.. In *OSDI*, Vol. 12. 107–120.
[26] Ristenpart T., et al. 2009. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *CCS '09*. ACM, 199–212.
[27] Rivest R., et al. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.
[28] Rodriguez M. and Neubauer P. 2010. The graph traversal pattern. *arXiv preprint arXiv:1004.1001* (2010).
[29] Saemundsson T., et al. 2014. Dynamic performance profiling of cloud caches. In *SoCC*. ACM, 1–14.
[30] Shao J., et al. 2010. A runtime model based monitoring approach for cloud. In *CLOUD*. IEEE, 313–320.
[31] Tan P., et al. 2000. Indirect association: Mining higher order dependencies in data. *Principles of Data Mining and Knowledge Discovery* (2000), 212–237.
[32] Varadarajan V., et al. 2012. Resource-freeing attacks: improve your cloud performance (at your neighbor's expense). In *CCS*. ACM, 281–292.
[33] Wang J., et al. 2015. Discover and Tame Long-running Idling Processes in Enterprise Systems. In *ASIACCS*. ACM, 543–554.
[34] Wu Y., et al. 2013. EagleEye: Towards mandatory security monitoring in virtualized datacenter environment. In *DSN*. IEEE, 1–12.
[35] Wu Z., et al. 2012. Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud.. In *USENIX Security symposium*. 159–173.
[36] Xu Z., et al. 2016. High fidelity data reduction for big data security dependency analyses. In *CCS*. ACM, 504–516.
[37] Yadwadkar N., et al. 2014. Wrangler: Predictable and faster jobs using fewer resources. In *SoCC*. ACM, 1–14.
[38] Yamaguchi F., et al. 2015. Automatic inference of search patterns for taint-style vulnerabilities. In *Security and Privacy*. IEEE, 797–812.
[39] Zhang H., et al. 2014. Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery. In *ASIACCS*. ACM, 39–50.
[40] Zhang H., et al. 2017. deQAM: A Dependency Based Indirect Monitoring Approach for Cloud Services. In *SCC*. IEEE, 27–34.
[41] Zhang Y., et al. 2014. Cross-tenant side-channel attacks in PaaS clouds. In *CCS*. ACM, 990–1003.