Journal of Universal Computer Science, vol. 22, no. 4 (2016), 494-520
submitted: 1/10/15, accepted: 30/3/16, appeared: 1/4/16 © J.UCS

Web Data Amalgamation for Security Engineering: Digital Forensic Investigation of Open Source Cloud

Asif Imran
(Institute of Information Technology, University of Dhaka
Ramna, Dhaka 1000, Bangladesh
asifimran@du.ac.bd)

Shadi Aljawarneh
(Software Engineering Dept
Jordan University of Science and Technology
Irbid 22110, Jordan
saaljawarneh@just.edu.jo)

Kazi Sakib
(Institute of Information Technology, University of Dhaka
Ramna, Dhaka 1000, Bangladesh
sakib@iit.du.ac.bd)

Abstract: The largely distributed nature and growing demand for open source Cloud makes the infrastructure an ideal target for malicious attacks that grants unauthorized access to its data storage and posses a serious threat to Cloud software security. In case of any nefarious activity, the Cloud provenance information used by Digital Forensic experts to identify the issue is itself prone to tampering by the malicious entities and results in insecure software running in Cloud. This paper proposes a scheme that ensures Software Security and Security of Cloud provenance in a series of steps, the first of which involves binding the provenance journals with user-data from which those were derived. Next, mechanisms for merging provenance with unstructured web data for improved $Security\ Intelligence$ (SI) is identified. Detection of attack models for nefarious malware activities in six Software as a Service (SaaS) applications running in real-life Cloud is taken as the research case and the performance of the proposed algorithms for those are analyzed. The Success Rates (SR) for melding the web data to secure provenance for the six specific SaaS applications are found to be 85.0554%, 96.7032%, 98.3871%, 93.9732%, 80.5000% and 84.9257% respectively. Hence, this paper proposes a framework for effectively ameliorating the current scheme of Cloud based Software Security, thereby achieving wider acceptance of open source Cloud.

Key Words: Software Security; Cloud Security Intelligence; Cloud Provenance Detection; Provenance-Web Data Amalgamation; Digital Forensic Investigation; Distributed Applications; Security, Integrity and Protection.

Categories: D.2.11, D.4.6

1 Introduction

The demand for open source Cloud computing has increased manifold since its inception, mainly due to rapid provisioning, low cost and high availability attributes of this technology. On the contrary, rapid adoption of Cloud computing makes it a prime target of malicious attackers [Choo, 2014]. Nefarious attacks can cause critical Cloud functionalities to cease and Software Security of Cloud based applications to be compromised, resulting in unauthorized obtainment of customer data that are stored in Cloud servers. Therefore, the service becomes unreliable and it makes the wider acceptance of open source Cloud questionable [Pearson, 2009].

Due to scalability and dynamic resource provisioning of Cloud computing from a wide array of physical machines, traditional methods of system security is no longer a feasible approach for Cloud-based Software Security. Previous research has effectively quantified the security parameters needed for the Cloud paradigm [Rahulamathavan *et al.*, 2014]. In this regard, the use of provenance meta-data have been advocated for detecting disputed activities in Cloud environment [Moreau *et al.*, 2008], [Li *et al.*, 2014]. However, the conventional approach of provenance based search and seizure technique is not applicable for the Cloud since the provenance is itself prone to tampering. Additionally, only provenance is not suitable enough to provide system wide security since it lacks freshness of information [Jajodia *et al.*,]. In this regard, research needs to be done to make Cloud provenance tamper-proof to ensure its integrity for Digital Forensic Investigation (DFI). In addition, the scope of amalgamating unstructured web data with provenance to obtain analytics on latest security threats in Cloud needs to be explored. More specifically, the following research questions need to be addressed:

1. **How can provenance meta-data be detected, annotated and secured for DFI of Software running in open source Cloud?**

2. **How can fresh web data be merged with system provenance to obtain improved Software Security in Cloud against threat models of specific typed?**

The arrival of Cloud computing provides new mechanisms of obtaining and utilizing resources as discussed in [Schwiegelshohn *et al.*, 2010]. Securing provenance and amalgamating web data to it for security intelligence of Cloud will make those technologies more widely acceptable to be used for a wide array computing purposes by the public. Trenwith *et al* identified that the data stored and produced in Cloud passes through a large amount of physical machines, making it difficult to conduct forensic investigation using traditional mechanisms [Trenwith and Venter, 2014]. The importance of provenance has been addressed in this regard by the authors, however the provenance itself needs to be tamper-proof to ensure its integrity. Provenance based access control mechanism for the Cloud has been proposed by Danger *et al* in [Danger *et al.*, 2015]. The authors proposed a new provenance graph generation technique over traditional ones that answers queries by ignoring missing fragments. Amalgamation of web data to this technique will increase the possibility of recovering the missing fragments in Cloud provenance and yield improved access control [Martin Gilje *et al.*, 2014]. Aljawarneh [Aljawarneh, 2011] believes security should be an intrinsic part of the System Development Life Cycle (SDLC), and so he presented a web engineering security methodology, based on software engineering principles, to secure distributed e-systems. Also Buyya *et al* addressed the importance of strong security measures in the Cloud to achieve trust and confidence of the customers [Buyya *et al.*, 2009]. In this case secure provenance and web data amalgamation can be a persistent security solution for open source Cloud.

The proposed scheme ensures the security ofsoftware running in Cloud via provenance in a series of steps, the first of which involves binding the provenance journals and user data from which the provenance was derived. $Active-threading$ is the proposed mechanism that checks multiple provenance journals and binds those from the vm's to pm's. $Active-threading$ treats all provenance journals as individual objects and those are then bounded with multiple user data. Next the bounded provenance are encrypted using the private key of the $ProvenanceOwner$ (PO). Next, those are passed to an independent trusted body called the *Appraisor*. The framework is implemented in real-life infrastructure of commercial Cloud service provider.

The *Appraisor* is included to ensure that the PO does not have a single point of control. The *Appraisor* process is controlled by an organization that is trusted by both the Cloud provider and the Cloud customer. The *Appraisor* encrypts the captured provenance using its private key. The public keys of the PO and *Appraisor* are shared to aid in decryption of the provenance when those are needed to be analyzed during DFI.

Provenance have been generally referred here as *keystone* data, signifying the criticality of those in Cloud–based Software Security Intelligence (SI). On the other hand, web data stores are referred to as *cornerstone* as specified earlier since those only add value in specific situations. The importance of melding cornerstone web data with keystone provenance is important since those provide freshness and large volume of useful information, thus ensuring Software Security. Using provenance for Software Security for applications running in Cloud can be increasingly effective if the strength of web cornerstone data can be harnessed to increase the information volume. In addition to the above, this paper also proposes a methodology and provides algorithms to acquire cornerstone data on the web from specific sources using watchword matching scheme [Hwang and Li, 2010], [Imran *et al.*, 2013b].

Specific attributes like owner rating, user rating and freshness of information have been considered as verification criteria for the algorithms to increase the acceptance of those. Next, $Effective-Descriptive$ set theory have been proposed for identifying valid web data that treats sources as specific set of elements [Imran *et al.*, 2013b]. Each set of applications consists of a subset of processes from which the cornerstone data are obtained. Analysis of results show that the SR for profiling and amalgamating the web data to the secured provenance for the six $SaaS$ applications running on real life Cloud are found to be 85.0554%, 96.7032%, 98.3871%, 93.9732%, 80.5000% and 84.9257% respectively and 45% of the instances faced an average delay of 7-8 seconds due to the application of the proposed mechanism which is desirable. The delay between 9-9.9 seconds is faced by 40% of the instances for encapsulating the provenance meta-files to achieve Software Security for applications running in the Cloud.

2 Related Work

Cloud computing security has been initially investigated in a number of key perspectives namely Critical data encryption based security, Topological analysis based security and Tracer data analysis and visualization based security. Recent research efforts on the three mentioned perspectives are described in the following sub-sections.

2.1 Critical data encryption in Cloud environment

Cloud security has been modelled from the perspective of securing user data only, leaving many important areas unaddressed [Martini and Choo, 2014a]. Inspired by earlier security encryptions of traditional data, the authors enforced strict encryption mechanisms of user data in Cloud [Zhang *et al.*, 2012]. The basic approach was to consider the data from each user as individual objects which were encrypted using private/public keypairs of the corresponding users. Additionally, $Auditors$ were proposed that verified the security fingerprints in each data object. The rationale behind encrypting user data is that most critical information are stored in those and it is frequently targeted during denial of service attacks [Ficco and Rak, 2014].

While this rationale is sometimes true, it does not take into account the system level data in open source Cloud environments, which is equally important since system level provenance can contain information about malicious activities which cannot be determined only by user data. Moreover, securing only the user data may not provide fresh information on latest security vulnerabilities in open source Cloud since those do not have provisions for merging with web data.

2.2 Topological analysis of provenance information

System that provides a forensic model to identify threats in vms of Cloud based on provenance is proposed in [Trenwith and Venter, 2014]. It emphasizes the discrepancies in traditional seizure approach of provenance and identifies the importance of generating provenance graphs to abstract the unclear information. Similarly, provenance cognition to detect malicious worms have been proposed in [Imran *et al.*, 2013a]. The intuition of the authors is that blocks of systems provenance can be studied and pattern matched with worm behaviors that in turn can be used to detect the presence of worms. Thus by capturing, storing and profiling provenance information with worm monitoring signatures, the authors tried to ensure Cloud security.

The authors in [Kumar *et al.*, 2013] and [Dykstra and Sherman, 2012] addressed the fact that significant research needs to be conducted for provenance usage information to reach decisions regarding Cloud computing security. A public key framework to ensure the integrity, confidentiality and availability of provenance information via encryption of provenance data before passing those over the network is discussed in [Cao *et al.*, 2014]. Every user was assumed to have a public/private key pair and the public key of each receiver was known to their respective senders [Cao *et al.*, 2014]. Auditors were used to verify the trusted connection among different nodes to ensure integrity.

The major contributions by the authors included comparison of the performance of the proposed framework with other provenance detection schemes [Krishnan *et al.*, 2012]. At the same time, provenance overhead was calculated and compared with existing solutions for performance analysis. Analysis of provenance with security schemes detecting malicious activities from those were considered to a limited extent.

While signature matching the Cloud provenance is effective in detecting some malicious entities (for example, host-based and network-based worms like $Ramen$, $Millen$ and $Satyr$ worms as specified in [Naval *et al.*, 2014]), for broader security of the Cloud, the provenance itself needs to be made tamper-proof through a generalized mechanism. Moreover, the requirement to meet system-wide security from recent malicious activities cannot be met with the system proposed by the authors as it only takes historical meta-data into contention and ignores the importance of amalgamation of web data.

2.3 Analysis of tracer data in Cloud

For a distributed and virtualized computing environment like the Cloud, it would be ideal to form a Directed Acycylic Graph (DAG) of provenance to identify the nodes through which the malicious entities have passed and caused infection. This mechanism can be implemented for sharing publicly viewable provenance while protecting the integrity of critical information in those [Xue and Hong,]. However, such a mechanism will not be suitable to thwart zero day attacks as it mainly focuses on historical log data and does not take recent data into contention. In particular, a state of the art Cloud security risk management

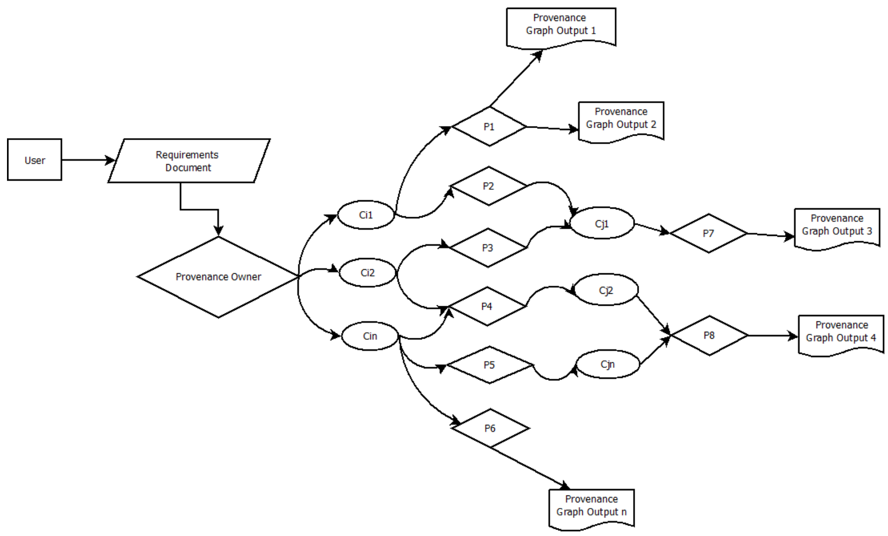

Figure 1: Encryption mechanism of the proposed framework

strategy was proposed in [Choo, 2014] that considered situational and environmental awareness to identify the motivations and goals of Cloud attackers. Data confidentiality, availability, integrity and authentication were considered to be primary target of attackers in Cloud. A $Bow - tie$ risk assessment mechanism was proposed to effectively encapsulate causations and reduce the probability of hazards from occurring. Securing the system level provenance of Cloud can be used to complement the proposal of the author and reduce chances of attacks.

A six-step process of collecting evidential data remotely for ensuring security of Cloud services is proposed in [Martini and Choo, 2014b]. The authors effectively addressed the importance of evidential data to prevent cyber attacks in the Cloud. The ever-increasing popularity of Cloud was considered to be an integral cause of increasing cyber attacks. A number of forensic preservations for the Cloud were identified and vMware vCloud was used as a case study to describe the existing Cloud artifacts. The proposed model can be enhanced to improve security of Cloud service through amalgamation of fresh web data that will provide knowledge about latest threats.

3 Active-Threaded framework for tamper-proofing provenance

The statements of the previous section emphasized the importance of not only capturing provenance, but also making the provenance tamper-proof to prevent unauthorized manipulations. The framework proposed here consists of encapsu-

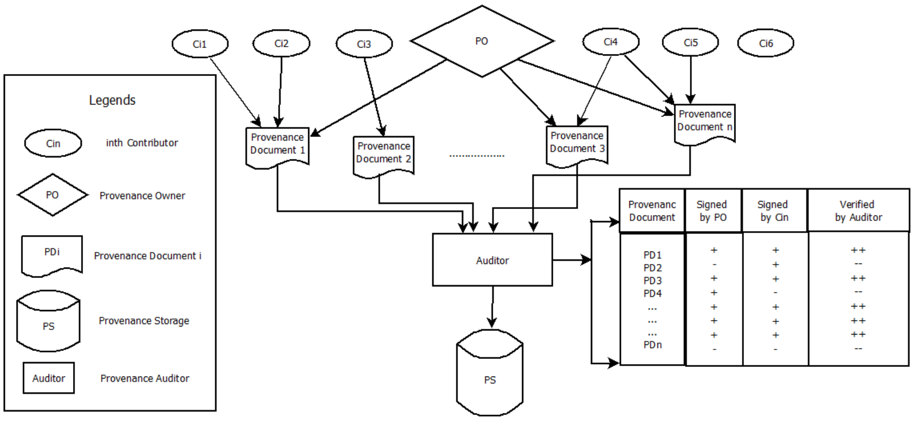

Figure 2: Architecture of the proposed provenance security framework

lation of the captured provenance meta-data with the original data using the proposed $ProvCapsule$ algorithm. Next $ProvOCal$ and $ObProv$ algorithms are used to ensure that the bounded provenance are encrypted with private keys from Provenance Owner (PO) and $Appraiser$. Both PO and $Appraiser$ processes are under the control of two different trusted bodies to ensure accountability and prevent the unwanted practice of single point evaluation. The PO and $Appraiser$ processes share the public keys with each other.

Figure 1 identifies the flow of sequence to ensure provenance security. The user can access a Cloud application through a browser or may request some data file in the Cloud storage. The data is stored in the C_i and provenance is collected from those using the ProvCapsule algorithm. Next the provenance file containing the time (t), source vm-id (svm), target vm-id (tvm) and date are stored in the provenance file which is then encrypted by PO using its private key. PO generates a private/public key pair and shares it with Appraiser. Appraiser receives key $K1$ from PO and provides its own public key $K2$ to PO.

The proposed framework consists of two trusted bodies that compose of many processes at the application level. The framework for secured provenance cognition is stated in Figure 2. Data requested in the Cloud are secured jointly by the Provenance Owner (PO) and $Appraiser$ processes stated in the legend of Figure 2. The critical processes identified in the legend are described in the following.

Security of the Cloud is often the most cited objection since customers are severely concerned about the security of the data stored on the Cloud servers [Yu *et al.*, 2012], [Chen and Lee, 2014]. The security issues which are used to protect the Cloud from attacks are similar to the ones implemented at large scale data centres, with the exception that both customers and not only Cloud administrators are responsible for the security. In addition to the two parties, a third party may be involved for the security of the Cloud as well which includes providers of value added services which are incorporated into the Cloud. For example, RightScale provides value added services, a few of which are automatically incorporated with Amazon Elastic Cloud Compute (EC2) and helps dynamic scale up or scale down of EC2 [Armbrust *et al.*, 2010].

3.1 ProvCapsule algorithm

Initially the provenance are stored in separate provenance files identified as F_{PROV} in Algorithm 1 that can be uniquely identified with an identity number. The files analyzed for provenance cognition are termed as F_{ORG} and those are stored in $Prov[y]$ array. F_{ORG} are next encapsulated with the related provenance information as identified in F_{PROV} [Chen and Lee, 2014]. Afterwards, each amalgamated file is assigned specific unique identifying information.

Algorithm 1 Provenance Capsule at Cloud Activity Layer

procedure ProvCapsule(a, b)
    $B_{ID} \leftarrow key$ and $F_{ORG} \leftarrow DataFile[x]$ and
    $F_{PROV} \leftarrow ProvFile[y]$ and
    $P_{SB} \leftarrow Loc[z]$
    while $B_{ID} \neq 0$ **do**
        Read inputs in F_{ORG}
        Record for every object a in $DataFile[x]$
        while $a = DataFile[x]$ **do**
            $B_{ID} = a.B_{ID}$ and
            F_{ORG}=F_{ORG+1}
            $DataFile[x+1]$=F_{ORG}
            continue
        end while
        Record for every object c in $ProvFile[y]$
        If $B_{ID}.F_{PROV}$=$c.F_{PROV}$
        $B_{ID}.F_{PROV} \leftarrow B_{ID}.F_{ORG}$
        $B_{ID} \leftarrow B_{ID} + 1$
    end while
    $ProvFile[y]$=($Loc[P_{SB}]$,$P_{SB}.Exec$,B_{ID})
    return $B_{ID}.F_{PROV}$ & $length.DataFile[x]$
end procedure

The proposed methodology aims for a unique identity number in the F_{PROV} file and the documents stored in the $DataFile$. In case of a match, the input I_a from the target document are read from the set of original inputs denoted as

B_{ID}. Next, provenance information such as date, time, operator and the type of operation in a specific activity are determined and enrolled into the $ProvFile$.

Afterwards the delivered information is stored in F_{PROV} and bounded to the F_{ORG} that contains the process data to uniquely identify the source of the detected provenance as stated in Algorithm 1. The trigger commands of the activity layer are responsible to move the encapsulated provenance files onto the Cloud storage as stated in Algorithm 2 of the next section.

3.2 ProvOCal algorithm

The algorithm described in this section identifies provenance data read from memory and it is also concerned with provenance data that are written on hard disks. At the same time, overhead incurred during information collection is identified here. In the stated algorithm, variable len_M and len_D are used to store the size of provenance data from memory and disks respectively. M_{inf} stores memory location of F_{org} together with length of the obtained provenance to address the requirement of measuring the space consumed by the software under observation. M_{sum} identifies the length of the disjoint memory blocks that is used to read provenance data to obtain the exact volume of memory consumption for the proposed methodology.

Algorithm 2 ProvOCal Algorithm for Provenance Detection in Memory and Disks

procedure PROVOCAL(a, b)
  $g \leftarrow 0$ and $h \leftarrow 0$
  $MInf = Loc[P_{SB} + 1]$
  while $g < len.MInf$ **do**
    $MSum = MInf$
    $S = min(g, len.MInf - 1)$
    $MSum = MSum + S$
  end while
  while $h < len.SBInfo$ **do**
    $SBSum = SBInf$
    $T = min(h, len.SBInf - 1)$
    $SBSum = SBSum + T$
  end while
  If $len.B_{ID} < len.MemInf$, then
  $f \leftarrow len.MemInf - len.B_{ID}$
  h = h+1;
  return Memory and Disk block consumption for provenance capsule
end procedure

As stated above, disk location can be calculated in a similar way to that of memory location since those occupy blocks of empty storage in the hard disk. After detecting the different blocks in the disk that stores provenance data files, those are summed up in M_{sum} and the total time taken to write down the provenance files on disk is calculated by T_{disk} since this information will be critical to estimate the time overhead of the proposed mechanism.

3.3 ReadProv algorithm

The *ReadProv* algorithm is designed to encapsulate the provenance meta-data together with the main message body. As a result, the provenance capsule produced by *ReadProv* consists of the Cloud provenance meta-data. Simultaneously users can enter the query that will generate the desired provenance information. The query is executed and the information is placed in the message array M_{sd} as shown in the algorithm. Variable g is used to store the size of the message array as this information is important to keep track of the storage consumption for calculation of the storage overhead caused for detection of provenance.

Algorithm 3 ObProv algorithm for capturing provenance at the application level

procedure ObProv(M_{sg}, M_{bd}, g)
    $M_{pr} \leftarrow Req(T, T_{vm}, S_{vm}, Date)$
    $M_{bd} \leftarrow msg_1, msg_2, .., msg_n$
    $Msg \leftarrow M_{pr} + M_{bd}$
    $A[x] \leftarrow 0$
    $g \leftarrow 0$
    while $g <= sizeof(Msg)$ **do**
        $A[g] = Msg$
        $g = g + 1$
        $ObProv \leftarrow ProvOwnerc, d$
    end while
    If $len.Msg < len.M_{pr}$, then
    $Msg \leftarrow len.Req(T, T_{vm}, S_{vm}, Date)$
    g = g+1;
    return Provenance information file Msg
end procedure

The array $A[x]$ consists of the messages that are stored as strings together with the timestamp T. The timestamps are used to compare the authenticity of the requests in the later algorithm. The request body $Req(T, T_{vm}, S_{vm}, Date)$ ensures that the critical provenance information as specified by the digital forensic experts. The contents are $T, T_{vm}, S_{vm}, Date$ to represent time, vm-id, storage allocated for the vm and date of access respectively. If the counter of g that consists of the size of the message that does not exceed, then the message Msg is placed the the array $A[g]$ in the location identified by the current value of g. At the same time, $ProvOwner$ is read from $ProbCapsule$ and placed in the owner variable of $ObProv$. Finally the Msg is assigned the Req body and send to the next algorithm called *Appraiser* that is described in the next sub-section.

3.4 Appraiser algorithm

The *Appraiser* of the process is responsible for verifying the request on the basis of the timestamps. The variable R_1 and R_2 are used to measure the size of the buffer array and at the same time implement the check constraint that it is under the limit of the initial consideration. Both variables are initialized to 0 to ensure that it is not reduced variables are incremented and counted, hence providing a variable that can be used as a counter function.

Algorithm 4 Appraiser: Setting permission for provenance access through timestamp authentication

procedure APPRAISER(R_1,R_2,R_3,R_4,$ObProv$,$size$)
  $R_1 \leftarrow 0$
  $R_2 \leftarrow 0$
  $Per_i \leftarrow 0$
  $size \leftarrow 1000$
  $Buf[size] \leftarrow ObProv$
  while $R_1 <= sizeof(Buf[size])$ **do**
    while $R_2 <= sizeof(Buf[size])$ **do**
      $R_1 = sizeof(Buf[size] + Msg)$
      $R_2 \leftarrow En.(Req(T, T_{vm}, S_{vm}, Date))$
      $R_1 = R_1 + 1$
    end while
  end while
  while $R_3 <= sizeof(Buffer[size])$ **do**
    while $R_4 <= sizeof(Buffer[size])$ **do**
      $R_3 \leftarrow timetoread.Buffer[size]$
      $R_4 \leftarrow timeofarrival.Buffer[size]$
      If $R_1 = R_3$ and
      If $R_2! = R_4$
      $Per_i = 1$
      else Per_i=0
    end while
  end while
end procedure

The two other variables R_3 and R_4 that are used to store the time-stamp when the first message is read and the arrival time of the second message respectively. The times of arrival of the new request is matched with the time when the first request if made. At the same time the vm-instance id that is making the request is also recorded.

The two time-stamps of the requests do not meet even in the case that the $vm{-}id$ of those matches, the Per_i will be set to 0 and the permission to access the provenance information will not be allowed because the mismatch of timestamp is caused by a different $vm - id$ philishing the identity of the authentic vm. The difference in the time-stamp will ensure that the vm requesting access does not have the authorization and it is philishing identity.

The algorithms discussed above function in line to detect system level provenance, at the same time, those are synchronized to ensure proper activity from provenance cognition, to secured provenance encryption. Hence, the 4 algorithms described here perform closely to achieve secured Cloud provenance to be used for DFI.

As stated earlier, the proposed mechanism detects provenance for enabling forensic experts to use it in digital forensic investigation. The next section identifies the amalgamation mechanism of secured provenance with recent web data to enhance the proposed framework's capability of identifying latest privacy and integrity threats in the Cloud.

4 Experimental Evaluation

This section identifies the effect of the proposed provenance securing and analysis schemes through experimentation on real life Cloud service environments. Performance benchmarks in terms of CPU overhead, delay, vm-counts and file sizes have been analyzed. Global Delay (TGD), Inter-message Delay (IMT) and Number of Retries for securing Cloud provenance have been analyzed upon comparison with existing benchmarks identified in [Slipetskyy, 2011]. Real life scenarios for the collection of provenance information namely file read, write and copy operations have been tested during experimentation and results obtained for those are shown here.

Securing the provenance using the proposed mechanism resulted in system delay and decreased throughput. Additionally frequencies and cumulative frequencies of the total number of retries for specific ranges of delay caused by securing the provenance have been analyzed using standard formulae identified in [Taylor *et al.*, 2010], [Zhu and Gong,].

After analysis of the provenance securing scheme, experimental results have been represented graphically for the algorithms proposed for web cornerstone data amalgamation with provenance to achieve increased SI. The cornerstone

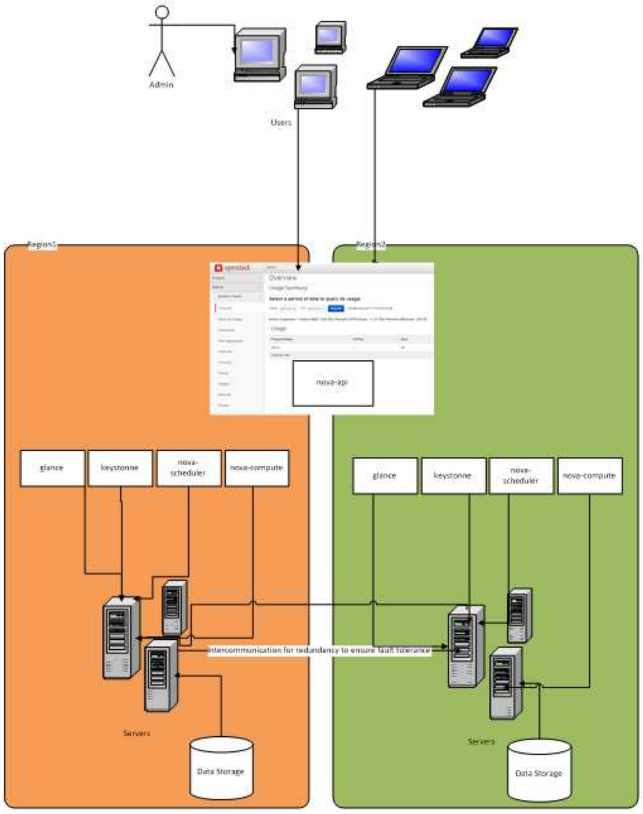

Figure 3: Representation of the real life environment for implementation of the proposed framework

data are crawled, converted to structured format and merged with provenance information as discussed in the previous sections. Next, data were collected for six real life applications running in pre-specified number of vm's identified by $vm - id$ in OpenStack Icehouse Cloud on Ubuntu 14.04 Long Term Service (LTS) servers. The accuracy of the amalgamation and the increased SI achieved through merging cornerstone and keystone provenance information have been presented. The rest of the section discusses in detail about the obtained results.

4.1 Sample Scenarios subject to Secured Provenance Detection

The experiments were conducted in real life Cloud environment running OpenStack Icehouse in Ubuntu 14.04 LTS servers. Total of 10 nodes were involved each with 32 GB of Random Access Memory (RAM), 2 Terabytes (TB) of hard drive and core i7 processors. Kernel Virtual Machine (KVM) was used for the

Table 1: Global Delay, Inter-Message Delay (IMT) and Retries of provenance capture

VM-Count	Size	TGD(s)	IMT(s)	Retries
140	512	1817	28.9	0
100	1024	1603	91.4	0
160	1536	1533	90.9	0
162	2048	1470	102.7	7
184	2560	1389	169.3	3
190	3072	1118	281.61	7

Cloud, that enabled obtainment of 20 vCPUs from each CPU and memory was shredded in different flavours in accordance to the requirements of the Cloud vm-instances. Next, the scenarios for which the provenance was collected for the system level was identified. In an Operating System each event is divided into a series of atomic steps. Each of the steps can be derived from the atomic actions and the kernel system calls. A specific pattern must be followed [Xu *et al.*, 2013], [Slipetskyy, 2011], which is called the signature of the operation. It is necessary for provenance detection since it characterizes and provides behavioral information of that process.

At the Data Layer, the main objective is to analyze the logs which are collected at the system level [Zeng *et al.*, 2012], [Yao *et al.*, 2012]. Journal files of Cloud Computing provide information that can be used to collect end-to-end system provenance. The provenance information collected in this way has been highly useful to achieve security and trust on behalf of the Cloud customers from the Cloud service providers.

Based on the analysis of the Cloud computing infrastructure and existing provenance models, provenance detection algorithms are placed forward with an aim of detecting file operations which have possibility of enabling data leakage. $ProvCapsule$ and $ProvOCal$ are such algorithms that are proposed in the paper which aim to drive real time provenance from Cloud activity layer without hampering the fault tolerance of the Cloud.

4.2 Analysis of results for the securing mechanism

The overhead incurred for encapsulating provenance in terms of global delay and message transmission delay needs to be measured for the algorithms proposed here. The results of the algorithms must be compared to established benchmarks to ensure that the overhead incurred does not exceed the pre-defined limit [Slipetskyy, 2011]. The following sections identify the performance of the algorithms in terms of overhead incurred for $Transmission\ Delay$ (TD) and

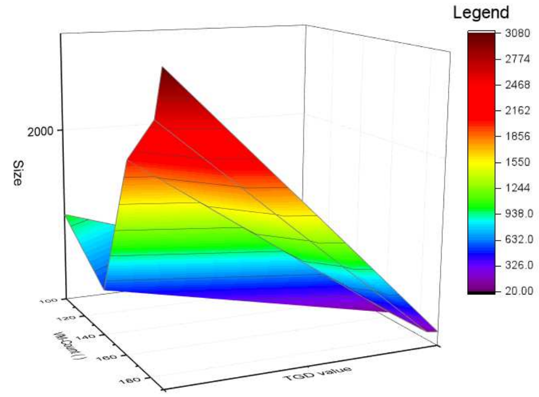

Figure 4: Graphical representation of the Total Global Delay (TGD)

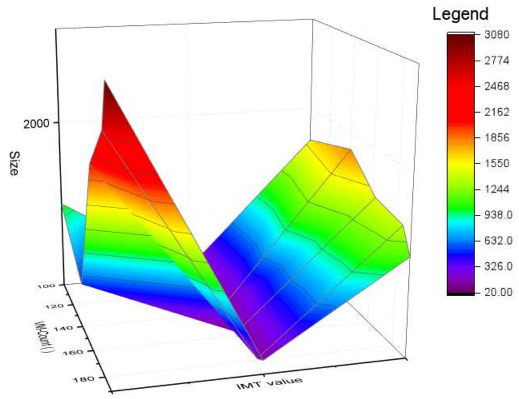

Figure 5: Graphical representation of the Inter Message Transmission Delay (IMT)

Global Delay (GD) for encapsulating provenance.

The results of overhead calculation for the proposed algorithms are tabulated in Table 1 and Table 3. The overheads of the proposed $ProvCapsule$ with $SecLaaS$ [Zawoad *et al.*, 2013] has been shown in Table 2. The number of vm-counts that transferred files of specific sizes ranging from 512-3072 MB are shown in column $vm\ count$. The Total Global Delay TGD is shown for the specific vm-instance counts in seconds. The TGD was found to be 1603 seconds and the minimum value is found to be 1118 seconds for 100 and 190 vm's respectively. In addition the $Inter\ Message\ Transmission\ Delays$ ($IMTD$) are also shown in seconds together with the number of retries of provenance encapsulation.

The algorithms were implemented in the Cloud controller that hosted 936

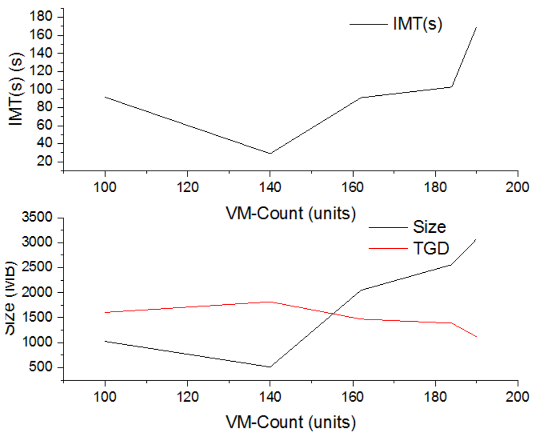

Figure 6: Comparison of TGD and File-size together with IMT for VM counts

Table 2: Overhead of the proposed model as compared to earlier mechanisms

Model	VM-C	Size	ORn(%)	OSq(%)	OAc(%)
ProvCapsule	200	512-1024	2.44	1.13	0.80
	200	2048-2560	0.66	0.48	0.24
SecLaaS[Zawoad *et al.*, 2013]	200	512-1024	4.12	2.06	NA
	200	2048-2560	1.88	1.37	NA

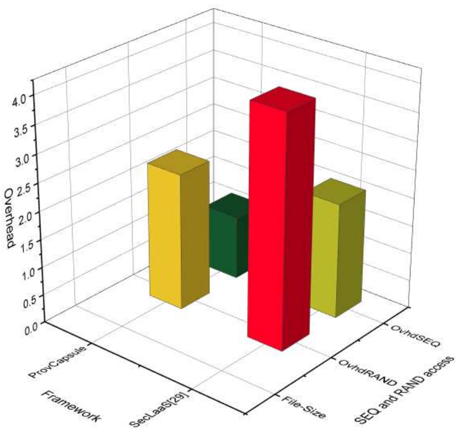

Figure 7: Overhead in terms of varying file size

vm-instances. Hence there were a finite population of vm-instances

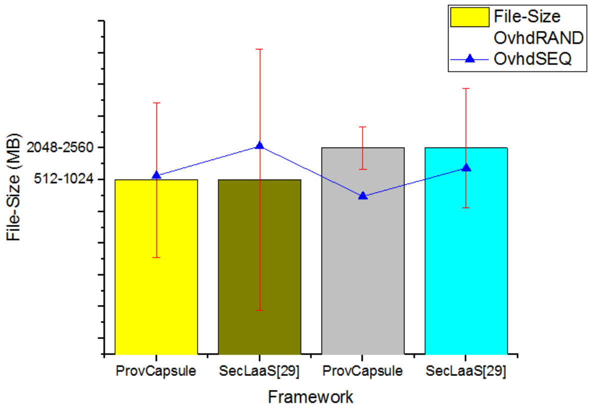

Figure 8: Error rate in terms of different files

Table 3: Mean Delay and Standard Deviation of Provenance Encapsulation

Size	M.Cont.	M.Time(s)	M.Delay(s)	STD.Dev
512	5.104	6.042		
100	5.924	8.584		
160	6.368	7.505	8.198	1.434
162	7.011	8.866		
184	8.026	9.990		

$$\overline{X_i} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{1}$$

The value can be obtained for a large number of observations as denoted by n. The variance of the delays in different ranges of Global Delays (TBD) and Inter-Message Time (IMT) is given by,

Table 4: Frequencies and Cumulative Frequencies of different Delay Ranges

Delay Range	Frequency	Cumulative Freq.(%)
1 - 2	14	1.50
3 - 4	22	2.40
5 - 6	104	11.1
7 - 8	422	45.1
9 - 9.9	374	40.0

F

$$S^2(n) = \frac{\sum_{i=1}^{n}(X_i - \overline{X(n)})^2}{n-1} \tag{2}$$

The closeness of the variance S_n^2 to mean μ can be determined as $Var(\overline{X(n)})$ is the ratio of total variance and number of occurrences for a given period t_i. The deviation calculated for each case were found through the formula in (3).

$$STD.dev = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \overline{X(n)})^2}{n(n-1)}} \tag{3}$$

Table 3 highlights the Mean Time ($M.Time$) required for provenance encapsulation for different sized files and different counts of vm-instances. The $M.Time$ is shown to be less than 10 seconds for 936 vm-instances. As identified in [Dastjerdi and Buyya, 2012], the benchmark of tolerable time for provenance encapsulation is 10 seconds for a large system. Taking that value as a benchmark, implementation of $ProvCapsule$ and $ProvOCal$ algorithms show that the time overhead is below 10 seconds for over 900 instances tested in real-life Cloud. The overhead is found to be 8.198 seconds on average which is acceptable. The standard deviation is found to be 1.434 which is desirable.

An important aspect of provenance management regarding data leakage is the causality of information. Information from one process can be causally related to information of another process, hence the atomic actions can be causally combined to evaluate against pre-specified benchmarks and reduce false-positives.

Finally Table 4 shows the delay range and frequency of delay for all the vm-instances. It is seen that 45% of the instances face an average delay of 7-8 seconds. The delay between 9-9.9 seconds is faced by 40% of the instances for encapsulating provenance metafile. Hence the average delay is below 10 seconds for both the algorithms.

The variance and the distribution of the vm-instance classes are shown in Figure 8a and Figure 8b. The variance is maximum for $M.Con$ whereas it is minimum for $M.Delay$ since the resources allocated in terms of memory and storage for each instance class was different. Hence the benchmark stated in [Dastjerdi and Buyya, 2012] is satisfied.

4.3 Analysis of Results for Merging Cornerstone and Keystone Provenance

The performance of the proposed algorithms were tested and analyzed in real life Cloud environments running OpenStack Icehouse Cloud. Six commercial real life applications provided as Software as a Service (SaaS) in Cloud vm-instances were analyzed at the servers of the Cloud service provider. The SaaS applications

Table 5: Performance evaluation of application level provenance cognition

App.	VM-id	Case	Alg Det.	SR (%)	MR (%)
IM	10	1084	922	85.0554	14.9446
AM	13	1001	968	96.7032	3.2968
HRM	07	834	772	98.3871	1.6129
ODM	16	896	842	93.9732	6.0268
SM	22	1200	966	80.5000	19.5000
CRM	20	942	800	84.9257	15.0743

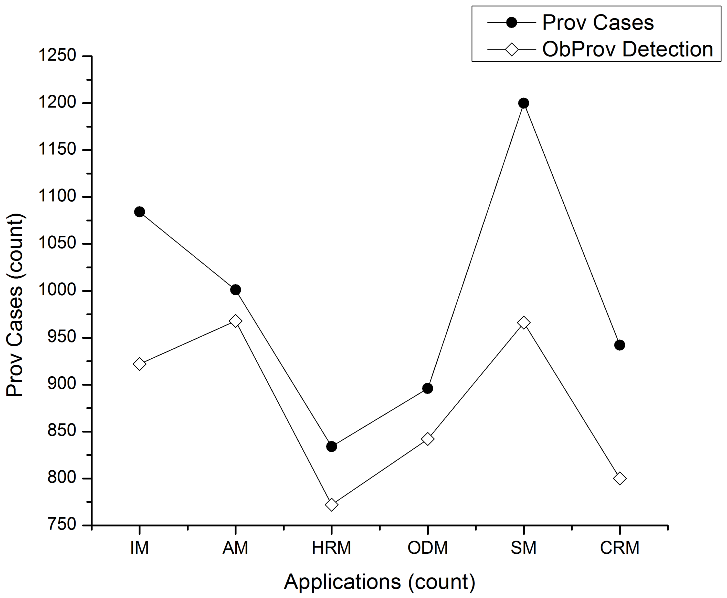

Figure 9: Sequence of transitions from one state to another in the proposed architecture

included Inventory Management Software (IM), Accounting Management Software (AM), Human Resource Management Software (HRM), Office Document Management Software (ODM), Sales Management Software (SM), Customer Relationship Management Software (CRM) as identified in Table 5.

There are a finite population of vm-instances so the number of accepted requests is finite as well. Hence we detect mean acceptance rate for any X_i as,

$$\overline{Acc_i} = \frac{\sum_{i=1}^{Count} Acc_i}{Count} \tag{4}$$

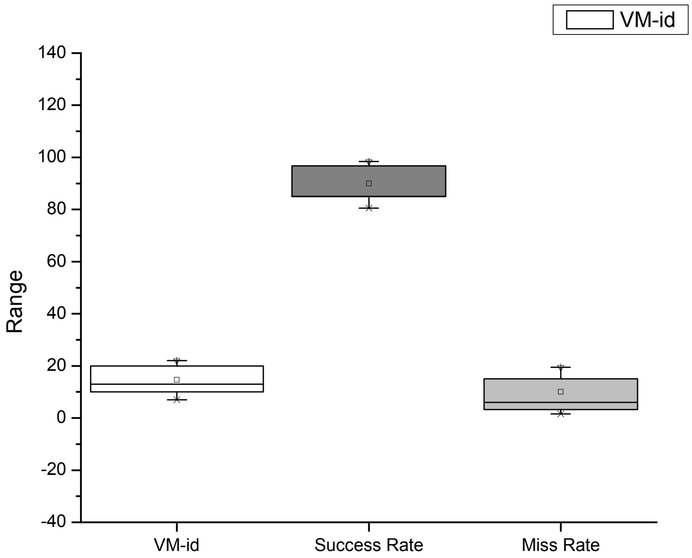

Figure 10: Box-plot of success rates

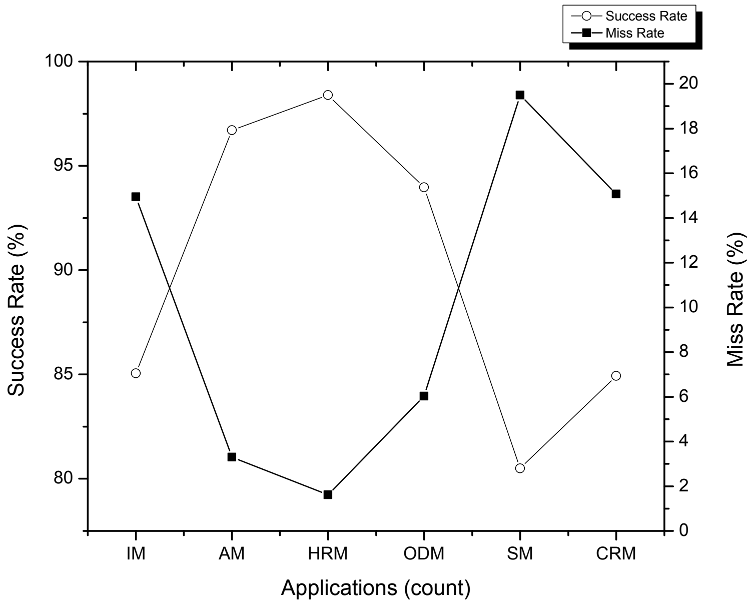

Figure 11: Comparison of malware capture and miss rates

The value can be obtained for a large number of observations n. The variance of the delays in different ranges of $Total\ Global\ Delays$ (TBD) and $Inter-Message\ Time$ (IMT) is given by,

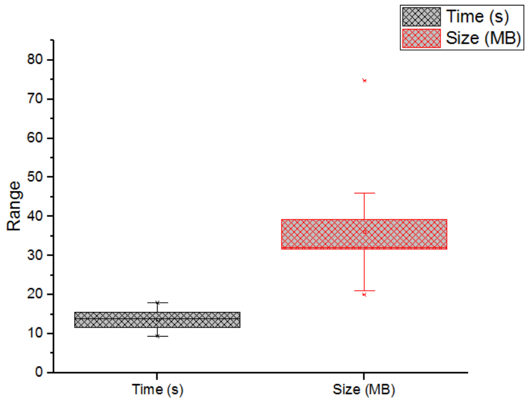

Figure 12: Graphical analysis of SR for the proposed algorithms

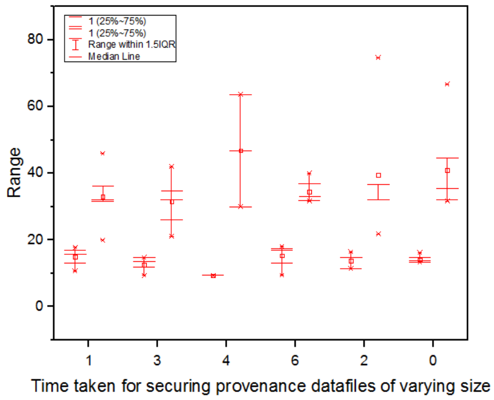

Figure 13: Query formulation to scenario based keystone and cornerstone data integration

$$S^2(n) = \frac{\sum_{i=1}^{Count}(Acc_i - \overline{Rej(Count)})^2}{Count - 1} \tag{5}$$

Individual vm-instances were allocated for each of the SaaS applications specified and provided to Cloud customers. Each customer has multiple users accessing the applications with their specific user names and passwords. For each customer, specific provenance information were collected that includes id-used to access, time and date of access, pages visited and changes made to files in terms of write operations. The provenance were detected from the SaaS level using *ObProv* algorithm and stored in a separate Cloud server running the *Appraiser*

Table 6: Mean Rejects and Cumulative frequency values for *ObProv*

Req. Size	Count	Acc.	Rej.	M.Delay	C.freq
10-100	63	13	50		
101-200	74	4	70		
201-300	69	20	49	64	21.43
301-400	82	18	64		
401-500	69	10	59		

algorithm for a period of 6 months to ensure that the data are in large volume, increased value and velocity, thereby satisfying the three v's of web cornerstone data.

The captured provenance using *ObProv* algorithm were stored in Cloud Provenance server that as *Appraiser* algorithm implemented on the machine using Linux platform and PostgreSQL database was used as it was already configured in the data center of the Cloud service provider where the experiments were conducted. The *Appraiser* algorithm provides authorized access to the provenance information based on timestamp authentication, thereby enabling the prevention of undesired access that would otherwise obtain the provenance files and tamper those [Zissis and Lekkas, 2012]. Hence *Appraiser* algorithm ensures tamper-proofness of provenance information. Table 5 highlights the results of provenance access acceptance and rejections by *Appraiser* algorithm.

5 Discussion of Results

Collection of provenance from the application layer are obtained for six real life applications provided as Software as a Service (SaaS) already identified in the previous section. The applications are used in real life by different number of users and provenance were collected from those in terms of operation executions, file accesses times and file user's identity. Next the system and application level provenance keystone data were secured by the *PO* and passed to the independent *Appraiser* process.

The *Appraiser* process further encrypted the provenance with its private key, as a result the causal chain of provenance security was used instead of onion framework as it would prevent the top level of encryption from being exposed because the second level of encryption is conducted by a different body than the first layer. At the same time the processes executed in the Cloud are dynamic and those occur in real-time. Hence the causal chain mechanism will have less overhead as found in the research because of its less layer of encryption. The onion framework provides increased security since it implements atleast 4 layers of encryption, however the overhead incurred in terms of CPU cycles and turnaround time is significantly higher for the onion framework as compared to the causal chain. Hence the causal chain framework is more applicable for the Cloud due to its low CPU overhead and adaptability to rapid response that is desirable for the Cloud system.

Web cornerstone data amalgamation results highlight the success at which unstructured cornerstone web data can be structured using watchword matching and fitted with keystone provenance. Six real life applications running as SaaS were monitored and provenance information were collected from those. Security cornerstone data were collected from malware and anti-virus websites as stated previously in the research.

Confidence on the cornerstone data were determined on the basis of user rating as specified in [Groth and Moreau, 2009]. Next the unstructured data were converted to structured format on the basis of watchword matching and stored in database using multi-dimensional *Spider Schemas*, giving rise to a fusion-cube. Section 4 identified the validity values from 1 to 4 assigned to the data obtained from specific web sources based on freshness, applicability in Cloud security intelligence, user ratings and owner ratings. The Success Rates (SR) for structuring and melding the cornerstone data to security keystone provenance of the six specific SaaS applications are found to be 85.0554%, 96.7032%, 98.3871%, 93.9732%, 80.5000% and 84.9257% respectively.

6 Conclusion and Future Work

This paper aims to identify a solution to the bottleneck of tamper-proofing provenance for open source Cloud and amalgamating those with web cornerstone data for improved SI for Software Security. The target was to provide a generalized framework for making the provenance itself tamper-proof for achieving effective Software Security for Cloud based applications and a case-specific mechanism of combining web cornerstone data to provide fresh information about open source Cloud's integrity and reliability.

The goal of this paper is to identify a solution to the issue of provenance detection and Software Security in a widely distributed environment such as the Cloud. Generalized framework for provenance cognition and mapping those to specific objects are provided in this paper to ensure Software Security. The proposed algorithms based on $Active - threading$ are capable of securing system-level provenance for vm-instances of Cloud, an improvement over traditional mechanisms that are concerned with provenance at the physical level only. The second goal is to propose novel algorithms for parsing cornerstone and ensuring the accuracy of those, thereby obtaining security of software running in cloud. Effective algorithms to conduct keyword based structuring of the cornerstone data has been proposed here. Secondly, algorithm for determination of the reliability and integrity of the cornerstone data is proposed based on owner ratings, user ratings and freshness of information to achieve Software Security. Finally, results of merging cornerstone data with keystone provenance are analyzed and performance compared for those.

The experimental environment for securing Cloud computing provenance in this paper have been implemented in real life Cloud service and the test-bed for the amalgamation of provenance with web data is described. Next, Spider Schema was proposed for the amalgamation of Cloud provenance keystone with web cornerstone data. The result of the proposed model yielded a fusion cube that consisted of multi-dimensional schemas. The fusion cube comprised of cornerstone data and keystone information for effective analysis of security threats to the Cloud. Graphical result analysis showed that the mean delay incurred for the proposed provenance securing schema is 8.1% which is desirable. For provenance amalgamation and fusion cube formation, the Mean Success Rate (MSR) was found to be 89.33%. At the same time the MR was 9.67% that is below the benchmark of 10.00%, yielding suitable results. The reason for the desirable SR is due to the web page rating mechanism on the basis of data freshness, user ratings and owner ratings as proposed in this paper. The SR for the six real life SaaS applications were determined to be 85.0554%, 96.7032%, 98.3871%, 93.9732%, 80.5000% and 84.9257%, yielding desirable results. Also 45% of the instances had a delay of 7-8 seconds caused due to implementing the algorithms that is acceptable based on specified benchmarks.

The proposed mechanism identifies a generalized framework of tamper proofing as an important mechanism for securing provenance and a case-specific framework for amalgamation of web data for better DFI in open source Cloud. The amalgamation of the cornerstone has been case-based and tested for specific real life Cloud applications running as $SaaS$. Performance of the proposed frameworks in other $SaaS$ applications not tested here can be an area of future research interest. In addition, amalgamation of data from sources other than the web (i.e. newspaper) can be tested as part of future research for checking the performance of those in Cloud forensic investigations.

Acknowledgment

This research has been supported by UGC, Bangladesh under the Dhaka University Teacher Grant No-Reg/Admn-3/2015/48743. The experiments have been implemented and tested in the real-life commercial cloud infrastructure of Panacea Systems LTD. The researchers would like to convey their gratitude to Panacea Systems LTD for providing the opportunity of implementing the framework.

References

[Aljawarneh, 2011] Shadi Aljawarneh. A web engineering security methodology for e-learning systems. *Network Security*, 2011(3):12–15, 2011.

[Armbrust *et al.*, 2010] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[Buyya *et al.*, 2009] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[Cao *et al.*, 2014] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *Parallel and Distributed Systems, IEEE Transactions on*, 25(1):222–233, 2014.

[Chen and Lee, 2014] Henry CH Chen and Patrick PC Lee. Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation. *Parallel and Distributed Systems, IEEE Transactions on*, 25(2):407–416, 2014.

[Choo, 2014] Kim-Kwang Raymond Choo. A cloud security risk-management strategy. *Cloud Computing, IEEE*, 1(2):52–56, 2014.

[Danger *et al.*, 2015] Roxana Danger, Vasa Curcin, Paolo Missier, and Jeremy Bryans. Access control and view generation for provenance graphs. *Future Generation Computer Systems*, 2015.

[Dastjerdi and Buyya, 2012] Amir Vahid Dastjerdi and Rajkumar Buyya. An autonomous reliability-aware negotiation strategy for cloud computing environments. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 284–291. IEEE, 2012.

[Dykstra and Sherman, 2012] Josiah Dykstra and Alan T Sherman. Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques. *Digital Investigation, Elsevier*, 9:90–98, 2012.

[Ficco and Rak, 2014] Massimo Ficco and Massimiliano Rak. Stealthy denial of service strategy in cloud computing. *Cloud Computing, IEEE Transactions on*, 3(1):80–94, 2014.

[Groth and Moreau, 2009] Paul Groth and Luc Moreau. Recording process documentation for provenance. *Parallel and Distributed Systems, IEEE Transactions on*, 20(9):1246–1259, 2009.

[Hwang and Li, 2010] Kai Hwang and Deyi Li. Trusted cloud computing with secure resources and data coloring. *Internet Computing, IEEE*, 14(5):14–22, 2010.

[Imran *et al.*, 2013a] Asif Imran, Alim Ul Gias, Rayhanur Rahman, and Kazi Sakib. Provintsec: a provenance cognition blueprint ensuring integrity and security for real life open source cloud. *International Journal of Information Privacy, Security and Integrity*, 1(4):360–380, 2013.

[Imran *et al.*, 2013b] Asif Imran, Alim Ul Gias, Rayhanur Rahman, Amit Seal, Tajkia Rahman, Farhan Ishraque, and Kazi Sakib. Cloud-niagara: A high availability and low overhead fault tolerance middleware for the cloud. In *Computer and Information Technology, 2013 International Conference on*, pages 164–170. IEEE, 2013.

[Jajodia *et al.*,] Sushil Jajodia, Krishna Kant, Pierangela Samarati, Anoop Singhal, Vipin Swarup, and Cliff Wang. *Secure cloud computing*.

[Krishnan *et al.*, 2012] Srinivas Krishnan, Kevin Z Snow, and Fabian Monrose. Trail of bytes: New techniques for supporting data provenance and limiting privacy breaches. *Information Forensics and Security, IEEE Transactions on*, 7(6):1876–1889, 2012.

[Kumar *et al.*, 2013] Naveen Kumar, Anish Mathuria, Manik Lal Das, and Kanta Matsuura. Improving security and efficiency of time-bound access to outsourced data. In *Proceedings of the 6th ACM India Computing Convention*, page 9. ACM, 2013.

[Li *et al.*, 2014] Jin Li, Xiaofeng Chen, Qiong Huang, and Duncan S Wong. Digital provenance: Enabling secure data forensics in cloud computing. *Future Generation Computer Systems*, 37:259–266, 2014.

[Martin Gilje *et al.*, 2014] Jaatun Martin Gilje, Pearson Siani, Gittler Frederic, and Leenes Ronald. Towards strong accountability for cloud service providers. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 1001–1006. IEEE, 2014.

[Martini and Choo, 2014a] Ben Martini and Kim-Kwang Raymond Choo. Cloud forensic technical challenges and solutions: A snapshot. *IEEE Cloud Computing*, (4):20–25, 2014.

[Martini and Choo, 2014b] Ben Martini and Kim-Kwang Raymond Choo. Remote programmatic vcloud forensics. In *Proceedings of 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2014)*, 2014.

[Moreau *et al.*, 2008] Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez-Salceda, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, et al. The provenance of electronic data. *Communications of the ACM*, 51(4):52–58, 2008.

[Naval *et al.*, 2014] Smita Naval, Vijay Laxmi, Neha Gupta, Manoj Singh Gaur, and Muttukrishnan Rajarajan. Exploring worm behaviors using dtw. In *Proceedings of the 7th International Conference on Security of Information and Networks*, page 379. ACM, 2014.

[Pearson, 2009] Siani Pearson. Taking account of privacy when designing cloud computing services. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 44–52. IEEE Computer Society, 2009.

[Rahulamathavan *et al.*, 2014] Y Rahulamathavan, PS Pawar, Pete Burnap, M Rajarajan, Omer F Rana, and G Spanoudakis. Analysing security requirements in cloud-based service level agreements. In *Proceedings of the 7th International Conference on Security of Information and Networks*, page 73. ACM, 2014.

[Schwiegelshohn *et al.*, 2010] Uwe Schwiegelshohn, Rosa M Badia, Marian Bubak, Marco Danelutto, Schahram Dustdar, Fabrizio Gagliardi, Alfred Geiger, Ladislav Hluchy, Dieter Kranzlmüller, Erwin Laure, et al. Perspectives on grid computing. *Future Generation Computer Systems*, 26(8):1104–1115, 2010.

[Slipetskyy, 2011] Rostyslav Slipetskyy. *Security issues in OpenStack*. PhD thesis, Masters thesis, Norwegian University of Science and Technology, 2011.

[Taylor *et al.*, 2010] Mark Taylor, John Haggerty, David Gresty, and Robert Hegarty. Digital evidence in cloud computing systems. *Computer Law & Security Review*, 26(3):304–308, 2010.

[Trenwith and Venter, 2014] Philip M Trenwith and Hein S Venter. A digital forensic model for providing better data provenance in the cloud. In *Information Security for South Africa (ISSA), 2014*, pages 1–6. IEEE, 2014.

[Xu *et al.*, 2013] Guoyan Xu, Zhijian Wang, Li Yang, and Xiaoyi Sun. Research of data provenance semantic annotation for dependency analysis. In *Advanced Cloud and Big Data (CBD), 2013 International Conference on*, pages 197–204. IEEE, 2013.

[Xue and Hong,] Kaiping Xue and Peilin Hong. A dynamic secure group sharing framework in public cloud computing. *Cloud Computing, IEEE Transactions on*, 3(1).

[Yao *et al.*, 2012] Junjie Yao, Bin Cui, Zijun Xue, and Qingyun Liu. Provenance-based indexing support in micro-blog platforms. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 558–569. IEEE, 2012.

[Yu *et al.*, 2012] Zhiwei Yu, Chaokun Wang, Clark Thomborson, Jianmin Wang, Shiguo Lian, and Athanasios V Vasilakos. A novel watermarking method for software protection in the cloud. *Software: Practice and Experience*, 42(4):409–430, 2012.

[Zawoad *et al.*, 2013] Shams Zawoad, Amit Kumar Dutta, and Ragib Hasan. Seclaas: secure logging-as-a-service for cloud forensics. In *Proceedings of the 8th ACM*

SIGSAC symposium on Information, computer and communications security, pages 219–230. ACM, 2013.

[Zeng *et al.*, 2012] Kai Zeng, Kannan Govindan, Prasant Mohapatra, et al. Chaining for securing data provenance in distributed information networks. In *Military Communications Conference, 2012. MILCOM 2012. IEEE*, pages 1–6. IEEE, 2012.

[Zhang *et al.*, 2012] Olive Qing Zhang, Ryan KL Ko, Markus Kirchberg, Chun Hui Suen, Peter Jagadpramana, and Bu Sung Lee. How to track your data: Rule-based data provenance tracing algorithms. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 1429–1437. IEEE, 2012.

[Zhu and Gong,] Shasha Zhu and Guang Gong. Fuzzy authorization for cloud storage. *Cloud Computing, IEEE Transactions on*, 2(4).

[Zissis and Lekkas, 2012] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Generation Computer Systems*, 28(3):583–592, 2012.