ProvIntSec: A Provenance Cognition Blueprint Ensuring Integrity and Security for Real Life Open Source Cloud

Asif Imran

Institute of Information Technology,
University of Dhaka
Email: asif.imran.anik@gmail.com

Alim Ul Gias

Institute of Information Technology,
University of Dhaka
Email: alimulgias@gmail.com

Rayhanur Rahman

Institute of Information Technology,
University of Dhaka
Email: rayhanur.rahman@outlook.com

Kazi Sakib

Institute of Information Technology,
University of Dhaka
Email: sakib@univdhaka.edu

Abstract: The distributed nature and growing demand for open source cloud makes the system an ideal target for malicious attacks and unauthorized file transfers. Requirements of provenance cognition scheme can come forward to solve the problem. However, such mechanisms of provenance detection has been considered to a limited extent for open source cloud computing. ProvIntSec is a novel mechanism that ensures effective collection of provenance information from a large pool of virtual machine (VM) instances on open source cloud platform. ProvIntSec captures critical system journals from VM instances and pattern matches those with predefined signatures to detect the presence of malicious activities. In addition, ProvIntSec identifies the Linux process trees to determine unauthorized file movements across different nodes. The experiments were executed in OpenStack Essex cloud environment running on real life system, and standard metrics were used to calculate the results. The obtained results show average precision values of **92.81**% and **81.24**% for malware detection and unauthorized file transfers respectively. At the same time, cumulative performance gains of **0.3991** and **8.77** are obtained. Upon comparison of the obtained results with benchmarks, ProvIntSec shows desirable gain in performance.

Keywords: provenance; cloud computing security; cloud computing integrity;performance evaluation.

1 Introduction

Open source cloud computing is becoming increasingly popular to cloud customers mainly due to the provision of ubiquitous, on-demand, inexpensive and dynamically scaling services. However, the rapid adoption and distributed nature of open source cloud makes it a prime target of malicious attacks [1], [2]. Such nefarious activities can affect critical processes and cause unauthorized transfer of important customer data. This causes degradation in the reputation of open source cloud service providers.

Due to the distributed nature of cloud computing, traditional methods of collection of metadata is no longer a practical approach [3], [5]. Research needs to be done whether it is possible for cloud administrators to identify malicious activities from provenance data. Architectural idea for a trusted provenance system, and the requirements for effective analysis of journal files of cloud computing systems should be identified for effective provenance detection. More specifically, answers to the following questions need to defined.

1. How can provenance information be collected across a large pool of virtual and physical machines in real life open source cloud environments?
2. How can those collected provenance information be used to detect the presence of malicious activities in the cloud system?

Cloud management mechanisms aid in proper maintenance of cloud platform with significant effectiveness [4]. Integration of provenance based malware and unauthorized file transfer detectors will increase the wide acceptability of such tools. In the literature, research for provenance detection to ensure the security and integrity of the cloud is important. Attacks by malicious agents have been considered to be a major issue in [2]. Ambrust et al identified malware activities as a leading obstacle to the widespread acceptability of cloud services [6]. Negative behaviors of attackers posing as customers can cause black listing of EC2 IP-addresses through spam prevention which ultimately limits the application services of the cloud provider [6]. Trusted email has been suggested as a solution to the above problem, which leads to microcosm of the issue. Slipetskyy [9] stated that malevolent activities like worm attacks cause tremendous damage to cloud customers by terminating critical processes and transferring important data to non-permitted bodies. In addition the top cloud computing threats were analyzed and the necessities of provenance detection to solve those have been stated. However, detection of nefarious activities from provenance information have been considered to a limited extent.

Provenance triggers processes to collect journals from system critical files of the cloud. Provenance captures and documents the inputs, entities, and triggers of the processes which provide a historical record of the information. The produced evidence can be used for data forensic investigations such as malware detection and analysis, detection of data compromise, and aiding in data audits. The data in the buffer are then transmitted over the network as data packets to the cloud controller. ProvIntSec analyzes the collected journals and matches those with predefined pathwords which are used to train ProvIntSec [7]. The proposed provenance cognition framework looks to pattern match the leftovers in the journals to determine the presence of malware activities. Journal files record activities that take place in a server which are stored in chronological order. The standard benchmarks used to analyze the results of the experiments are based on [7] and [8]. The base values have been considered for evaluation and comparisons of the results in this paper.

In addition to the above, ProvIntSec is capable of detecting file movements across multiple machines in the distributed environment. Malicious worms are capable of transferring critical system files from cloud master machine to VMs without permission. ProvIntSec detects unauthorized file transfers by maintaining a lookup table of VM-instance identity (id) numbers that are allowed to receive the file. Analysis of the recorded provenance can be used to identify a process tree. The process tree indicates the processes through which data are transferred across VM-instances using socket. Parsing the process trees of ProvIntSec enables the determination of all the virtual and physical machine processes to which the malware has spread.

The research methodology focuses on capturing provenance and analyzing those to identify the existence of malware or unauthorized file transfers. In this experiment, OpenStack cloud running in the real life environment of cloud service provider was used as the cloud platform to execute the tests. The test for malicious activities included 8 Linux worms namely Adore, Kippo, Lupper, Mighty, Millen, Satyr, Slapper and SSHBruteForce [9].

In terms of provenance detection for file movements, test files of 8 different sizes were transferred 5600 times to non-permitted VM-instances. The instance ids of the VMs that received the files were captured from the provenance journals. Next the provenance journals were inspected to generate process trees which spanned across multiple machines. The ids of the VMs that received the files were matched with the entries of the lookup table to detect unauthorized file movements.

On the basis of standard benchmarks discussed above, the precision are determined for the experiments. Next, the performance gain of ProvIntSec is detected with respect to the baseline values. Experimental results show that ProvIntSec gives a precision value of **92.81**% in detecting the presence of malware. The precision value in detection of unauthorized file movements sum up to **81.24**%. The frequency and impact of the 8 worms used in the experiments when ProvIntSec is installed are provided in a risk matrix. It can be said from the experimental results that the attained performance is applicable to worms which were not even used in the experiment.

The rest of the paper proceeds as follows. Section 2 identifies the related research in the field of cloud computing security and integrity. Section 3 proposes provenance cognition scheme. Section 4 describes the experiments conducted in real life cloud environment and the metrics used to obtain the performance results. Section 5 discusses the results and Section 6 draws the conclusion and identifies the scope of future research.

2 Related Work

Ensuring cloud computing integrity and security is a significant challenge to the widespread acceptance of cloud among the people. The models and frameworks which aim to increase cloud computing trust, accountability and acceptability are identified in this section. At the same time, the schemes are analyzed to identify scope of improvement of those. Key challenges for provenance detection in the cloud are identified in [10]. At the same time data coupling, multi-object causal ordering, data independent persistence and efficient querying of provenance data should be identified as the usefulness of provenance detection [10].

A cloud computing management tool by the name of Scientific Utility for Cloud Harnessing (SUCH) has been proposed in [4]. The mechanism provides automated and user-friendly deployment of applications to Microsoft Windows Azure cloud platform. Integration of a cloud provenance capture and analysis mechanism enabling malware presence detection may add more value to the framework stated in [4].

Provenance digital signatures and optimal security hardening of information network is proposed to facilitate information trustworthiness in distributed environments [11], [12]. In the study, a trustworthiness graph is formulated using abstraction with respect to am piece of information where the nodes represent a principal and arcs represent transmission of messages. A new cryptographic primitive called provenance digital signature has been proposed that go beyond the concept of digital signature preserving historical data [12]. Trustworthiness of captured provenance will ensure integrity of the provenance data [11], [13], [14] . Similar provenance based digital signatures can be studied for detection of malware presence in distributed environments.

Malicious packet dropping attack has been considered to be a major security threat in [2]. The authors proposed a data provenance based mechanism to detect the attack and the attacker node. Watermarking based secure provenance transmission mechanism have been used to detect packet loss, attack identification and attacker node detection. The problem space of secure provenance management and and solution to those have been addressed in [15]. The provenance model can be extended to detect the presence of unauthorized file transfers in the cloud. In addition the provenance management issues are required to be studied to analyze and derive inferences from provenance data.

The remote storage of data and remote computation are considered as critical risks for both the cloud service providers and customers [16]. The authors proposed that accountability must be ensured for both the customers and service providers of cloud. In case of any problem, both parties must be made aware about the liabilities of the concern. Secure auditing, recording and evidence of information stored on the cloud were proposed as probable solutions to this research issue. The incorporation of journal based provenance detection for the cloud may strengthen the proposed solution by increasing data integrity and security.

Ko et al identified accountability of cloud services as a complex challenge to achieve by developers due to the distributed nature of cloud computing [17]. Through the description of real life scenarios which included storing images on cloud servers, real time tracing of operations executed on the cloud were identified. It was described how malicious attackers tried to transfer files from the cloud to servers outside using email services. Regulations which hinder the blocking of such transfers were also identified. In addition to detecting data transfers through emails, cloud provenance information can also be used to detect unauthorized file transfers using copy and move commands in the cloud environment. The use of provenance to detect malware attacks were considered to a limited extent in [17].

Technical and procedural approaches to ensure privacy is a key goal for software engineers when building a cloud for the production environment [18]. Ensuring lower levels of privacy through the use of privacy management tools for system level operations has been highlighted in [18]. Use of pseudonymisation tools to hide the name of real users is proposed. The technology includes features like anonymous login, pseudonymous identity number and email addresses. Addition of journal based file and system level provenance can make the proposed tool stronger through the detection of malware activities and non-permitted file transfers in the cloud.

Lu et al [19] and Davidson et al [21] identified privacy aware provenance of scientific workflows in terms of data module and privacy policy. These research identified questions regarding provision of unlimited or fixed number of allowable provenance queries. The provenance information is to be made available to the user assuring the security and privacy of the information. They comprised of providing both composite and simple workflows of provenance data as acyclic graphs and then obtained inference from those. Both of these research are concerned with provenance detection for services executed in the cloud, not with the cloud system itself.

Recent technologies for provenance detection in scientific workflow systems have been identified in [22]. Prospective and retrospective provenance capture, storage and retrieval of provenance data from database, modeling and representation of information are the primary purpose of the research in contention. The data are stored in systems using specialized Semantic Web Languages and XML dialects that are stored as files and the tuples stored in relational databases. The research is only concerned with scientific workflow systems and does not consider provenance detection, storage and management of system-level files.

‘Accountability as a Service (AaaS)’ has been identified for cloud computing environments to increase wide acceptability of cloud computing services [23]. The research presented a model in which cloud service providers are held responsible for accountability issues related to the services deployed in the cloud. This is achieved by making the service provider accountable for the faults and breaches in the Service Level Agreement (SLA). The methodology allows fault detection of predefined requirements through binding web addresses. The research in [24] has modified [23] through incorporation of cloud users beside providers and making cloud users equally accountable for any malicious activities. The fault detection methodology of [23] has been improved in [24] by ensuring that the root of the fault is not always concluded to be the guilty, each conclusion must be backed up by proofs which are non-disputable. However both these research focused very little on the need of maintaining the captured provenance and devising an easy to understand methodology for provenance presentation. Also the benefits of ensuring accountability through provenance detection were highlighted to a little extent.

Zhou et al stated the importance of Role Based Access Control (RBAC) to ensure security and integrity of data shared in the cloud [25]. They proposed a Role Based Encryption (RBE) technique that uses cryptography to encrypt the data and enforces the RBAC methodology. Hence the users who have the permissions for the assigned role can decrypt the shared information. The methodology includes managers posing rights to accept and revoke membership of roles without obtaining permissions from other participants. Provenance detection of the activities of managers may enable to identify nefarious acts from the ends of the managers and prevent internal sabotage.

A framework to ensure trust and security of cloud environments with respect to large volumes of virtualization and distributed storage management is proposed in [26]. In the framework both technical and policy level approaches have been taken into account due to the importance of those in determination of the Service Level Agreement (SLA) . The methodology involves capturing provenance information in three different layers; namely system, data and workflow. The research does not address the opportunities of detecting malware activities and unauthorized file movements from the cloud provenance captured at the system and data layers respectively.

Wei et al discussed rule based provenance detection and tracing of data provenance to ensure actual detection of identity [27]. In the methodology, rule correlation engine was used which implemented provenance algorithms on various existing cloud software, and in terms of performance data leakage were analyzed. Data leakage identified the number of times the engine failed to detect data movements. The importance of system-centric provenance and visual representation of provenance were highlighted to a little extent.

Provenance detection for distributed computing has been analyzed in [28]. The provenance detection scheme identified provenance data using Axis 2C, a service provided by Apache Axis and Mule that logs the actions to reproduce the results of the operations. Also the authors considered the standards that can be followed to detect provenance data. However this research does not state how the proposed mechanism can be extended into the cloud environment.

As stated above, these research address the importance of journal based provenance detection framework to increase trust in open source cloud computing services [17], [18], [20]. Although provenance detection have been stated for stand-alone systems, detection of provenance between virtual and physical machines have been covered to a limited extent [27], [26]. Research is needed in the field of analyzing performance of provenance detection algorithms [28]. Although the provenance capturing from system and data files for the detection of malware and unauthorized file transfers are acknowledged to a nominal amplitude, further research is required.

3 ProvIntSec: Provenance Cognition Blueprint

As discussed in the related work, cloud provenance based trust issues can be drawn on the ground of the limited work done on provenance detection for cloud environments. The principal reason behind this is the limited availability of frameworks for file-centric and system-centric provenance tracking for the cloud.

The reliability and accountability of open source clouds must be ensured [10]. Provenance detection framework for the open source cloud architecture is needed to aid cloud forensic experts and audit trials. The mechanism may augment the security of cloud and increase acceptability among the business community.

A framework with effective performance analysis results must be presented to identify the process trees in the cloud and evaluate modifications to system essential processes. This will ensure better detection of forgery and malicious activities in open source cloud environments [29].

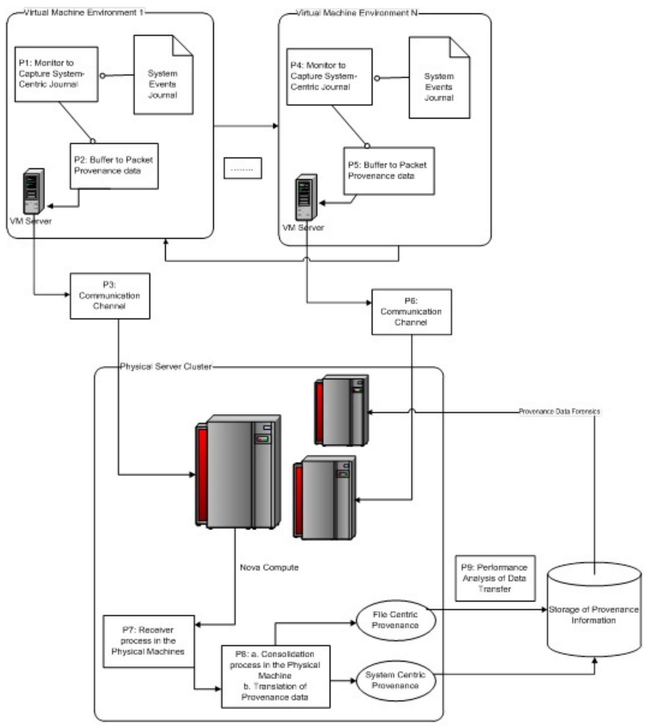

Virtual Machine Environment 1
P1: Monitor to Capture System-Centric Journal
System Events Journal
P2: Buffer to Packet Provenance data
VM Server
....
Virtual Machine Environment N
P4: Monitor to Capture System-Centric Journal
System Events Journal
P5: Buffer to Packet Provenance data
VM Server
P3: Communication Channel
P6: Communication Channel
Physical Server Cluster
Provenance Data Forensics
Nova Compute
P9: Performance Analysis of Data Transfer
Storage of Provenance Information
File Centric Provenance
P7: Receiver process in the Physical Machines
P8: a. Consolidation process in the Physical Machine
b. Translation of Provenance data
System Centric Provenance

Figure 1 Model showing journal based provenance detection in VM-PM mapping on the cloud

3.1 Provenance Framework

The need for journal based provenance detection scheme is manifold. Hence a framework for provenance detection to monitor earnest activities executed in open source cloud platform such as OpenStack is proposed in this paper. Figure 1 illustrate the set of processes P_1 to P_8 through which the provenance information parses before it can be analyzed.

For detection of worm in the cloud, a heuristic based approach is used that consists of pattern matching worm leftovers with previously identified keywords. The worm leftovers are collected from initially conducted tests. Next 8 worms were executed multiple times and the captured leftovers of those were pattern matched with the keywords. ProvIntSec parses through the system processes and journals from time to time collecting provenance information that matches the malware pathwords. Hence ProvIntSec is a pull-based provenance collector.

ProvIntSec collects provenance from cloud computation, storage and network processes. Hence it continuously monitors the cloud process journals. Such a provenance detector that specifically monitors cloud processes to detect the presence of malware is of limited availability. Secondly ProvIntSec reads provenance data from instances which are sent over cloud controller using the message passing server of the cloud. Hence ProvIntSec leverages cloud computing's message passing framework.

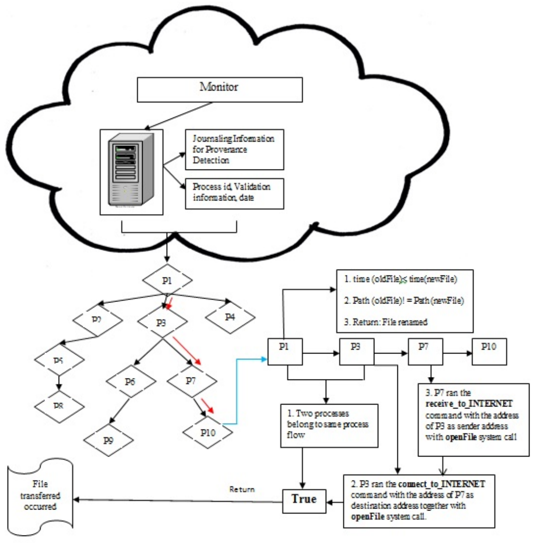

Monitor
Journaling Information for Provenance Detection
Process id, Validation information, date
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
1. time (oldFile)≤ time(newFile)
2. Path (oldFile)! = Path (newFile)
3. Return: File renamed
P1
P3
P7
P10
1. Two processes belong to same process flow
3. P7 ran the receive_to_INTERNET command with the address of P3 as sender address with openFile system call
2. P3 ran the connect_to_INTERNET command with the address of P7 as destination address together with openFile system call.
True
Return
File transferred occurred

Figure 2 Detailed functionality of the provenance framework that detects critical system file movements across multiple machines

The translated data can be used by data forensic experts which include analysis of the user centric and system centric journals for potential threats. At the same time cloud administrators can use the information to decide whether intrusions took place via the virtual machine instances running on the cloud.

ProvIntSec maintains a lookup table of allowable instance ids. The table consists of ids of the instances that have permission to transfer and receive files. One a file transfer has occurred, ProvIntSec identifies the instance ids of the VMs that took part in the transfer. Secondly the ids are searched in the lookup table to determine whether the instances are allowed to receive the file. If the instance id does not match with an id stored in the lookup table, the instance is not allowed to receive the file. Hence an unauthorized file transfer is detected and passed to the output file.

When files are manipulated across the same physical machine, only a single process can be analyzed to identify the change. However, for a distributed system such as the cloud, a process tree needs to be built which can be used to identify the change across multiple machines. Figure 2 shows a detailed functionality of the proposed scheme where a process tree is generated and tracked to identify changes made to system files and user files. The data captured from journal events are passed through different processes. As the file moves from one process to another within the same process tree, the system calls made on the file are detected. The main aim of the detection is to identify whether a file has been copied from the cloud environment to another physical machine outside the cloud. System important files can also be renamed within the same process. System critical files are those which if renamed, copied or accessed by unauthorized individual, can be used to hamper the proper functionality of the cloud infrastructure. Generally, the malicious code is a payload or a dropper with the sole aim of obtaining the lading and executing those on the server. The payload constitutes of multiple actions that creates backdoors. The enforcement of load on the active processes of the servers degrades cloud performance. At the same time recoiling system critical

$$RODx = \frac{(TDx)(100)}{TAx}, \forall x \in X \tag{1}$$

$$Ox = RODx - BLx, \forall x \in X \tag{2}$$

$$f(x) = \begin{cases} favorable & \text{if } Ox \text{ is greater than 0} \\ unfavorable & \text{if } Ox \text{ less than or equal to 0} \end{cases}, \forall x \in X \tag{3}$$

data and files will compromise critical cloud processes [29]. The malware tested in the experiments exhibited certain properties as illustrated below.

1. ***Tenacity and Secrecy:*** The ability of the malware to persist in the system without triggering the security mechanism of the system.
2. ***Blackout:*** The ability of the malware to pass critical system information to the attacker. The malware obtains unauthorized access to the machine and commits itself to system memory.
3. ***Dissemination:*** The capability of the malware to impose itself and attack the remaining machines of the system within a short period of primary attack.

To test the ability of ProvIntSec, malware with tenacity were executed. The test results recorded provenance information that helped to detect unauthorized file transfers that were critically important.

The malware codes were carried out in ten VM instances, with ProvIntSec installed in all of those. The VMs were created using OpenStack cloud and all had Ubuntu Servers installed as the base operating system. Malicious agents were acquired in one VM-instance, that spread to multiple VMs using the network. ProvIntSec gathered the provenance data and pattern matched those with leftovers of malware to detect the presence of malicious activities.

The precision of the proposed scheme is determined by finding the Rate of Detection (ROD). For the Total Amount of tests for malware x (TAx), the RODx is determined in terms of the Times Detected (TDx) variable. The performance gain (Op) for any precision of ProvIntSec is used to determine whether the obtained precision is favorable in terms of the baseline values. The Performance gain (Ox) for RODx is determined by the formula in equation (1). The results of this experiment satisfy the favorable condition f(x) as shown in equation (3). Based on Table 1, the precision graphs of ProvIntSec for the malwares can be seen in Figure 3. The figure illustrates that ProvIntSec is capable of assuring precision of over 80 percent for the malwares used in the experiments.

3.2 ProvintSec Algorithm

The algorithms of ProvIntSec involve the detection of provenance information and identification of malware and unauthorized file transmissions from those. Figure 4 shows the events in which a process P_1 monitors the journal events in Virtual Machine Environment 1 (VME_1) and passes the collected data to another process P_2. Process

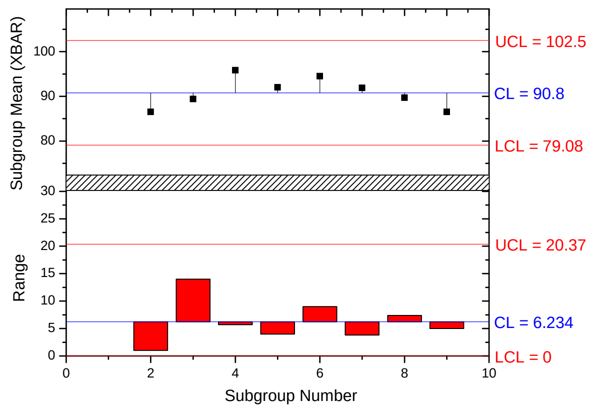
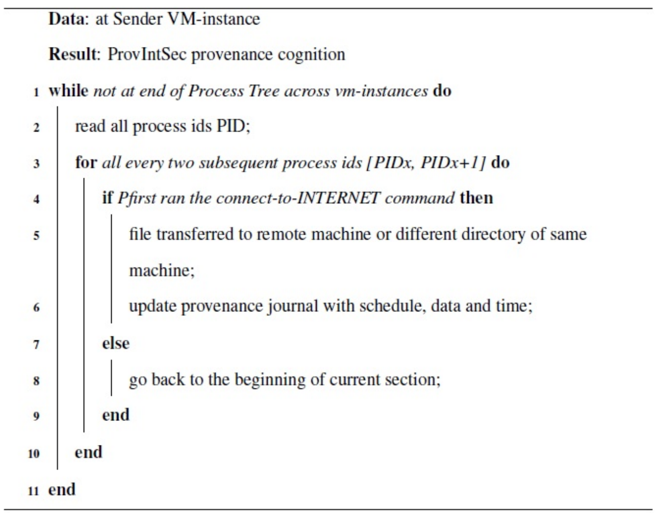


Figure 3 Precision Graph of ProvIntSec for the experimented results



```
Data: at Sender VM-instance
Result: ProvIntSec provenance cognition
1  while not at end of Process Tree across vm-instances do
2      read all process ids PID;
3      for all every two subsequent process ids [PIDx, PIDx+1] do
4          if Pfirst ran the connect-to-INTERNET command then
5              file transferred to remote machine or different directory of same
               machine;
6              update provenance journal with schedule, data and time;
7          else
8              go back to the beginning of current section;
9          end
10     end
11 end
```

Figure 4 Algorithm of ProvIntSec provenance detection at sender VM-instance

```
Data: at Receiver VM-instance
Result: ProvIntSec provenance cognition
while not at end of Process Tree across vm-instances do
    read all process ids PID;
    for all every two subsequent process ids [PIDy-1, PIDy] do
        if Pfirst+1 ran the connect-from-INTERNET command and OpenFile
        command executed using scp or mv then
            file received from remote or different directory of same machine;
            update provenance journal with schedule, data and time;
        else
            go back to the beginning of current section;
        end
    end
end
```

Figure 5 Algorithm of ProvIntSec provenance detection of data movement at the receiver VM-instance

P_3 passes the collected provenance over the cloud network from the VM to the cloud controller. The second process places the data in fixed sized blocks known as the data buffer. The buffer is then divided into fixed sized packets with header and footer information, and passed over a virtual transmission/communication channel to the physical servers. Since VME_1 is located at a specific physical machine, if the framework is implemented in the same machine, the transmission of packets is still required since VME_1 is completely separate virtual entity. Process P_4 is a process in the physical machines which compounds the free packets to obtain the buffer. Finally the buffers are unpacked using the header and footer information attached with those to obtain the data. After the information is obtained process P_5 organises those and prepares the information for consolidation and analysis.

Process P_8 consolidates the journal data into two groups. The data concerned with user files and documents are grouped into the user centric provenance object. The remaining system demanding process journal files are placed in the system centric provenance objects. The completion of the consolidation leads to the information being mathematically analyzed and translated through methods shown in the later sections.

Algorithms in Figure 4 and Figure 5 show the process of tracking when a file is copied from the master machine of the cloud to any other physical machine through secure copy (scp) command in Linux. At sender, all the processes from P_1 to P_n within the same process tree are parsed. For each consecutive process pairs, if the CONNECT-TO-INTERNET system function is called and before the openFile function is called, the flag of fileCopy has been returned. At receiver, the processes in the same process tree are parsed to identify two consecutive process pairs. Next the trigger of CONNECT-FROM-INTERNET process is detected through analysis of provenance journals. Finally the openFile process is triggered and ProvIntsec returns that the file has been received from a remote machine.

Table 1 Results of ProvIntSec in Malware Detection provenance

Malware Name	Type	Rounds	VMs	Detected	Precision	Baseline	Gain
Adore	Worm	200	Exp-1	174	87.0	86.0	0.01
Kippo	Worm	200	Exp-1	193	96.4	82.4	0.12
Lupper	Worm	200	Exp-1	197	98.7	93.0	0.06
Mighty	Worm	200	Exp-1	189	94.0	90.0	0.04
Millen	Worm	200	Exp-1	199	99.0	90.0	0.09
Satyr	Worm	200	Exp-2	181	90.0	93.8	0.04
Slapper	Worm	200	Exp-2	197	93.4	86.0	0.07
SSHBruteForce	Worm	200	Exp-2	168	84.0	89.0	0.05

The performance of the proposed framework needs to be evaluated through experimentation. The experimental environment is setup for open source cloud in Linux operating systems. The experimental results and analysis is given in Section 4.

4 Experiments and Results

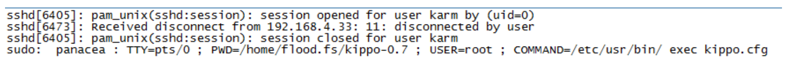
Malware that appear in real life distributed systems are diversified. In particular, any type of entity that causes unwanted or abnormal behavior can be a subject of investigation. Table 1 identifies the compiled results for malware detection. In the malware samples for our study, the malicious agents have the ability to cause blackout, maintain stealth and exfiltration. Such malware have been originally considered to attack real-life distributed systems and cause widespread damage [29].

The experiments were carried out in real life open source cloud environment on production servers. The testbed for the experiments consisted of OpenStack Essex cloud which was the latest OpenStack platform during the period of the research [29]. The cloud controller or compute service is the main process that controls launching and terminating of virtual machine instances. Provenance journals consist all the activities involving the cloud controller. The positive matches that are read from the output file are tabulated in Table 1.

4.1 Metrics for Malware Detection

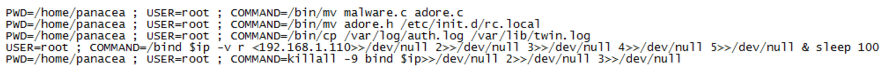
The primary requirement for detection of malware is to identify the malicious characteristics from the provenance journals. A malware such as a worm consists of numerous characteristics that are needed to be identified. Such characteristics are illustrated below.

1. The malicious code infects the VM-instances of both the cloud master and slave servers.
2. The attack imposes on causing malfunctioning of the important cloud processes.
3. The attack aims to persist as long as time permits.



```
sshd[6405]: pam_unix(sshd:session): session opened for user karm by (uid=0)
sshd[6473]: Received disconnect from 192.168.4.33: 11: disconnected by user
sshd[6405]: pam_unix(sshd:session): session closed for user karm
sudo:  panacea : TTY=pts/0 ; PWD=/home/flood.fs/kippo-0.7 ; USER=root ; COMMAND=/etc/usr/bin/ exec kippo.cfg
```

Figure 6 ProvIntSec provenance journal for Kippo worm tested in the experiment



```
PWD=/home/panacea ; USER=root ; COMMAND=/bin/mv malware.c adore.c
PWD=/home/panacea ; USER=root ; COMMAND=/bin/mv adore.h /etc/init.d/rc.local
PWD=/home/panacea ; USER=root ; COMMAND=/bin/cp /var/log/auth.log /var/lib/twin.log
USER=root ; COMMAND=/bind $ip -v r <192.168.1.110>>/dev/null 2>>/dev/null 3>>/dev/null 4>>/dev/null 5>>/dev/null & sleep 100
PWD=/home/panacea ; USER=root ; COMMAND=killall -9 bind $ip>>/dev/null 2>>/dev/null 3>>/dev/null
```

Figure 7 ProvIntSec provenance journal for Adore worm obtained in the experiment

4.2 ProvIntSec for Malware Detection

As stated in the previous section, the malwares were executed 1600 times in the VM instances. The journals of ProvIntSec were analyzed to identify the critical path words which were leftovers for each malware. In this regard detection of malware standard test and performance of ProvIntSec have been described in this section.

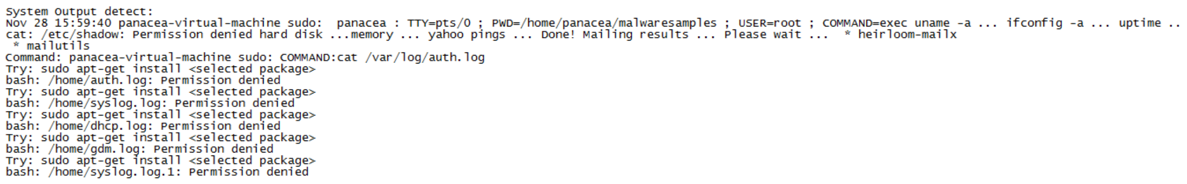
1. ***Detection of Tenacity and Secrecy:*** One of the primary actions taken by the malicious application is to ensure that the malware ran as long as time allowed. An important aspect of ensuring tenacity for a malware is to assure the secrecy factor, which is the ability to remain undetected or unrecognized while executing malicious objectives. Once the malware achieves secrecy, the system monitor is not being able to identify the malware as it crawls through the system. In order to rerun, the malware writes itself in system hard drive. As a result it is executed every time the system is rebooted. A goal of Linux malware is to successfully infect the *init.d* process, which runs every time a system is loaded.

   In the experiment of ProvIntsec, it was seen that Kippo, Satyr and Slapper worms infected the *init* process by affecting *rc.local* or *rc.h* files. ProvIntSec collected journals of the attacks that are shown in Figure 6.

   As seen in the ProvIntSec provenance journals collected for Slapper worm, signs were obtained that indicated the infection of the *rc.h* file. Upon comparison with the leftovers of the worm, positive match was found between the malicious leftovers and the provenance journals.

2. ***Detection of Blackout:*** Blackout is the process of secretly removing or copying important system files or data and transferring those to the remote attacker. In this regard, the Adore, Kippo, Satyr and SSHBruteForce show strong signs of blackout as they transfer data to remote host. In this experiment the attack was made from a single malicious VM instance to multiple VMs. The aim was to transfer the authentication logs of the target VM to the attacker. The provenance journals of the proposed scheme detected the above activity as shown in Figure 7 through subsequent match with worm leftovers.

   The worm called SSHBruteForce monitors earnest system processes such as accessing web, checking ping status, mail and system uptime services. Next the worm blocked those services for the authenticated user and provided malicious attackers access to the VMs' web based dashboards. Afterwords the attacker can steal important system data to a remote node using handshaking between the target and a malicious entity. During this experiment, this activity of SSHBruteForce was captured in the journal block of ProvIntSec as shown in Figure 8.



```
System Output detect:
Nov 28 15:59:40 panacea-virtual-machine sudo:  panacea : TTY=pts/0 ; PWD=/home/panacea/malwaresamples ; USER=root ; COMMAND=exec uname -a ... ifconfig -a ... uptime ..
cat: /etc/shadow: Permission denied hard disk ...memory ... yahoo pings ... Done! Mailing results ... Please wait ...   * heirloom-mailx
 * mailutils
Command: panacea-virtual-machine sudo: COMMAND:cat /var/log/auth.log
Try: sudo apt-get install <selected package>
bash: /home/auth.log: Permission denied
Try: sudo apt-get install <selected package>
bash: /home/syslog.log: Permission denied
Try: sudo apt-get install <selected package>
bash: /home/dhcp.log: Permission denied
Try: sudo apt-get install <selected package>
bash: /home/gdm.log: Permission denied
Try: sudo apt-get install <selected package>
bash: /home/syslog.log.1: Permission denied
```

Figure 8 ProvIntSec provenance journal for SSHBruteforce worms obtained in the experiment

Table 2 Results of ProvIntSec in Socket provenance detection

File Size	Total	Detected	Missed	Accuracy	Baseline	Gain	Time
1KB<=X<1MB	800	622	178	77.8	80.0	+2.25	0.874
5MB	800	628	172	78.5	80.0	+1.50	2.129
20MB	800	669	131	83.6	80.0	-3.63	6.820
128MB	800	647	153	80.9	80.0	-0.88	13.83
256MB	800	661	139	82.6	80.0	-2.63	16.44
512MB	800	664	136	83.0	80.0	-3.00	18.22
1GB	800	659	141	82.4	80.0	-2.38	19.93

3. ***Detection of Dissemination:*** The experiments involved worms that spread through the Hyper Text Transmission Protocol (HTTP) using the Apache Web Service. The stated activity was captured by ProvIntSec in the case of Mighty worm. The worm disseminates by connecting to a socket port of another VM instance from the attacker site. Next the socket address is used to send HTTP request to the target VM instance of cloud. The malware code is hidden in the request from the target VM. Once the request is accepted by the target VM, the worm crawled through the VM instance. Infectious operations such as obtaining unauthorized access to machines using the OpenSSL security system are executed by the worm as well.

   ProvIntSec analyzed obtained journals to detect the unauthorized access of OpenSSL security system and matched the entities with leftovers to detect the presence of Mighty worm. The framework successfully detected the existence of Satyr worm that binded logical memory and disseminated whenever that memory block was copied. If the memory block is copied to the master machine, the master machine is infected. Such inferences were deduced from ProvIntSec using leftover matching.

With respect to equation (3), performance gain implies that the provenance scheme is favorable for detection of the malicious activity. Figure **??** shows that ProvIntSec performs the best for Lupper worm, and worst for SSHBruteForce with a least gain in performance. Based on the proposed model a risk matrix identifying the probability of occurrence of the malwares tested in this experiment together with their threat levels is presented in Figure 10.

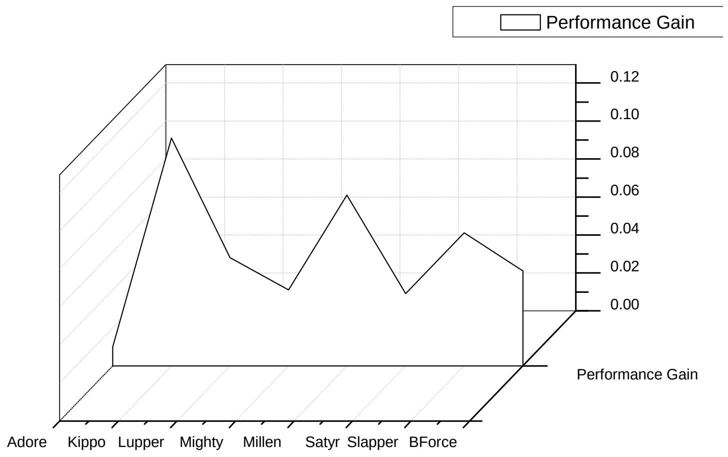
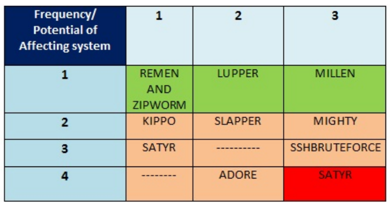


Figure 9 ProvIntSec Performance Gain Graph



Frequency/ Potential of Affecting system	1	2	3
1	REMEN AND ZIPWORM	LUPPER	MILLEN
2	KIPPO	SLAPPER	MIGHTY
3	SATYR	----------	SSHBRUTEFORCE
4	--------	ADORE	SATYR

Figure 10 ProvIntSec Risk Matrix for 10 Malicious Linux Worms

4.3 ProvIntSec for Socket provenance Detection

Implementation of socket provenance involved variable sized data files as data sets. The data were collected from real life operational servers. Data sets of variable sized files were transferred over 800 times during the experimentation. The size of the files ranged from 1 Kilobyte to 10 Gigabytes. Files transferred with secure copy (scp) or move (mv) command in Linux was detected through inspection of the provenance journals across multiple VM instances. The number of successful detections from ProvIntSec and total number of file transfers were identified and tabulated in Table 2. The relationship between the transfer time and precision of ProvIntSec were derived from the table.

As seen in the previous section, ProvIntSec's ability for detection of malicious entities exhibited desirable performance. Similar results were obtained for socket based provenance detection using ProvIntSec. A challenge of provenance detection in cloud is to determine file transfers across multiple VM instances. Due to the highly distributed nature of the cloud, keeping track of the large number of file movements and identifying unauthorized file transfers is a significant challenge. The performance of ProvIntSec for file transfer detection were tested in real-life cloud environment. ProvIntSec identified file transfers which constituted of the entire process tree across multiple VMs involved in the transmission process. Secondly the time of action and command executed

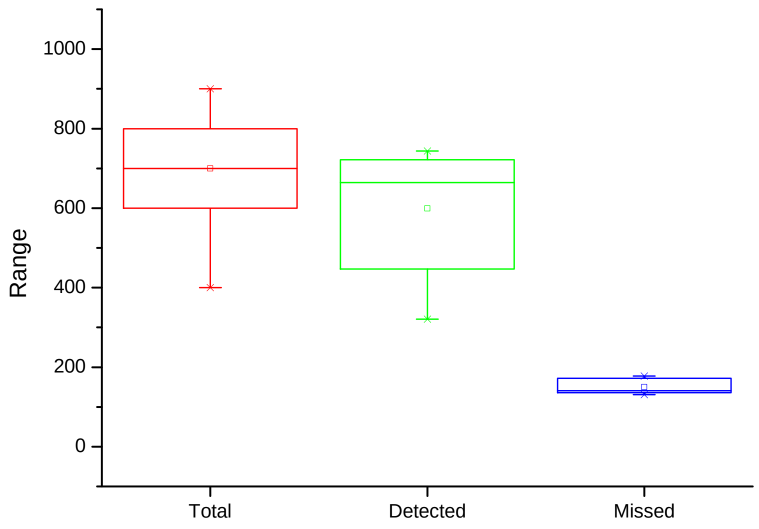


Figure 11 ProvIntSec precision graph of Level of Accuracy to File Size

for transferring the data were identified. Finally intermediate child processes that participated in the transmission of data were also detected to capture file transfers across multiple VM instances.

5 Discussion

The study aimed to detect provenance to detect the presence of malware and unauthorized file transfers in the cloud. The experiments carried in the previous section are analyzed and performances evaluated. Figure 11 shows the accuracy level of ProvIntSec for the different file sizes expressed as percentage. As seen in the figure, small sized file transfers show less accuracy in provenance capture. The reason for the stated behavior is due to the inability of Linux *inotify* to capture multiple file movements within 0.018 seconds. Since Linux *inotify* has been used in the experiments as the base of ProvIntSec, small file sizes pose a comparatively less performance gain. So the proposed model harnesses the strength of Linux systems to obtain high precisions.

Contrary to the above, desirable performance are identified for large file sizes. The unhindered performance was achieved as Linux *inotify* is capable of detecting files transferred at a rate of greater than 0.018 seconds. The results show that ProvIntSec performs better for larger data files. The tabulated results of this experiment shown in Table 2 depicts the baseline which is 80% for all file sizes.

The density graph in Figure 12 identify the number of successful detections out of the total of 800 rounds of transmissions. During the experimentation process, a total of 5600 rounds of files have been transferred in real-life cloud environment. Out of the total number of files transferred, 4550 rounds were detected with a cumulative performance gain of 8.77. The positive performance implies that the system does not have significant factors that can degrade the performance of ProvIntSec.

Finally Figure 13 identifies through the scattered diagram, the gain in performance of respective test cases for different file sizes. The comparative performance distribution of ProvIntSec for differing file sizes is highlighted in the figure.

For both malware provenance and socket provenance detections, ProvIntSec show favorable performance with respect to the benchmarks. The ability to detect malware presence from provenance information will enable Cloud Administrators to identify nefarious activities. Also detection of unauthorized file transfers across multiple VMs will ensure integrity and security of open source cloud. The ability of ProvIntSec to detect provenance for two types of operations, socket and malicious, make ProvIntSec complete All Provenance Detector (APD) [30].

It is well known that success of provenance detection scheme depends on the system it is applied. ProvIntSec has shown desirable performance for the cloud environment. This paper aims to solve the problem of provenance detection in the cloud. The solutions to the stated problems have been summarized below.

1. ***Limited work on Provenance collection for open source cloud:*** It is stated in the related work that research on journal-based provenance for open source cloud has been conducted to a limited extent [28]. The reason is most research in that area are funded by proprietary cloud service providers. Such projects have little interest in the open source cloud platform. This paper presents a journal based provenance detection model for open source cloud. The contribution of this paper will enable forensic experts to collect data for investigation.
2. ***Lack of accountability:*** The proposed provenance detection model in this paper will work to increase accountability of the cloud administrators. Cloud forensic experts can use the provenance information to detect whether any file was removed from the cloud storage.
3. ***Lack of trust by cloud customers:*** Potential customers of cloud services do not get enough trust on the cloud because of the limited capability to track the data in cloud storage. Information obtained from ProvIntSec could be shared with the customers to help them track the movement of the stored data. The framework will work to increase the trust, integrity and reliability of the cloud.

The study is based on provenance detection mechanism for the cloud itself. With respect to the above, there is an increased necessity to integrate ProvIntSec into the existing cloud architecture.

6 Conclusion and Future Work

This paper aims to ascertain a solution to the bottleneck of capturing provenance data in a widely decentralized system such as cloud computing. The aim was to device a blueprint that provides a journal based provenance detection framework for open source cloud computing.

In the experimentation, ProvIntSec successfully analyzes system important files of VM-instances as well as cloud controllers and nodes. The detected data are used

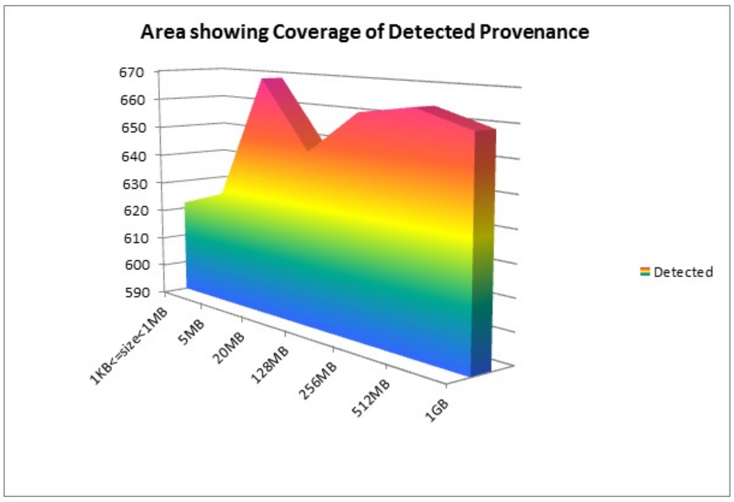


Figure 12 Area curve showing the extent to which file movement in the cloud has been detected using ProvIntSec

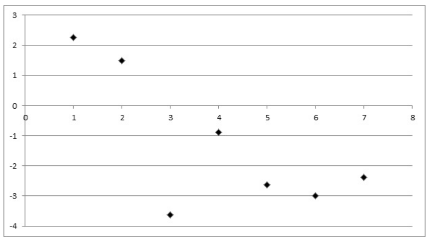


Figure 13 Scatter diagram showing the positions of performance gains of different test cases used in the experiment

to render useful information that enables cloud administrators to detect malware and unauthorized file transfers.

Traditional provenance detection techniques function in standalone systems. However ProvIntSec is capable of detecting provenance in a highly virtualized environment like the cloud. The results of ProvIntSec are obtained from a total number of 1600 runs, and 5600 file transfers that are carried out across VM-instances to detect socket provenance.

Empirical investigation is carried out to evaluate the performance of ProvIntSec in open source cloud environment. A precision rate of 92.81% and 81.24% are attained on average for malware and socket provenance respectively. Upon comparison of the test results with benchmark values, cumulative performance gains of +0.399 and +8.777 are obtained.

ProvIntSec detects and analyzes journals for the identification of Linux worms and unauthorized file movements. Performance evaluation for other malicious entities like rootkits, trojans and exploits can be considered as well. The case where contents of the old file are copied to the new file in a remote location of the distributed cloud environment can be tested for ProvIntSec.

References

[1] D.J. Pohly, S. McLaughlin, P. McDaniel and K. Butler. Hi-fi: Collecting high-fidelity whole-system provenance. 2012.

[2] Sultana, Salmin and Bertino, Elisa and Shehab, Mohamed A provenance based mechanism to identify malicious packet dropping adversaries in sensor networks. *31st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 332--338, IEEE, 2011.

[3] P. Storage. The case for content search of vm clouds. 2010.

[4] H. Yu, J. Rann and J. Zhan. SUCH: A Cloud Computing Management Tool. In *5th International Conference on New Technologies, Mobility and Security (NTMS)* , pages 1--4. IEEE, 2012.

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

[6] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50--58, 2010.

[7] Kolbitsch, C. Comparetti, P. M. Kruegel, Christopher Kirda, E Zhou, X. Wang, X. Feng. *Effective and efficient malware detection at the end host*. *Proceedings of the 18th conference on USENIX security symposium*, pages 351--366, USENIX Association, 2009.

[8] Moser, Andreas and Kruegel, Christopher and Kirda, Engin. *Limits of static analysis for malware detection*. *Computer Security Applications Conference*, pages 421--430, IEEE, 2007.

[9] R. Slipetskyy. *Security Issues in OpenStack*. PhD thesis, Norwegian University of Science and Technology, 2011.

[10] K.K. Muniswamy-Reddy, P. Macko and M. Seltzer. Provenance for the cloud. In *Proceedings of the 8th USENIX conference on File and storage technologies*, pages 15--14. USENIX Association, 2010.

[11] Bertino, Elisa and Dai, Chenyun and Kantarcioglu, Murat. The challenge of assuring data trustworthiness. In *Database Systems for Advanced Applications* , pages 22--33. Springer, 2009.

[12] Xu, Shouhuai and Qian, Haifeng and Wang, Fengying and Zhan, Zhenxin and Bertino, Elisa and Sandhu, Ravi. Trustworthy information: concepts and mechanisms. In *Web-Age Information Management* , pages 398--404. Springer, 2010.

[13] Dai, Chenyun and Lin, Dan and Bertino, Elisa and Kantarcioglu, Murat. ATrust evaluation of data provenance. In *Technical Report CERIAS TR 2001-147, Purdue University*, pages 2--4. 2008.

[14] Dai, Chenyun and Lim, Hyo-Sang and Bertino, Elisa and Moon, Yang-Sae. Assessing the trustworthiness of location data based on provenance. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 276--285. ACM, 2009.

[15] Xu, Shouhuai and Ni, Qun and Bertino, Elisa and Sandhu, Ravi A characterization of the problem of secure provenance management. *IEEE International Conference on Intelligence and Security Informatics (ISI'09)*, 310--314, IEEE, 2009.

[16] A. Haeberlen. A case for the accountable cloud. *ACM SIGOPS Operating Systems Review*, 44(2):52--57, 2010.

[17] R.K.L. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B.S. Lee. Trustcloud: A framework for accountability and trust in cloud computing. In *Services (SERVICES), 2011 IEEE World Congress on*, pages 584--588. IEEE, 2011.

[18] M. Mowbray, S. Pearson, and Y. Shen. Enhancing privacy in cloud computing via policy-based obfuscation. *The Journal of Supercomputing*, 61(2):267--291, 2012.

[19] R. Lu, X. Lin, X. Liang, and X.S. Shen. Secure provenance: the essential of bread and butter of data forensics in cloud computing. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 282--292. ACM, 2010.

[20] I.M. Abbadi. A framework for establishing trust in Cloud provenance. *International Journal of Information Security*, 1(2):1--18, 2012.

[21] S.B. Davidson, S. Khanna, S. Roy, J. Stoyanovich, V. Tannen, and Y. Chen. On provenance and privacy. In *Proceedings of the 14th International Conference on Database Theory*, pages 3--10. ACM, 2011.

[22] S.B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1345--1350. ACM, 2008.

[23] J. Yao, S. Chen, C. Wang, D. Levy, and J. Zic. Accountability as a service for the cloud. In *2010 IEEE International Conference on Services Computing (SCC)*, pages 81--88. IEEE, 2010.

[24] S. Chen, C. Wang. Accountability as a Service for the Cloud: From Concept to Implementation with BPEL. In *Services Computing (SCC), 2010 6th World Congress*, pages 91--98. IEEE, 2010.

[25] L. Zhou, V. Varadharajan, and M. Hitchens. A flexible cryptographic approach for secure data storage in the cloud using role--based access control. In *International Journal of Cloud Computing*, 1(2): 201--220, 2012.

[26] R.K.L. Ko, B.S. Lee, and S. Pearson. Towards achieving accountability, auditability and trust in cloud computing. *Advances in Computing and Communications*, pages 432--444, 2011.

[27] J. Wei, X. Zhang, G. Ammons, V. Bala, and P. Ning. Managing security of virtual machine images in a cloud environment. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 91--96. ACM, 2009.

[28] M. Imran, H. Hlavacs. Provenance in the Cloud: Why and How? In *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 106--112. 2012.

[29] S.N. Dhage, BB. Meshram. Intrusion detection system in cloud computing environment. *International Journal of Cloud Computing*, 1(2):261--282, 2012.

[30] M.M. Masud, T.M. Al-Khateeb, K.W. Hamlen, J. Gao, L. Khan, J. Han, and B. Thuraisingham. Cloud-based malware detection for evolving data streams. *ACM Transactions on Management Information Systems (TMIS)*, 2(3):16, 2011.