Data Security and Malware Detection in Cloud Storage Services

Ting Cao, Jianzhou Mao, Tathagata Bhattacharya, Xiaopu Peng
Wei-Shinn Ku, and Xiao Qin
Department of Computer Science and Software Engineering
Auburn University, Auburn, AL.
{tzc0028, jzm0122, tzb0063, xzp0007, weishinn, xqin}@auburn.edu

***Abstract*—Cloud storage systems facilitate a platform to store and manage cyberspace data for a wide range of applications. In cloud storage connected to the Internet, a large amount of data is uploaded and accessed by numerous users. Thus, security and privacy of data is of utmost importance to end users regardless of the nature of the data being stored in clouds. After outlining the development of cutting-edge of cloud storage systems, we elaborate data security issues in cloud storage. We pay particular attention to malware detection techniques customized for cloud storage. The overarching goal of our solution is to guarantee that data are malware free in cloud storage prior to being accessed by end users. Inspired by the architecture of cloud storage, we propose a popularity-aware malware detection strategy to enhance the security of cloud storage systems by protecting high-risk data. Data risk is gauged through popularity, because popular data deserve high priority when it comes to access frequencies. Our proposed technique speculates data popularity, which is an avenue to prioritize data objects amid time-consuming malware detection procedures. Our technique is conducive to keeping malware at bay when popular data are frequently accessed by clients.**

***Index Terms*—cloud, popularity awareness, machine learning, malware detection, data security, data process scheduling**

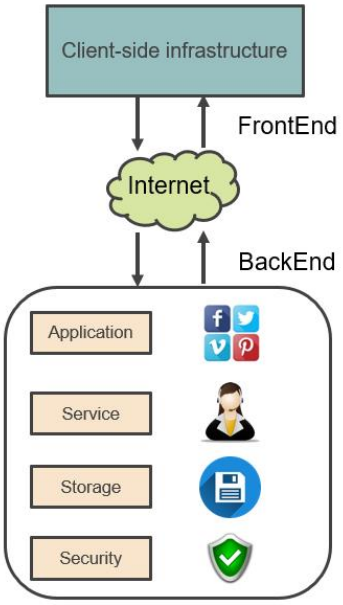

Fig. 1: Four lays of services in cloud computing environments.

I. Introduction

Security concerns pose a major challenge as cyber criminals compromise individual machines and network infrastructures to tamper with confidential data for financial gain or to launch denial-of-service attacks. More often than not, attackers take full advantage of malicious software or *malware* to impose serious threats and vulnerability of computing and network systems [1]. Malware detection systems are becoming an integral part of cloud and big-data storage systems.

With the advancement of cloud computing technologies, it is not hard to imagine that in the not-too-distant future a growing number of business applications will be moved into clouds. Fig. 1 shows a cloud computing architecture where there are four layers of diversified services. Malware may be designed to spread directly in clouds via various means. One key factor stimulating the spread of malware is the large number of accesses and downloads from users.

One demanding requirement of malware detection systems is to deliver superb performance amid the diagnosis of big data. After a thorough investigation of the well-known malware techniques, we observe that a many techniques are optimized by deploying machine learning algorithms in various fashions. We begin this paper by illustrating features of cloud computing, emerging techniques in cloud storage, and security issues in clouds. After that, we survey a diversity of malware detection techniques and algorithms. In particular, we examine the data processing and scheduling issues in cloud-oriented malware detection.

Existing malware detection techniques customized for cloud data storage systems are focused on either boosting high detection accuracy or lowering I/O cost. Based on the records from Indexing Service, intruders are interested in attacking popular data rather than non-popular ones; that is, unpopular data have semantic security but there is weaker security for popular data [2]. Therefore, popular data that are accessed frequently exhibit a higher risk than less popular counterparts. We speculate that launching malware detection techniques on high-risk data followed by low-risk one is an effective way keep cyber intrusions at bay in the realm of online services. Because popular data have much higher access frequency than those of non-popular data, it is urgent to detect malware among hot data prior to cold data that are less likely to be retrieved. We argue that popular data entail high risks, because storage

systems must repeatedly identify malware in hot data that are frequently accessed and manipulated by users.

An effective way to prevent the infection of popular data affecting numerous users is to predict data popularity from user preferences in each time slot. With the assistance of such a popularity predictor, the malware detector ensures that data are malware free before retrievals. In this study, we propose a popularity-aware malware detection scheduler to boost the security of online cloud computing systems by deploying a collaborative filtering algorithm.

Recommendation systems offer a prediction list to active users. In this study, we propose deployment of a recommendation algorithm to schedule malware detection sequence. Since active users keep changing all the time, depending on the preference of active users, the data popularity is not stable. The recommendation algorithm dynamically calculates data popularity for cloud storage through active users' access history. In this case, the malware detection scheduler distinguishes popular data and unpopular data based on the real access possibility of active users.

To speed up malware detection in cloud storage, we devise a popularity-aware malware detection system that seamlessly integrates two distinct modules - a popularity predictor and a malware detector. The first module makes data popularity prediction possible by adopting the *user-based collaborative filtering* algorithm or UBCF [3][4]. The second module is in charge of detecting malware in data objects on the basis of future popularity. Hot data that are likely to be frequently accessed receive a high priority to go through malware screening. With the assistance of the popularity predictor, the malware detector ensures that data are malware free before being retrieved.

Two reasons motivate us to make popular data high priority during the course of malware detection. First, popular data are accessed by a large number of users. When a popular data object is approved to be malware-free, all requests accessing the data are protected from malware. Any malware-free data that are popular can immediately benefit a large group of users. Second, popular data are likely to be accessed in the not-too-distant future, making it urgent to identify malware from the popular data. If malware detection is not carried out on popular data in a timely manner, the data may pose potential security threats to users. Our proposed malware detection scheduler ensures that popular data pass through a thorough malware detection earlier than unpopular ones.

The overarching goal of our scheduler is to optimize a malware detection sequence of data objects by deploying a machine-learning-enabled recommendation algorithm to prioritize high-risk data in cloud storage systems. More specifically, a UBCF-based management module periodically sorts data objects according to popularity. The schedule works in full capacity to dynamically set up the priorities of data objects with respect to popularity measures.

The rest of this paper is organized as follows. Section II provides an overview of cloud storage systems. In Section III, we articulate various malware detection, data processing, and scheduling techniques in cloud storage. Section IV outlines our proposed popularity-aware malware detection scheduler. We conclude this paper in Section V.

II. Cloud Storage Systems

A. Features of Cloud Computing

Clouds furnish highly scalable computing platforms by the virtue of virtualized resources shared among users. In general, a cloud computing architecture is divided into two layers, namely, a bottom resource layer and an upper service layer. The bottom resource layer, an underpinning for computing clouds, is fueled by virtualized resources in the form of storage and computing components. The upper service layer is intended to offer a diversity of specific services. Fig. 2 illustrates the basic architecture of cloud computing systems.

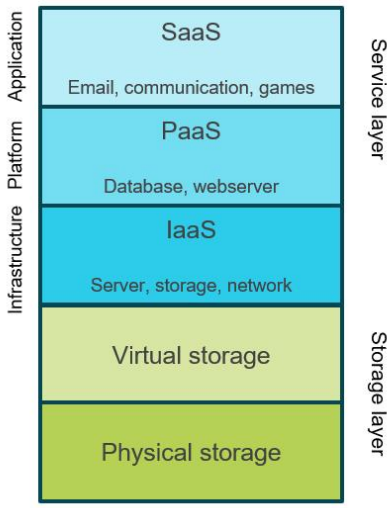

Fig. 2: A cloud computing architecture. The bottom layer is fueled by virtualized storage and computing resources, whereas the upper service layer offers a diversity of services.

Cloud computing judiciously combines data-sharing models and service statistical models. From a technical point of view, cloud computing can be categorized into the following three models [5].

- *Software as a Service* (SaaS) facilitates software services on the cloud, where a single occurrence of a software application can be used by multiple users or organizations over the cloud.
- *Platform as a Service* (PaaS) provides a cloud-based computing environment to scaffold the development of a variety of services. A PaaS sometimes embraces a rich set of libraries and Application programming interface (API), from which applications and services can be readily built.
- *Infrastructure as a Service* (IaaS) is comprised of essential computing and storage services over network infrastructures. Sample network infrastructures provided in IaaS include, but are not limited to, physical computing resources, location, security, scaling, data partitioning, and backup.

Regardless of the aforementioned three models, cloud services share the following distinctive characteristics [5].

- *On Demand Self-service.* Services enable provisioning of cloud resources to vendors on demand or whenever the resources like network storage are required without human interaction.
- *Broad Network Access.* Cloud computing capabilities are offered through network infrastructures, which are connected by heterogeneous client platforms such as laptops, tablets, and smart phones.
- *Resource Pooling.* Resources of cloud providers are pooled over computing servers. Consumers are assigned various resources - physical or virtual ones - without being aware of detailed locations of the assigned resources except at an abstract level (e.g., state, city, country or data center).
- *Rapid Elasticity.* Services are elastically provisioned in an automatic fashion, thereby rapidly scaling outward and inward in accordance with dynamically changing demand. Cloud computing capabilities, from the perspective of users, appear to be unlimited at any time period.
- *Measured Services.* Computing clouds dynamically manage and optimize resources by proactively monitoring and reporting usages, which are transparent to users utilizing the services.

B. Emerging Techniques in Cloud Storage

From the access-scope perspectives, cloud computing systems are categorized into three families, namely, public clouds, private clouds, and hybrid clouds. Public clouds perform as service providers to the general public; private clouds are referred to as the computing property owned by a company; hybrid clouds can be envisioned as a blend of public and private clouds. Most of the existing cloud services are provided by top-notch companies such as Google, Amazon, and IBM. In a private cloud, only authorized users can access computing services from the provider. In a pubic cloud, on the other hand, anybody is enabled to make use of cloud services. The hybrid cloud contains the mixed concept of both public and private clouds.

Cloud storage systems mainly offer data storage as a service to users through a unified interface. Users can easily store, retrieve, and manipulate data in a large-scale cloud storage infrastructure. Moreover, the storage service can be subscribed at any time on a pay as you go basis; the service is released freely when it is no longer needed. A wide range of storage services are governed by service-level agreements or SLAs configured between users and service providers.

A variety of emerging techniques have been advanced in the realm of cloud storage services. For example, Tawalbeh *et al.* proposed the master-cloudlet-based cloud computing model for mobile users [6]. Huo *et al.* developed the layered view for a cloud storage architecture, which delivers high parallel I/O performance [7]. ACDAS [8] investigated a data access model for cloud data storage systems, where the model achieves the security goals along with the practical efficiency of storage, computation, and communication compared with the other related schemes. Xu *et al.* devised a scalable data storage and sharing system for the Internet of Things (IoT) cloud, in which a ciphertext-policy attribute-based encryption scheme is implemented and deployed [9].

C. Security Issues in Cloud Storage

Security concerns should be rectified to make a cloud environment trustworthy for individual and enterprise users. A trustworthy environment becomes the vital prerequisite to win confidence of users to adopt the cutting-edge cloud storage technology. Cloud providers are aiming to boost system resource utilization while cutting down operational cost. On the flip side, consumers have a desire to operate resources as far as needed while being able to expand or shrink resources consumption based on dynamic demands. The cloud computing model fulfills user requirements by delivering two key characteristics - multitenancy and elasticity. Multitenancy refers to the concept of sharing computational and storage resources as well as services and applications among a group of tenants. Elasticity implies the capability of scaling up or down resources on the fly to meet dynamically changing demands. Scaling up and down of resources for a tenant offers ample opportunities to the other tenants to utilize previously assigned resources [10]. Confidentiality issues should be addressed from the perspectives of both multitenancy and elasticity. Traditional approaches to solving such a problem largely rely on the theories of isolation [11] and location transparency [12].

In what follows, we point out an array of challenges to be examined in the development of cloud storage.

- The number of users changes dynamically in cloud storage. By the same token, services requested by users drastically change over time. Such a dynamic nature of user behaviors makes it extremely difficult if not futile to carry out user classifications [13].
- New service delivery models of cloud computing suggest that cloud services and resources may be jointly owned and handled by multiple providers. Due to various security policies implemented by multiple parties, deploying unified security measures becomes a nontrivial task [14].
- Protection for user data. This issue includes location of user data stored, the way of data storage, data recovery, data encryption and data integrity protection [15].

Cloud service providers have to proactively keep track of data health. To prevent data from being corrupted by malicious code, continuously monitoring and protecting data become necessities. Such a safeguard can guarantee continuous data availability to users in the event of any faulty storage nodes due to accidents and natural calamity [16]. Moreover, user-cloud interactive is a noteworthy factor affecting the overall security of clouds. To address the above challenges, we propose in Section IV an approach to boosting the malware-detection efficiency in cloud storage systems.

III. Detecting Malware in Cloud Storage

Machine learning techniques are widely adopted in the malware detection field [17]. We start this section with a brief introduction on leading-edge malware detection techniques fueled by machine learning algorithms (see Section III-A). Then,

we elaborate a handful of advanced data processing techniques applied in cloud computing systems in Section III-B.

A. Machine-Learning-Based Malware Detection in Clouds

Malware detection techniques generally fall into two camps, namely, static analysis [18] and dynamic analysis approaches [19]. In what follows, we articulate malware detection solutions from these two angles - static and dynamic malware detection. Fig. 3 summarizes the high-level procedures of static and dynamic malware detection schemes.

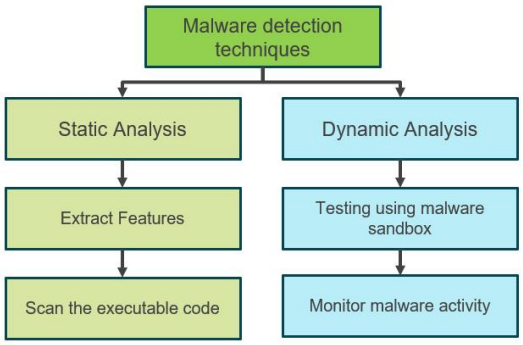

Fig. 3: The high-level procedures of static and dynamic malware detection schemes.

Static Analysis Approaches. Amid a static analysis of malware detection, no execution of executable programs takes place. The goal of static-analysis-based approaches is to finish up malware detection tasks in a swift manner without being interrupted and slowed down by third-party programs. Such malware detection solutions entail the process of analyzing executable files by examining the code without executing them. A static analysis procedure is comprised of two steps. First, an executable file is disassembled or reverse engineering disassembled to retrieve the code. Next, detection of malware is carried out by scanning the executable code derived from the previous step. Smart malware may evade static-analysis-based methods by embedding syntactic code errors that mislead disassembly while performing functions during executions. Alternatively, an analysis can be accomplished by looking through executable binary files followed by applying machine-learning-based detectors to diagnose malicious software [20].

Deep learning models have been introduced into malware detection systems [20][21][22], where malware files are diagnosed without being executed. When it comes to online malware detection, most static analysis schemes found in the literature cope with single samples without addressing the mislabeling problem or time windows of identifying malicious patterns. [23]

Dynamic Analysis Approaches. Unlike static analysis counterparts, dynamic analysis approaches are normally deployed in an isolated environment such as sandboxes or virtual machines (VM). In such detection schemes, information is gathered during executions like system calls, memory accesses, and network communications. The family of dynamic analysis solutions is applicable for malware-file classification in addition to online malware detection. It is noteworthy that online malware detection is closely related to intrusion detection systems [24].

Dahl *et al.* proposed a dynamic analysis solution to collect features from malware code that runs in a lightweight virtual machine [25]. Unfortunately, this approach is inadequate for online malware detection. Abdelsalam *et al.* devised an online malware detection system in clouds. This online detection system makes use of convolutional neural network (CNN) to maintain an optimal number of running virtual machines according to dynamic workload. More specifically, the number of virtual machines is dynamically scaled up or down based on load incurred malware detection. The online detection system is quite practical, because the system detects malicious behaviors while the other applications keep running on clouds. In a nutshell, machine-learning algorithms such neural network techniques are widely and judiciously employed to detect malware in clouds [26][17].

More often than not, hackers embark on large-scale distributed denial of service attacks or DDoS through malware code, phishing, and email spamming [27]. Many prior studies (see, for example, [9]) have addressed cloud security issues from various aspects such as networks, hypervisors, virtual machines, and operating systems, to name just a few. These cutting-edge security solutions are derived from rule-based intrusion detection systems accompanied by statistical anomaly detection models.

B. Data Processing and Scheduling

Existing malware detection techniques are mainly focused on accuracy, I/O resources, and online/offline strategies. Due to exploiting vulnerabilities in web browsers, however, a drive-by download is likely to fetch malicious codes from the Internet followed by executing the codes on victims' local machines [28]. Evidence clearly indicates that approximately 70-80% of malware codes in software originate from data downloading processes in popular websites [29].

Thanks to online services, cloud computing systems are widely used to support a variety of application domains. Active users may access cloud-based applications at any time from anywhere through the Internet. There is a dire demand to embark on time-consuming malware detection procedures while users are accessing data from the clouds. There are two concurrent research paths toward tackling this challenge.

- First, resources must be fittingly partitioned and allocated among users, systems, and malware-detection services. Making a good tradeoff is the fundamental key in resource management for future malware detection systems in clouds. On the one hand, if the majority of resources are dedicated to malware detection, user experience will be dragged down. On the other hand, if we favor user response time by limiting resources for malware detection, data are likely to be accessed without malware diagnosis.
- Second, it is difficult, if not futile, to scan a massive amount of data to detect malware in big data arenas. This issue can be addressed by either detecting subsets of the big data or lowering the detection frequency. Data

selection algorithms should be developed to pick security-sensitive data from non-sensitive ones. A malware detection system ought to guarantee that high-risk data are diagnosed before being accessed by users. Frequency selection algorithms should be in charge of determining the most appropriate interval between two consecutive detection instances.

- Third, after security-sensitive data are elected for the malware detection procedure, high-risk data deserve to be handled in a swift manner. Given limited computing resources assigned to malware detection services, a scheduler has to be devised so that high-risk data will be processed before imposing any threat to users.

Data processing and scheduling are among the dominant methods to optimize the efficiency of malware detection systems deployed in a wide range of online cloud services. In what follows, we shed bright light on novel data processing and scheduling techniques aiming to improve system execution efficiency.

In the past few years, data-driven methods have been gaining popularity in detecting unknown attack patterns while keeping a low false alert level [30]. Among the data-driven approaches, a practical algorithm family is clustering and classifying outliers [31], which is adroit at addressing the weaknesses of knowledge-base intrusion and anomaly detection techniques. Supervised learning methods significantly outperform the unsupervised ones if the test data contain no unknown attacks.

Many data processing techniques boost system performance from multiple perspectives. For example, Mao *et al.* proposed *Decima* to optimize data processing clusters through reinforcement learning and neural network algorithms [32]. Recognizing that data preprocessing plays a vital role in anomaly detection, Davis and Clark investigated the data preprocessing techniques designed for anomaly-based network intrusion detection systems [33]. The findings show that data preprocessing predominantly depends on domain knowledge of experts to identify relevant parts of network traffic and to build candidate traffic features. Automation is kicked in to extract features for the purpose of data dimensionality reduction as well as feature selection.

When it comes to advanced scheduling techniques, Li *et al.* devised a predictive algorithm to assign threads to machines under the guidance of prediction results [34]. The novel scheduling algorithm pushes the distributed stream-data processing to the next high level by virtue of a topology-aware model facilitating performance prediction. Very recently, we have developed the *POST* system to schedule data reconstruction sequences in distributed storage systems, thereby substantially reducing average waiting time of I/O requests issued by active users [35].

Inspired by the above newly developed algorithms and techniques, we envision that scheduling mechanisms will be an immaculate game changer in enhancing the efficiency of modern malware detection systems. In the next section, we propose a research road map leading to popularity-aware malware detection, at the heart of which schedulers are positioned to speed up the detection process.

IV. Popularity-Aware Malware Detection Techniques

In Section IV-A, we start the research road map by presenting a basic idea for the development of popularity-aware malware detection techniques. We depict a system architecture for schedulers supporting popularity-aware malware detection in cloud computing platforms in Section IV-B. Then, we elaborate the concepts and main steps in Section IV-C. Finally, in Section IV-D we outline the scheduling algorithms that optimize the performance of malware detection systems.

A. Basic Idea

It is arguably true that cloud computing platforms can be jointly optimized by incorporating multiple dimensions like connection efficiency, access security, data placement, and scheduling. In this study, we pay heed to security issues centered around cloud storage systems. The evidence from the prior studies (see, for example, [36] and [37]) shows that they cannot unify the service because of the untrusted remote machines. In addition, any security solution customized for cloud storage is required to impose a negligible performance overhead. In our pilot study, we aim to demonstrate that scheduling techniques are capable of enhancing the performance of malware detection deployed in modern cloud storage.

The basic idea of our solution is to schedule a sequence of data objects in which malware are detected. Scheduling decisions should be made in a way that high-risk data are scanned by a malware detector in an early phase followed by low-risk data. When we set the high risk data object prior to low risk data on malware detection sequence, data security will be improved because the high-risk data experience a high intrusion possibility to systems and a high infection possibility to users.

One of the vital factors affecting the probability of successful intrusions is the number of access points [38]. As such, we advocate for gauging the risk of data objects using popularity measures. Compared with non-popular data, popular data objects have a high access frequency from active users. Popular data are treated as high-risk data because of the following two reasons.

- First, malware infection in popular data becomes a serious threat for enormous number of users. Each popular data object is retrieved by a large group of users, who will be victimized the malware codes.
- Second, there is a strong likelihood for popular data to be frequently accessed in a short time period. The malware detection system ought to ensure that the popular data are malware-free before being accessed by users.

For the above reasons, scheduling a detection sequence among data objects according to access popularity improves the data security of cloud storage systems by detecting malware of high-risk data in the first place.

B. System Architecture

Fig. 4 depicts the system architecture of our proposed malware detection scheduling system, where a malware detection

manager collaborates with a scheduler in clouds. The entire system is responsible for scheduling a detection list in which data objects are scanned for malware before being actively accessed by users. During the online malware detection procedure, the scheduler makes judicious decisions on a detection order, in which popular data are assigned high priorities to mitigate malware threats.

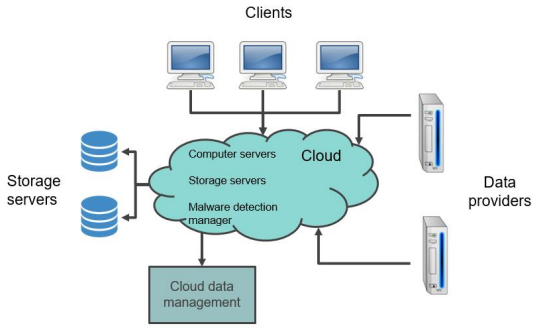

Fig. 4: The architecture of a malware detection scheduler running in clouds. Popular data objects receive a high priority to be scanned earlier than unpopular data objects.

In cloud computing environments, data are distributed across multiple storage nodes in data centers. Unsurprisingly, it is straightforward to incorporate prevalent malware detection techniques into cloud computing systems. For example, the leading-edge malware detection systems reported in [19] and [39] can be applied to fight malware in cloud storage services.

The overarching goal of the system is to optimize malware-detection performance by alleviating malware risks. To achieve this design goal, a management module is engaged to gauge data popularity measures, which are maintained as key components in metadata. Popular data objects are likely to be accessed by a growing number of users within in a short time period. It is arguably true that the scheduler in this architecture is adroit at governing the malware detection procedure, where detection priorities are configured in accordance to monitored data popularities. With the scheduler in place, popular data objects are handled by the detection module earlier than unpopular counterparts. Such scheduling decisions play a vital role in speeding up malware detection performance, because the time spent in identifying malware can be conserved by postponing the detection of unpopular data that impose low risks in clouds.

C. Concepts and Main Steps

To assess the popularity of data objects, we adopt a recommendation scheme in the malware-detection scheduling module. The recommendation scheme predicts a list of recommended data objects for each user who is actively accessing data from cloud storage. The malware-detection manager merges individual recommendation lists of multiple users into a single scheduling list for the malware detector in our system. Specifically, the manager carries out the four steps to consolidate multiple recommendation lists into a detection list, where the most popular data are scanned by the malware detector in an early stage. The data structure of detection lists is an array of object-weight pairs, which is defined below. Given object o_i and its weight $w_{x,i}$ with respect to user u_x, we express object-weight pair $\phi_{x,i}$ as:

$$\phi_{x,i} =< o_i, w_{x,i} > . \quad (1)$$

Table I summarizes a list of recommendation algorithms that can be plugged into the malware-detection scheduler. The popularity of data objects can be assessed by one of the following recommendation algorithms in Table I. Each recommendation algorithm has its unique advantages, depending on workload conditions such as number of active users, number of data objects, and the performance of cloud storage. As a case study, we import user based collaborative filtering as an underpinning technique to implement the popularity-aware scheduler.

Let us introduce the fundamental concepts of *key*, *key-value pairs*, *weight*, and *blocks* before diving into the description of the following four steps. We define data objects' identifiers (IDs) as *keys*, meaning that any data object can be readily referenced through its key. A data object is organized in the data structure of a *key-value* pair, where value is the content of the data object. We refer to the access frequency of a data object as a weight - an importance feature to capture the popularity of the data object. In a cloud storage system, data objects are basic storage units, which form large chunks called *blocks*. In other words, a data block is comprised of a group of data objects; all the data blocks share a fixed size. It is noteworthy that the block size can be configured by in the cloud storage, in which the default block size of our system is 64 MB.

- To retrieve data objects and the corresponding weights in a single user recommendation list to calculate the number of occurrences and weight for each key.
- Data blocks and data objects entail a two-layer data organization, where each data object belongs to a parent data block. The second step is to map the data objects' keys to their data blocks in the cloud storage.
- To calculate the summation of weights of each data block so that a detection list is constructed to embrace to-be-scanned data blocks accompanied by the corresponding weights. The weight of each data block indicates the block's popularity, which measures the future access frequency of the block.
- To schedule the items in the detection list according to the decreasing values of weights associated to the data blocks. In this step, data blocks with high weights are treated as popular data that are likely to be accessed by a large group of users in the not-too-distant future.

D. Algorithms

Algorithm 1 depicts the procedure of a popularity-based malware detection system in cloud storage, in which the above four steps (see Section IV-C) are carried out to schedule a detection list by merging all the recommendation lists predicted for active users in clouds.

The input information of Algorithm 1 includes user I/O access history and ratings as well as a set of data blocks in a cloud storage, which is formally defined as $B =$

recommendation type	basic idea	advantages	common algorithms
Content-based recommendation [40]	recommend an item to a user based on a description of the item and a profile of the user's interests	small number of structured attributes, simplicity, understandability	decision tree
Collaborative Filtering Recommendation [3] [41]	use the existing user's past behavior or comments to provide the product which conforms with the current user's requirements	good performance on large number of user and items	user based collaborative filtering item based collaborative filtering
Knowledge-based Recommendation [42] [43]	Systems that rely on knowledge sources of user requirements and domain knowledge	rely on knowledge sources that were not being employed by the more widely-used techniques	case-based recommendation constraint-based recommendation
Hybrid recommendation [44]	a combination of recommendation components or logic	high accuracy, easy to implement	Feature combination, weighted, switching

TABLE I: A list of candidate recommendation algorithms are readily plugged into the popularity-aware malware detection scheduler.

$\{b_1, b_2, ..., b_n\}$. Here, we assume the total number of blocks managed in the cloud storage is n.

The output of Algorithm 1 is a scheduled detection list $detList$, which contains an array of *block-weight* pairs. Ideally, the length of scheduled list $detList$ is identical to the size of set B. Thus, we have

$$detList.size() = |B| = n. \quad (2)$$

If system administrators opt for cutting back the overhead spent in diagnosing a large number of blocks, a subset of set B will be elected to originate scheduled list $detList$. Intuitively, increasing the length of list $detList$ can substantially raise the overhead of scheduling data blocks and detecting malware. List $detList$'s length ought to be appropriately chosen based on the system utilization and workload of the cloud storage.

Given data block b_k and its weight w_k, we define $\alpha_k = (b_k, w_k)$ as a block-weight pair for the kth block b_k. The detection list - an output of Algorithm 1 - is formally expressed as Eq. 3, where all the block-weight pairs are scheduled by the Algorithm in a decreasing order of the weights in the block-weight pairs.

$$detList = \{\alpha_1, \alpha_2, ..., \alpha_n\} = \{(b_1, w_1), ..., (b_n, w_n)\}, \text{ where } w_1 \geq w_2 \geq ... \geq w_n. \quad (3)$$

It is noteworthy that the weight of a block captures the popularity of data objects residing in the block. The most popular blocks are scheduled to be detected at the beginning of $detList$; the least popular ones are postponed toward the end of the list.

In Algorithm 1, Steps 1-3 repeatedly carry out user-based collaborative filtering to construct recommendation lists for the users in set U. More specifically, the collaborative filtering strategy is implemented by function $UbasedCoFiltering()$ in Step 2.

In our design, user-based collaborative filtering ought be performed in *Step 2* prior to establishing recommendation lists. This order is expected, because recommendation lists (see *Step 2*) are judiciously maintained by cloud storage regardless of the malware detection procedure. In a real-world cloud, the recommendation lists are proactively updated while data objects are being accessed by users.

Steps 4-6 control the initialization of the weights of the block-weight pairs in $detList$; the initial value of the weights is 0. Let U' represent a set of users who are actively accessing cloud storage amid the malware detection process. Because user information is retained in user set U, U' is a subset of U (i.e., $U' \subseteq U$). *Steps 7-14* repeatedly calculate each user's weights with respect to data objects. In particular, *Step 9* derives data object o_i from object-weight pair $\phi_{x,i}$ (see also the $GetDataObject()$ function). *Steps 10* and *11* obtain weight $w_{x,i}$ and block b_k from object o_i (see also the $GetWeight()$ and $GetBlock()$ functions). In *Step 12*, weight w_k is modified by augmenting intermediate result $w_{x,i}$ yielded from *Step 10*.

Finally, Step 11 sorts block-weight pairs in a non-increasing order of weights in detection list $detList$. Such a detection schedule is made by the $sort()$ function in the algorithm.

We reckon that the malware detection scheduler imposes relatively low computation overhead, because recommendation lists normally are originated prior to the malware detection procedure. In the worst case scenario, the lack of user requests (e.g., $U' = \emptyset$) makes it strenuous build a recommendation list to project data popularity. In this case, the system gracefully downgrades to offline malware detection, in which a detection schedule is no longer needed. In such an offline malware detection case, *Steps 7-14* In Algorithm 1 will be excluded.

Algorithm 1: The high-level controller of malware detection.

Input:
User I/O access history and rating records;
$B = \{b_1, b_2, ..., b_n\}$; /* data blocks to be detected */
Output:
$DetList = \{\alpha_1, \alpha_2, ..., \alpha_n\}$; /* A detection list */
1: **for** all $u_x \in U$ **do**
2: $P(u_x)$ = UbasedCoFiltering(u_x);
3: **end for**
4: **for** all $\alpha_k \in DetList$ /* Initialize $DetList$ */ **do**
5: SetWeight(α_k, 0); /* Initialize the weight of α_k */
6: **end for**
7: **for** all $u_x \in U'$ **do**
8: **for** all $\phi_{x,i} \in P(u_x)$ **do**
9: o_i = GetDataObject($\phi_{x,i}$);
10: $w_{x,i}$ = GetWeight($\phi_{x,i}$);
11: b_k = GetBlock(o_i, B);
12: detList[w_k]+= $w_{x,i}$;
13: **end for**
14: **end for**
15: Sort(detList[w_k]);
16: **return** $detList$;

V. Conclusions

We started this paper with an overview of cloud storage systems. After presenting a list of emerging techniques adopted

in cloud storage, we discussed cloud security issues to be addressed in data storage systems in clouds. The focus of our study is centered around malware detection in cloud storage. We illustrated multiple ways of applying machine learning solutions to efficiently identify malware codes in clouds. Next, we shed bright light on the challenges confronted in malware detection in the realm of cloud storage. To overcome these challenges, we proposed a popularity-aware malware detection system, which schedules a malware- detection sequence in a way that high-risk data are detected prior to low- risk counterparts in clouds.

At the heart of our proposed malware detection system, we designed a user-based collaborative filtering module to predict data popularity using established recommendation lists. We delineated the popularity-aware algorithm to (1) prioritize data blocks and (2) make detection schedules in a way to enhance the security of cloud storage systems. Along this line, we expect that a diversity of machine learning techniques can be employed to forecast data popularity, which in turn can determine malware-detection schedules. It is intriguing to quantitatively compare a handful of prediction solutions to figure out which one delivers the best performance for the malware detection system in cloud computing environments, where big data must be constantly scanned.

ACKNOWLEDGMENT

This work is supported in part by the U.S. National Science Foundation under Grants IIS-1618669, OAC-1642133, and CCF-0845257.

REFERENCES

[1] R. Anderson, C. Barton, R. Böhme, R. Clayton, M. J. Van Eeten, M. Levi, T. Moore, and S. Savage, "Measuring the cost of cybercrime," in *The economics of information security and privacy*. Springer, 2013, pp. 265–300.

[2] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data deduplication scheme for cloud storage," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 99–118.

[3] Z.-D. Zhao and M.-S. Shang, "User-based collaborative-filtering recommendation algorithms on hadoop," in *2010 Third International Conference on Knowledge Discovery and Data Mining*. IEEE, 2010, pp. 478–481.

[4] J. Hu, J. Liang, Y. Kuang, and V. Honavar, "A user similarity-based top-n recommendation approach for mobile in-application advertising," *Expert Systems with Applications*, vol. 111, pp. 51–60, 2018.

[5] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing," 2011.

[6] A. T. Lo'ai, W. Bakheder, and H. Song, "A mobile cloud computing model using the cloudlet scheme for big data applications," in *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. IEEE, 2016, pp. 73–77.

[7] Y. Huo, H. Wang, L. Hu, and H. Yang, "A cloud storage architecture model for data-intensive applications," in *2011 International Conference on Computer and Management (CAMAN)*. IEEE, 2011, pp. 1–4.

[8] D. Tiwari, G. K. Chaturvedi, and G. Gangadharan, "Acdas: Authenticated controlled data access and sharing scheme for cloud storage," *International Journal of Communication Systems*, vol. 32, no. 15, p. e4072, 2019.

[9] S. Xu, G. Yang, Y. Mu, and X. Liu, "A secure iot cloud storage system with fine-grained access control and decryption key exposure resistance," *Future Generation Computer Systems*, vol. 97, pp. 284–294, 2019.

[10] M. Almorsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem," *arXiv preprint arXiv:1609.01107*, 2016.

[11] I. Alobaidan, M. Mackay, and P. Tso, "Build trust in the cloud computing-isolation in container based virtualisation," in *2016 9th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2016, pp. 143–148.

[12] M. A. L. Peña and I. M. Fernández, "Sat-iot: An architectural model for a high-performance fog/edge/cloud iot platform," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 2019, pp. 633–638.

[13] X. Jing and Z. Jian-Jun, "A brief survey on the security model of cloud computing," in *2010 ninth international symposium on distributed computing and applications to business, engineering and science*. IEEE, 2010, pp. 475–478.

[14] D. Chen and H. Zhao, "Data security and privacy protection issues in cloud computing," in *2012 International Conference on Computer Science and Electronics Engineering*, vol. 1. IEEE, 2012, pp. 647–651.

[15] Z. A. Almusaylim and N. Jhanjhi, "Comprehensive review: Privacy protection of user in location-aware services of mobile cloud computing," *Wireless Personal Communications*, vol. 111, no. 1, pp. 541–564, 2020.

[16] P. R. Kumar, P. H. Raj, and P. Jelciana, "Exploring data security issues and solutions in cloud computing," *Procedia Computer Science*, vol. 125, pp. 691–697, 2018.

[17] M. Abdelsalam, R. Krishnan, Y. Huang, and R. Sandhu, "Malware detection in cloud infrastructures using convolutional neural networks," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 162–169.

[18] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection," in *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*. IEEE, 2007, pp. 421–430.

[19] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *Journal in computer Virology*, vol. 7, no. 4, pp. 247–258, 2011.

[20] B. Athiwaratkun and J. W. Stokes, "Malware classification with lstm and gru language models and a character-level cnn," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2482–2486.

[21] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2015, pp. 11–20.

[22] S. Seok and H. Kim, "Visualized malware classification based-on convolutional neural network," *Journal of the Korea Institute of Information Security & Cryptology*, vol. 26, no. 1, pp. 197–208, 2016.

[23] N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé *et al.*, "Deep android malware detection," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, 2017, pp. 301–308.

[24] M. Abdelsalam, R. Krishnan, and R. Sandhu, "Online malware detection in cloud auto-scaling systems using shallow convolutional neural networks," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2019, pp. 381–397.

[25] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3422–3426.

[26] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 559–570, 2013.

[27] M. R. Watson, A. K. Marnerides, A. Mauthe, D. Hutchison *et al.*, "Malware detection in cloud computing infrastructures," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 192–205, 2015.

[28] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM computing surveys (CSUR)*, vol. 44, no. 2, pp. 1–42, 2008.

[29] R. Rehman, G. Hazarika, and G. Chetia, "Malware threats and mitigation strategies: a survey," *Journal of Theoretical and Applied Information Technology*, vol. 29, no. 2, pp. 69–73, 2011.

[30] S. Omar, A. Ngadi, and H. H. Jebur, "Machine learning techniques for anomaly detection: an overview," *International Journal of Computer Applications*, vol. 79, no. 2, 2013.

[31] I. Santos, C. Laorden, and P. G. Bringas, "Collective classification for unknown malware detection," in *Proceedings of the International Conference on Security and Cryptography*. IEEE, 2011, pp. 251–256.

[32] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 270–288.

[33] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: A review," *computers & security*, vol. 30, no. 6-7, pp. 353–375, 2011.

[34] T. Li, J. Tang, and J. Xu, "Performance modeling and predictive scheduling for distributed stream data processing," *IEEE Transactions on Big Data*, vol. 2, no. 4, pp. 353–364, 2016.

[35] T. Cao, X. Peng, C. Zhang, T. K. Al Tekreeti, J. Mao, X. Qin, and J. Huang, "A popularity-aware reconstruction technique in erasure-coded storage systems," *Journal of Parallel and Distributed Computing*, vol. 146, pp. 122–138, 2020.

[36] J. Li, M. N. Krohn, D. Mazieres, and D. E. Shasha, "Secure untrusted data repository (sundr)." in *Osdi*, vol. 4, 2004, pp. 9–9.

[37] S. Contiu, S. Vaucher, R. Pires, M. Pasin, P. Felber, and L. Réveillère, "Anonymous and confidential file sharing over untrusted clouds," in *2019 38th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2019, pp. 21–2110.

[38] K. Chatterjee, V. Padmini, and S. Khaparde, "Review of cyber attacks on power system operations," in *2017 IEEE Region 10 Symposium (TENSYMP)*. IEEE, 2017, pp. 1–6.

[39] G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra, "Madam: a multi-level anomaly detector for android malware," in *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*. Springer, 2012, pp. 240–253.

[40] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web*. Springer, 2007, pp. 325–341.

[41] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[42] R. Burke, "Knowledge-based recommender systems," *Encyclopedia of library and information systems*, vol. 69, no. Supplement 32, pp. 175–186, 2000.

[43] A. Felfernig and R. Burke, "Constraint-based recommender systems: technologies and research issues," in *Proceedings of the 10th international conference on Electronic commerce*, 2008, pp. 1–10.

[44] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.