PROVINTSEC: A PROVENANCE COGNITION BLUEPRINT ENSURING INTEGRITY AND SECURITY FOR OPEN SOURCE CLOUD

ASIF IMRAN
BIT0119

A Thesis
Submitted to the Bachelor of Information Technology Program Office
of the Institute of Information Technology, University of Dhaka
in Partial Fulfillment of the
Requirements for the Degree

BACHELOR OF INFORMATION TECHNOLOGY
(SOFTWARE ENGINEERING)

Institute of Information Technology
University of Dhaka
DHAKA, BANGLADESH

PROVINTSEC: A PROVENANCE COGNITION BLUEPRINT ENSURING INTEGRITY AND SECURITY OF OPEN SOURCE CLOUD

ASIF IMRAN

Approved:

Signature *Date*

Supervisor: Dr. Kazi Muheymin-Us-Sakib

Committee Member: Dr. Md. Shariful Islam

Committee Member: Dr. Soo-Bong Kim

To *Dr. Kazi Muheymin-Us-Sakib*, my research supervisor

Abstract

Open source cloud computing provides free, on-demand, rapidly scaling, ubiquitous computing and data storage services that promises a significant number of prospective customer base. However, the largely distributed nature and growing demand for open source cloud makes the infrastructure an ideal target for malicious attacks and unauthorized file transfer activities. The cloud administrators are continuously worried about the potential for harmful system-centric and file-centric activities in open source cloud. Devising a provenance cognition scheme capable of collecting provenance information from largely complex and parallel system like the cloud has proven to increase the security and integrity of open source cloud platform. This in turn promises to accelerate the acceptability of open source cloud to the customers through increased trust on the security and integrity.

ProvIntSec is an effective provenance cognition mechanism that ensures collection of provenance information from a large pool of virtual machine (vm) instances running on the cloud, thereby addressing the stated problem. ProvIntSec analyzes critical system journals to collect provenance information from those which are then analyzed to detect malevolent exercises or unsanctioned file transfers in open source cloud environment. In this way the problem of ensuring the security, integrity and trust of the open source cloud is addressed.

The research analyzes the possibilities of detecting malicious activities and unauthorized file copy across multiple vm-instances in the cloud. Since the open source cloud environments are built on Linux platforms, the experiments involve executing numerous ladings and droppers of **10** Linux worms on Linux servers with open source OpenStack cloud. Next the provenance journals of ProvIntSec were collected and analyzed to identify malicious activities on the system. The obtained results of the experiments were compared against standard benchmarks to identify a performance

of over **92.81** percent precision for the test cases. A significantly low overhead of **-0.3991** was obtained through tests carried out with standard baseline values which prove the generalized capability of ProvIntSec to detect malicious worm activities in Linux based open source cloud platforms.

In terms of detecting file movements in the cloud using ProvIntSec, large number of file transfers of varying file sizes were sent over the cloud network across multiple vm-instances. ProvIntSecs provenance journals were then collected and the number of file transfers captured by ProvIntSec was measured. Upon calculation of the accuracy and comparison with predefined benchmarks, the precision of ProvIntSec in determination of file transfers in open source cloud was found to be above **81.24** percent with a cumulative overhead of **-8.77** which is highly desirable. The conclusion of this research suggests the effectiveness of the ProvIntSec algorithm and scope of improving the scheme to ameliorate provenance detection of open source cloud computing such as advanced file transfer techniques.

Preface

This thesis is submitted to the Bachelor of Information Technology Program Office of the Institute of Information Technology, University of Dhaka, in partial fulfillment of the requirements for the Degree. It contains the work done from June 2012. My supervisor on the research has been Dr. Kazi Muheymin-Us-Sakib, Program Chair, BIT Program Office. The thesis has been made solely by the author; some of the viewpoints are based on the research of others, and references to those sources are provided.

The purpose of engaging in the research was to solve the research issue of devising an effective provenance cognition scheme for the open source cloud. The proposition and performance evaluation of the open source provenance detection blueprint was the decisive factor in this research. The research is aimed to ensure security and integrity of open source cloud computing through effective provenance detection.

The reason for this research is the experience of the author in the field of cloud computing for over a year. The main inspiration came from the potential of open source cloud computing once it becomes more secured and increase in the trust of the people. The results of the experiments identify the areas where the proposed scheme performs better and also the scope of improvements and future work. The thesis concludes with a discussion of the research and the additional research questions that have come up in the course of this research.

The author conveys gratitude to the supervisor of this research for the continuous support and reviews. At the same time the author is grateful to all the members of the Institute of Information Technology, University of Dhaka and Panacea Systems Ltd, the software industry that contributed to the success of the research.

Acknowledgments

This thesis is about devising a scheme for provenance detection in open source cloud that highlights exactly my research interest for the last one year. Although the thesis is one semester long, I have been doing my research on this topic for more than one year now. It has been mainly possible for the constant effort of my thesis supervisor Dr. Kazi Muheymin-Us-Sakib sir, whose continuous support has enabled me to attain international publications in this topic. It all began in 2011 when our respected program chair, head of intern office and now my research supervisor Dr. Sakib sir sent me to Hayestech Pty Ltd to pursue my internship. It was at Hayestech where I got my first flavors of cloud computing and there was no looking back from then on. I am extremely grateful to Dr. Sakib sir for lending me this opportunity. From that moment I started to learn about the cloud using Google to good effect. I connected to many people who deal with this technology, electronically and face to face. Since it is impossible for me to thank them all, I will take this opportunity to mention the names of those without whom this study would have never been completed.

My motivation to proceed with the research got a tremendous boost when one of our research papers got accepted at the reputed HPCS 2012 conference to be held in Spain. Dr. Sakib sir was my supervisor in that research as well, and from that moment he has been guiding me in the research field, identifying my mistakes by going through every line of my write ups and my experimental results, investing significant portions of his valuable time to my development. At the same time he was always concerned about keeping my motivation high and maintaining a high spirit within the research team. Dr. Sakib sir, without your guidance I would have been lost in this vast research field a long time ago. Thank you so much for supervising, encouraging and helping me in the research and taking the burden and responsibility of reviewing my findings so that they gain acceptability.

I am also very grateful to Dr. Mahbubul Alam Joarder sir, in his role as the Director of the Institute of Information Technology, University of Dhaka, for allowing me pursue my research in a healthy and competitive atmosphere of IIT. I will also take this opportunity to thank Dr. Md. Shariful Islam sir and his panel for the effective feedback during the second presentation of my thesis.

As I said earlier, I first learnt the cloud at Hayestech. Hence I convey my heartfelt gratitude to Mr. Shaniur TIM Nabi, Chief Technology Officer (CEO) of Hayestech, for keeping faith in me and enabling me to learn the cloud as an intern at his company. Also, my heartiest thanks to Mr. Rezwanul Karim Ansari, Chief Executive Officer (CEO) of Panacea Systems Ltd, for appointing me as the Lead Research, Cloud Computing at his company. Hence I was fortunate enough to be provided with the powerful cloud servers of Panacea where I executed the tests and obtained the results of the thesis experiment.

Finally I would like to thank my family and friends who have directly and indirectly helped me in the research. Special thanks go to my mother and father, who have supported me and provided the logistics that enabled me to visit Madrid, Spain and present our research at the HPCS 2012 conference. My elder brother Nahid was always a support and source of confidence. I would like to thank all the students of BIT01 batch who are seeking completion of the Bachelor in Software Engineering (BSE) program through their participation in research and projects. Thanks go to the staffs of IIT, who have been a continuous source of help. The members of IIT family have really made this research fun for me and made the moments very enjoyable.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Open source cloud computing is becoming increasingly popular to cloud customers mainly due to provision of ubiquitous, on-demand, inexpensive and dynamically scaling services. The rapid adoption and distributed nature of open source cloud make it a prime target of malicious attacks and unauthorized transfer of system critical data across vm-instances of open source cloud [1]. This results in lack of trust from the customer end and thereby poses a threat to the large number of potential open source cloud customers. Provenance detection scheme that captures provenance data from a highly complex and distributed environment is a potential solution to the problem.

Cloud provenance detection is the process of capturing meta-data about the cloud environment that enables effective monitoring of the system with regard to malicious attacks and file transmission that use current vulnerabilities. The above problem of cloud security is compounded due to the problems associated with provenance detection from a highly virtualized environment from the cloud, the premiere bottleneck being provenance collection from a large number of virtual machines as well as physical machines. An effective provenance detection blueprint for the cloud will enable cloud administrators to monitor the large array of vm instances in an effective manner. The proposed scheme will aid in the detection of malevolent activities or transmission of unauthorized data across the cloud network.

1.1 Addressing the Research Problem

The provenance detection scheme for the cloud must possess the capability of analyzing system journals to detect the presence of nefarious activities. The problem

persists with the ever increasing number of vm-instances of the open source cloud providers, since collection of provenance data from all the vms is certainly an issue of research interest. Nefarious activities like malware attacks and unauthorized file transfers and copying can cause tremendous damage to the cloud customer by terminating critical processes and transferring important customer data to non-permitted bodies. The reputation of cloud service providers are affected in the process as well.

The malicious activities discussed above frequently occurs using stealth and the malware may easily persist in the cloud environment for a prolonged period if effective mechanisms of journal analysis are not implemented. As a result effective provenance detection mechanisms needs to be implemented for open source cloud computing environments which can easily observe and collect both system centric and data centric provenance and devise a blueprint to successfully identify malicious activities.

Another important characteristics of cloud environment is the movement of large number of files across a large pool of vm-instances. Critical and secret system files may be propagated across multiple vms and transferred to a remote host without authorization of the file owner. Malicious insiders can copy important files of the customer organization from the cloud servers and send those over the network. As a result of the problems identified, file-centric provenance detection for open source cloud is an issue with immense impact and research interest.

1.2 Cloud Provenance Cognition Scheme Characteristics

The provenance model should be such that an administrator should be able to monitor provenance data [1]. The model should easily provide up to date provenance data when required by the cloud administrator. A comprehensive provenance framework is

essential for researchers to verify quality of data [1]. Also, the provenance metadata should be linked to the data which it describes, as the self-descriptiveness will ensure that the data is easily comprehended. To make the provenance data part of the web data, both must adhere to the same publication principles [2]. The goal of this framework is to ensure the quality and trustworthiness of provenance data. In case of cloud computing, lack of physical access presents a new challenge for provenance collection [2].

Due to decentralized nature of cloud computing, traditional methods of collection metadata about data is no longer a practical approach [2]. Research needs to be done whether it is possible for cloud administrators to carry out provenance detection on their data which is stored on the cloud from a technical standpoint. Architectural idea for a trusted provenance system, and the requirements for effective analyzing of journal files of cloud computing systems should be identified for effective provenance detection.

With regard to the increasing need for provenance cognition blueprint of open source cloud, this research proposes a scheme that effectively parses through the system journal files of the vm-instances and physical machines of cloud environment and collects provenance information which can be used to identify malicious activities on the cloud. Cloud provenance schemes must be capable of detecting file movements across multiple machines in the distributed environment, and record the processes through which data are transferred across vm-instances using socket, providing the opportunity to identify the process tree. ProvIntSec is capable of achieving the above and hence addresses one of the most recent research issues on cloud computing.

1.3 Research Methodology

The research analyzes the problems for provenance stated above in the case of open source cloud environments. Since a significant portion of open source cloud platforms are based on Linux, so Linux servers were used for the experiments with open source cloud called OpenStack installed on those. VM-instances were launched using OpenStack and ProvIntSec installed on both the vm-instances and physical master servers and nodes of cloud. Figure 1.1 provides a view of the open source cloud computing environment set up for the experiment. The services shown in Figure 1.2 on the cloud would be monitored during the research and the various provenance journal files will be analyzed to keep track on the activities that occur on the cloud.

The target of the model is to capture the data on the journal files (i.e provenance data) and retrieve those so that those are presented in a readable format. Analysis of existing provenance protocols for the cloud is necessary to ensure that all the important aspects of cloud during the development of the proposed model are considered. In this experiment, to test for malicious activities, 10 Linux malware worms were considered as test cases in the first test. Each malicious worm was executed 200 times with the exception of Remen and ZipWorm [2]. Secondly, for evaluating performance in terms of provenance detection for file movements, files of 8 different file sizes were transferred 5600 times across vm-instances and from physical-to-virtual machines. Afterwards the provenance journals of ProvIntSec were inspected to identify the number of detections for both cases.

On the basis of standard benchmarks discussed above, the precision were determined for both the tests and the performances were compared to a set predefined baseline values. Next, the overhead of ProvIntSec was detected with respect to the baseline values to obtain a clear understanding of the performance. Upon obtaining the results, the capabilities and drawbacks of ProvIntSec was analyzed and areas

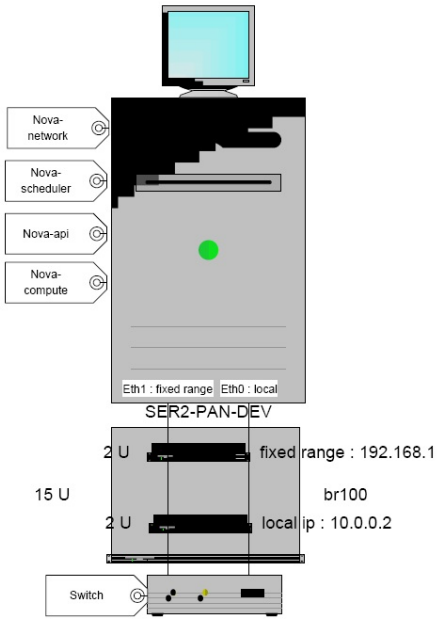


Figure 1.1: The Cloud Architecture for the research

where ProvIntSec is stronger were identified. Also the frequency and impact of the 10 worms used in the experiment when ProvIntSen is installed were provided in a risk matrix. A visual business model was also provided to enable open source cloud service providers to integrate ProvIntSec into their system.

Provenance of an output corresponds to the raw data and the processes that are involved in producing the output. Provenance is necessary to identify which process or user is responsible for creation of the final data product. Provenance aids data forensic experts to trace the origin of the data. Provenance is important for the cloud because it identifies when the data was modified, together with the users and processes which modified the data. Capturing provenance also includes records whether multiple data products were derived from the same raw data. It is seen that these factors are highly important for reusing and sharing the data stored on the cloud. At the same time these information enable scientists to repeat the results of experiments which is very much necessary for the computational process.

In general, recording provenance has significant social benefits in addition to increased popularity within the scientific research groups [3]. Provenance can be used to capture complex analysis of data which are processed on the computation services of cloud computing. Scientific calculations must retrieve information for the obtained data products [3]. For this reason provenance information on scientific experiments executed on the cloud must be retrieved systematically. Provenance information contains key aspects of scientific processes which is used for documentation purpose, since these data determine the ownership and quality of the workflows stored on the cloud, and reproduce the results of scientific experiments so Those can be validated and verified [3].

Complex processes of exploring large data volumes can be simplified if provenance data are used in the exploration process. Effective provenance models allow reusing the data that are stored on the cloud [3]. This is done by keeping the log of all the processes that are sequentially executed on the cloud using the data set. As a result the data can be reused by repeating the sequence of steps. Thereby, flexibility of data usage can be ensured on the cloud infrastructure.

Due to increased research concentration on performance analysis between various experiments, provenance can be used to compare the set of steps of one computation with another [3]. Complex data flows of the computational processes can be compared using the provenance captured on those, as it provides useful knowledge specifying the detailed methodologies of the two experiments. This information provides empirical data which can be used to compare the performance of the two processes.

A significant social impact of provenance can be seen in many of the educational institutions which store the knowledge banks on the cloud and use provenance data captured from logs to reap the benefits stated above [3]. Efficient explorations of large knowledge banks can be carried out using the precise information captured by

the provenance of that database [4]. Provenance models are mostly applicable for educational data that participate in large number of exploratory tasks such as data mining [4]. This is because throughout the experiments regarding data mining, the course instructors and students can use the provenance model to verify every step of the experiment where Those used data from the large data set. The provenance model can identify which command accessed the data banks and requested for the data. Students can also use the provenance data on the cloud to determine which steps and data are needed for specific assignments.

Another notable social impact of capturing provenance from cloud log files comes with the increasing demand for information systems. The information systems technologies are becoming greatly dependent on the compute and storage capabilities of the cloud due to their ability to provide these services at comparatively lower cost to traditional infrastructure [4]. With the growing demand for these applications in business organizations, the providers of these applications are using the cloud infrastructure to store the large volume of data. Managing such a large amount of applications is a huge challenge for cloud administrators [4]. Hence the demand for effective techniques to manage such information is highly demanded by cloud architects and administrators. Developing a model based approach to capture provenance of these data provides cloud administrators with the capability of analysing the steps through which these data were produced together with the ownership of the data. This yields better management of the information systems computation and data on the cloud.

From the above discussion it can be deduced that the social benefits of cloud computing can be increased to a substantial level by capturing provenance of cloud computation and storage. The benefits of repeating the computation, analyzing data for forensic purpose, and explore complex experiments to a finer detail by incorpo-

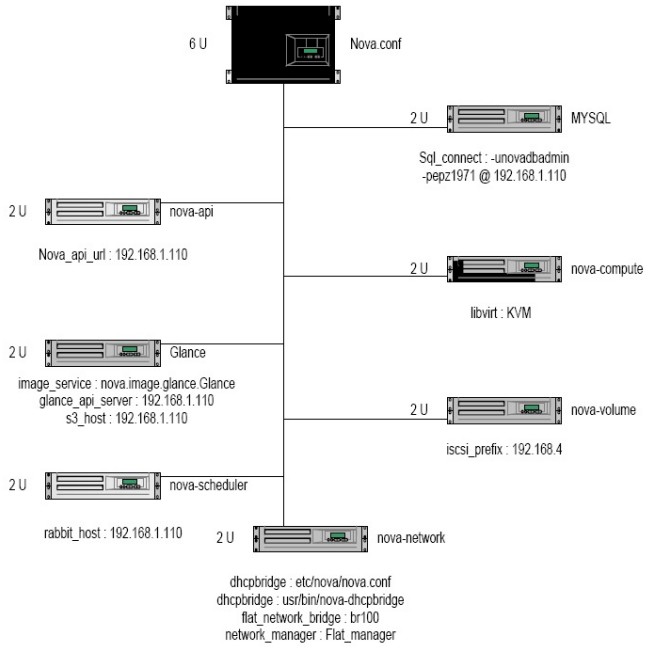


Figure 1.2: Simulated diagram depicting the various cloud services under provenance

rating the provenance service with the cloud infrastructure. Hence the aim of this research is to analyze existing provenance models and provide a method of capturing provenance in cloud data. The research will have impact of substantial significance and can be used by the companies to better manage their cloud. This research will play important role to make cloud more acceptable to the technology community. It will be a small step towards eradicating the common security concerns and social backdrops over cloud computing. This will be achieved by ensuring proper identification of data ownership and authorization, thereby providing greater data control to the authenticated owners.

The rest of the document proceeds as follows. Chapter 2 addresses the research issue and identifies the related research done in the field of provenance capturing and analysis. Chapter 3 gives a literature review of the journal based provenance detection and specifies the attributes of privacy aware provenance. Chapter 4 describes the ProvIntSec model, identifies the effectiveness of the novel mechanism and describes

how this framework solves the addressed problem. Chapter 5 describes the real life experimental bed which was used for the experiments and identifies the metrics of performance analysis. The chapter also describes the performance of ProvIntSec through the study of obtained results. Chapter 6 consists of the conclusion and discusses about the scope of future research in this field.

Chapter 2

Background

In this chapter, the distributed architecture of cloud computing, research assumptions and pre-requisites for the research have been analysed with respect to the research environment. Cloud has enabled many organizations to carry out complex computation with limited resources which would otherwise be impossible [5]. Cloud is an evolving technology, the basis of which is virtualization of physical servers to form virtual machines (vm) instances [5]. Despite the advancement in cloud, limitations exist in terms of security. However, cloud computing has limitations when it comes to acceptability, accountability, traceability and security. Customers are unwilling to provide their data and computation on the cloud despite the financial benefits, the chief reason for this is the loss of control and the security of tracking the activities which occur with the computation and data. There is a strong requirement to propose effective provenance mechanisms for this distributed and dynamic environment.

In addition to the problems posed by distributed applications, flexibility, representation and management of provenance information present new challenges for the cloud [5]. Log based provenance is required to support those. In order to solve these issues, there is an increasing need for research on file-centric and system-centric provenance detection from simultaneously generated process logs of cloud computing environment. Provenance detection will enable customers to keep track of their data and computation on the cloud.

Existing mechanisms are not suitable for cloud provenance. Ko. et al identified accountability and audibility as prime challenges for cloud computing [5]. Through real life scenarios of malicious cloud attacks, the vulnerability of cloud to attacks was shown. However the research did not focus on how provenance detection can effec-

tively prevent such attacks. Technical and procedural approaches to ensure security of cloud were discussed in [5]. Schemes such as anonymous login and identification of anonymous users through pseudomonas file read were proposed in the research. The challenges of the proposed provenance detection approach were not shown.

Specific challenges for provenance detection on the cloud must be addressed. These challenges must highlight log-based provenance tracking for file system logs and identification of platform specific provenance detection. Visual representations of provenance data are an additional challenge. Atomic actions at the system level of cloud and critical cloud process files should be identified and studies to find provenance of those. From analysis of existing research, a rule based provenance scheme must be proposed for provenance capture and representation.

Provenance in cloud computing is of increasing importance, as reflected by numerous research works [6],[7],[8]. Provenance in cloud is different from provenance of other distributed system since the provenance data should find the relationship between the physical machines and the virtual machine instances. This is done to aid cloud administrators identify which physical machine is providing computing and storage resources to which particular virtual machine instance.

2.1 Provenance

Provenance is the critical information to ensure security, reliability and trustworthiness of distributed computer environments [8]. Provenance is metadata that pertains to the derivation history of data artifacts [8]. Artifacts are the data objects, processes, operations, and other activities which are executed by the users and administrators of distributed environments. Information regarding the creation time, the modifier and the process used to modify the data stored or the processes running in distributed

environments allows system administrators and data forensic experts to estimate the validity and the reliability of data.

Recognizing the importance of provenance, a large body of research are concerned with the detection, storage, representation and analysis of the correctness of provenance data from provenance information captured real time or stored in provenance repositories. Many research organizations use provenance information collected from distributed applications to analyze and obtain answers to research questions. Hence many organizations are hosting their provenance data so that research communities can use the data and carry out experiments.

A specialized application within a pipeline of tasks is run in a distributed environment due to resource dependencies, that environment may collect and store provenance for all data products derived by that application. The distributed environment should provide the ability to collect and store provenance for all operations executed at the physical level as well as the virtual level, while provenance collected from other parts of the workflow may go to a different repository. As data sharing across heterogeneous environments becomes more common with multidisciplinary research, the need to query or reassemble provenance from across multiple repositories becomes increasingly necessary.

2.2 Assumptions

The research is taken forward while keeping in mind certain assumptions and obstacles regarding cloud computing provenance detection. The first two are related to security, the next two are related to adoption and the final two are derived from regulations and financial backdrops. Each problem is paired with an opportunity showing how provenance detection can help solve the issues, ranging from provenance detection to

analysis of the captured provenance data.

2.3 Data Confidentiality and Audibility

Security of the cloud is often the most cited objection since customers are severely concerned about the security of the data they store on the cloud. 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.

The cloud users are responsible for the security of the vm instances which are sanctioned. The cloud administrators are responsible for the security of the physical layers, as well as the security of vm layer by monitoring data regarding launching of vm instances and tracking logs of file access and changes made to those. So security of the intermediate layers and vm instances are shared among the cloud administrators and users. If the users are exposed to greater details of vm instances, more responsibilities are handed over from the administrator to the user. Some cloud providers outsource the security maintenance task to third parties who have independent IT management to manage the cloud services. Amazon EC2 users have more technical responsibility than Azure users, who in turn have more technical responsibilities than AppEngine users [9].

The primary security mechanism that can be used in todays cloud infrastructure

is log based provenance detection. Log based provenance detection can be used to monitor the vm instances from launching to termination [9]. This is a powerful tool to ensure confidentiality and audibility, since attempts by attackers to deactivate vm instances or stop underlying cloud processes can be identified. However, provenance detection must be based on log analysis and it is not bug free as certain safe actions can be considered as potential attacks. Incorrect provenance detection mechanisms can result in attacker having unauthorized access to the customers vm state information. The challenges are similar to large data centres which do not implement cloud computing, where various programs need to be protected from each other physically. Any large data centre, be it cloud or non-cloud, will look to detect provenance of the vm layer to detect malicious activities.

Similarly, audibility must be added as a security mechanism which would include provenance detection of physical data centres, which is beyond the reach of any outsider, be it a user or a malicious attacker. Provenance detection can provide full security to the vm layers and also to the physical layers, and makes the cloud arguably more secured as compared to other vendor-provided security applications which act only at the vm layer. Provenance detection scheme can ensure detection of any unauthorized activity about vm instances as well as the system itself. Such new features reinforce the shift in viewpoint of cloud computing from only the physical layer to the vm layer as well.

2.4 Provenance Overhead

Cloud provenance data is metadata about activities and operations. Usually a large amount of provenance information is gathered. The amount of computation and storage overhead required for provenance detection is an area of future research interest.

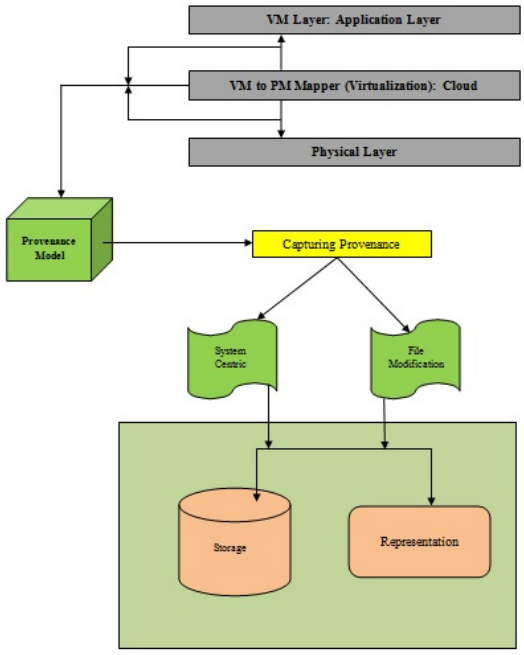


Figure 2.1: Provenance Storage Structure

How can provenance data be effectively stored is also an active research area. Also an effective storage procedure and storage unit of provenance data should be determined in future research. A provenance system must be user-friendly from the viewpoint of users and cloud service providers. The research challenge is to device a provenance mechanism that can be easily incorporated and removed from the cloud as wanted by the stakeholders. The goal of this research will be to make the cloud architecture unaffected from the provenance mechanism.

2.5 Limited availability

Compelling research is required to identify the various cloud computing processes and log the activities of those in order to accurately detect provenance and understand the behaviour of the cloud system. This is necessary to identify any problems that

may occur in the cloud administration.

One of the difficult challenges in cloud computing is the limited availability of provenance detection systems for open source cloud. A common occurrence is that these limitations are not present for proprietary cloud services, so most of the research for provenance detection is based on those. One opportunity may be devising a scheme that tracks provenance on the basis of both system and file-centric logs in Cloud Computing. Many open source cloud computing providers developed their infrastructure without using any provenance detection, either because the recent surge in security of cloud through provenance detection was not comprehended or the infrastructure was developed solely for research purpose. Log based provenance detection can capture information of vms which would otherwise been impossible for open source cloud.

2.6 Log-based Provenance Limitations

Provenance detection can be widely used in data forensics of the cloud to identify the root cause of system failures. The cloud should be available 24*7. However even reputed cloud service providers like Amazon EC2 and Google AppEngine have faced service breaks. With effective provenance detection of logs, data forensic experts have successfully determined the reason of the flaws. Such provenance detection schemes, if developed for open source cloud platforms, can also help detection of reasons for service breaks. Following tables illustrates some of the service disruptions of proprietary cloud services and the reasons identified through provenance information for such failures [10].

Table 2.1: Failures due to lack of provenance data

Service	Reason of Failure	Duration
Amazon S3	Authentication mechanisms overloaded by remote attacks	2 hours
Google AppEngine	Error from the end of the maintenance engineers program	4.6 hours
Gmail	The contact list mechanism crashed	1.4 hours

2.7 Compatibility and Automation

Cloud computing involves a large number of process execution, system calls, file operations within a short time period. It is very important to make the provenance detection mechanism completely automated and stable to handle the large data load.

The provenance detection scheme must be tested to ensure that failure does not occur. The scheme must have the same performance of scalability, reliability and fault tolerant as the cloud itself. In this regard two important testing research that should be carried out are failure testing and failover testing of the cloud.

2.8 Platform Dependency

Open source cloud requires detailed understanding, whereas proprietary clouds have closed source. The aim of this research is to develop a provenance detection model for the cloud that is independent of any platform and fits into any cloud environment.

Since large volume data is captured as provenance in the cloud, research should be carried to present his provenance data in a visual format to the viewers of provenance information. Visual representations of provenance data must be specific for the cloud customers, service providers and regulators since their needs are different. Hence research should be carried out in this regard.

2.9 Provenance in the Cloud

Provenance is information which outlines the history of an artefact or object. It must be mentioned that provenance is an essential component for data stored on the cloud and properties of provenance that enable its utility should be identify. Current cloud offerings and design to implement protocols for maintaining data/provenance in cloud stores has to be identified. The protocols should represent different points in the design space and satisfy different subsets of the provenance properties. One can select a protocol based upon the provided properties without sacrificing performance.

Despite the feasibility to provide provenance as an additional layer on top of the existing cloud, provenance should be incorporated in the cloud as a core cloud feature. Traditional work in storage and file systems addresses the storing of information and making it available to users. Provenance addresses that correct information are made available to the customers at the correct time. Provenance, sometimes called lineage, is metadata detailing the derivation of an object. If it were possible to fully capture provenance for digital documents and transactions, detecting insider trading, reproducing research results, and identifying the source of system break-ins would be easy. Unfortunately, the state of the art falls short of this ideal. Current research has demonstrated the feasibility of automatically capturing provenance at all levels of a system, from the operating system to applications [10].

Provenance is particularly crucial in the cloud, because data in the cloud can be shared widely and anonymously. Without provenance, data consumers have no means to verify its authenticity or identity. The web has taught us that widely shared, easy-to-publish data are useful, but it has also taught us to be sceptical consumers. For shared data on the cloud, it is impossible to know exactly how updated or trustworthy are the data on the web. We should solve the problem now while cloud services are still new and evolving.

2.10 Attributes of Provenance Systems for the Cloud

Provenance for the cloud must ensure that the access time, manipulation type and accessory are correctly identified. Following are some essential attributes which satisfy the stated capabilities.

1. ***Provenance Data Coupling:*** : The data-coupling property states that an object and its provenance must match that is, the provenance must accurately and completely describe the data. This property allows users to make accurate decisions using provenance. Without data-coupling, a client might use old data based on new provenance or might use new data based on old provenance. In both of these cases, the user relying on the provenance is misled into using invalid data. Systems that do not provide data-coupling during writes can detect data-coupling violations on access and withhold or explicitly identify objects without accurate provenance. For example, if the provenance includes a hash of the data, we can compute the hash of a data item to determine if its provenance refers to this version of that data.

   Detection is, at best, a mediocre replacement for data-coupling, because although users will not be misled, they cannot safely use available data when its provenance is wrong.

2. ***Eventual data coupling:*** In eventual data coupling, the data and its provenance might not be consistent at a particular instant, but are guaranteed to be eventually match. With eventual data-coupling, a system requires detection, since there may exist intervals during which an object and its provenance do not match.

3. ***Causal Ordering:*** This property acknowledges the causal relationship among objects. If an object O, is the result of transforming input data P, then the

provenance of O is the super-set of the provenance of P. Thus, a system must ensure that an objects ancestors (and their provenance) are persistent before making the object itself persistent. Multi-object Causal Ordering violations occur when the system writes an object to persistent store before writing all its ancestors, and the system crashes before recording those ancestors and their provenance. Similar to eventual data coupling, a weaker form of the property Eventual Causal Ordering is realizable. A system still requires detection to account for the intervals during which an objects provenance may be incomplete, because its ancestors and their provenance are not yet persistent or not available due to eventual consistency.

4. ***Data-Independent Persistence:*** This property ensures that a system retains an objects provenance, even if the object is removed. As in the last section, assume that P is an ancestor of O. If P were removed, Os provenance still includes the provenance of P, so a system must make sure to retain Ps provenance, even if P no longer exists. If Ps provenance is deleted when P is deleted, parts of the provenance DAG will become disconnected. If P had no descendants, then a system might choose to remove its provenance, since it would no longer be accessible via any provenance chain. Another approach to solving this problem is to copy and propagate an ancestors provenance to its descendants. This is inefficient in terms of space and can quickly become unwieldy.

5. ***Efficient Query:*** Since provenance is created more frequently than it is queried, efficient provenance recording is essential. However, efficient query is also important as provenance must be accessible to users who want to access or verify provenance properties of their data. In scenarios where the number of objects is few or users already know the objects whose provenance they want to

access, efficiency is not an issue. Efficiency matters, however, when the number of objects is sizeable and users are unsure of the objects they want to access. For example, users might want to retrieve objects whose provenance matches certain criteria. In scenarios such as his, if a system stores provenance, but that provenance is not easily queried, the provenance is of reduced value.

6. ***Rate of False Positives (FP):*** High rates of FP will result in increased workload in analyzing and responding to events. They may also result in reduced productivity due to the prevention of legitimate documents and messages from reaching employees.

7. ***Rate of False Negatives (FN):*** As with other security measures, a high rate of false negatives will lead to a false sense of security, plus potentially placing the organization in jeopardy from confidential data which is leaked without being identified. At the same time, provenance data should have the ability to detect data flooding, file type/format manipulation.

2.11 Critical Scheduling of Cloud Systems

The scheduler maps the nova-API calls to the appropriate OpenStack components. It runs as a daemon named nova-schedule and picks up a compute server from a pool of available resources depending on the scheduling algorithm in place.

1. ***No prior knowledge about incoming processes:*** A good process scheduling algorithm of distributed systems should work with no prior knowledge of the processes or operations which are to be executed. Algorithms which function based on prior information pose an additional burden on the users since the users then must explicitly specify the characteristics of the operation they

submit.

2. ***Dynamic load balancing capability:*** The algorithm should have the capability of managing the dynamically changing load of the nodes. Process assignment on different nodes should be based on the current load status of the nodes and not on some pre specified policy.

3. ***Thrashing Resistant:*** Processor thrashing occurs when the algorithm causes all the nodes of the system to spend all their time migrating processes without accomplishing any useful work.

4. ***Scalable:*** The algorithm should be capable of supporting a large number of nodes from a small number without computational complexity. The attribute must sustain even when the number of nodes is suddenly reduced.

5. ***Fault Tolerant:*** The algorithm should not suffer from crash of a single node of the system. The system should continue functioning with the available nodes which are up at that point of time. The cloud controller should be capable of provisioning resources from its own physical machines which would increase the fault tolerance capability of the system.

6. ***Quick decision making:*** The scheduling algorithm must exhibit good decision making capability. This is very important at times when the computational requirements of the processes are very high. For example, a customer executing scientific calculations on the instances may require sudden requirement of storage. The algorithm must be able to provide the required storage without termination of the instances.

A scheduler can base its decisions on various factors such as load, memory, physical distance of the availability zone, CPU architecture, etc. The nova scheduler

implements a pluggable architecture. Currently the nova-scheduler implements a few basic scheduling algorithms

1. ***Chance:*** In this method, a compute host is chosen randomly across availability zones.

2. ***Availability zone:*** Similar to chance, but the compute host is chosen randomly from within a specified availability zone.

3. ***Simple:*** In this method, hosts whose load is least are chosen to run the instance. The load information may be fetched from a load balancer.

2.12 Load Balancing approach of the resource scheduler

In this approach all the processes submitted by the user are distributed among the nodes of the system so as to equalize the workloads among the nodes. Another approach is the load sharing approach in which the target is to conserve the ability of the system to perform work by assuring that no node is idle while processes wait for being processed. Following are a number of desirable features for the distributed environment scheduler of the cloud.

2.13 Message passing mechanisms in distributed environment

Traditional interprocess communication (IPC) systems contain protocols developed on the message passing model. Despite the protocols ability to support distributed

applications, their flexibility remains a lot to be desired [11]. As a result applications of heterogeneous types are not supported in such IPC systems. Remote Procedure Call (RPC) provides a communication mechanism that solves this problem. This mechanism can support distributed applications with heterogeneous attributes.

In this chapter, a developed distributed application that consists of taking two integers as input at the client then sending the numbers over to the server and using the RPC model and executing the mathematical equation.

The report show how RPC mechanism is implemented on the basis of send and receive primitives for communication with a disjoint machine. It has also been identified that how various components of RPC can be used to satisfy the applications requirements. Also the importance of RPC elements to abstract the internal complexity of message passing from the user is shown. The RPC mechanism used to implement this application consists of the RPCRuntime, that hides the underlying network details from the end user. Five important elements of the RPC technique used here are: Client process, Client stub, RPCRuntime, Server process and Server stub.

1. ***At Client:*** 1.1 A call is generated from the client process to the client stub. 1.2 The client stub packs the request with the correct parameters and sends it over to the RPCRuntime for transferring it to the server. 1.3 The RPCRuntime transfers the message from the client to the server machine over the network.

2. ***At Server:*** 2.1 The RPCRuntime of the server receives the message. The message is passed to the server stub. 2.2 The message is received and unpacked at the serve stub. The stub invokes the call to teh desired server process. 2.3 On receiving the unpacked message, the required actions are executed by the server process and the results are forwarded to the server stub.

3. ***Attributes of the application:*** The application is based on stateless server mechanism. The moment the connection is lost, the server does not keep track of the client machine. At the same time the parameter arguments are called by reference. As a result of this application the server and clients are on the same network.

4. ***Limitations:*** The application can be extended to a great length. The application will not work on remote machines outside the network of the server. The client executable must be run each time a pair of numbers are given as input

2.14 Rabbit MQ Server

OpenStack communicates among themselves using the message queue via AMQP(Advanced Message Queue Protocol). Nova uses asynchronous calls for request response, with a call-back that gets triggered once a response is received. Since asynchronous communication is used, none of the user actions get stuck for long in a waiting state. This is effective since many actions expected by the API calls such as launching an instance or uploading an image are time consuming.

2.15 Critical Security process identified for distributed applications

Provenance information should be able to thwart a number of security breaches in the distributed environment. The cloud computing major security concern need to be listed and the attacks which can be detected through the analysis of provenance information must be identified at an early stage of the research. There are two

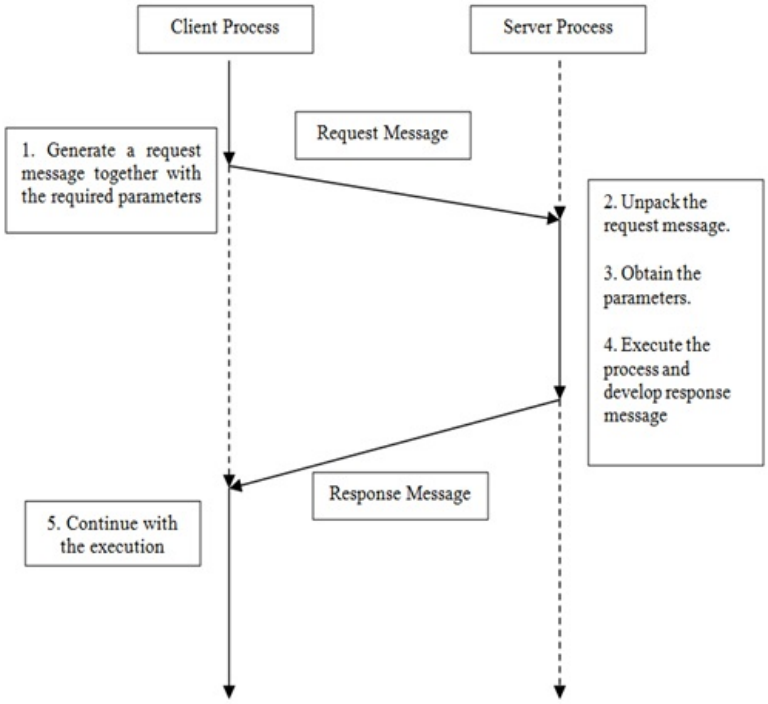


Figure 2.2: Message Passing in the Cloud

approaches to management of malicious cloud activities which are discussed below.

1. ***Preventive approach:*** Preventive approach includes prevention of attacks through implementation of security features at the physical levels. These include installing firewalls and network filters to prevent attacks from being made [11].

2. ***Detective approach:*** The detective approach involves implementation of methods to detect and monitor potential malicious activities. The activities include detection of data leakage, any misbehaviour in the administration of the cloud and detection of potential attacks on the basis of provenance data collected from log files [11].

3. ***Estimation of risk levels:*** The level of risk is estimated on the basis of the likelihood of an incident scenario, mapped against the estimated negative impact. The likelihood of an incident scenario is given by a threat exploiting

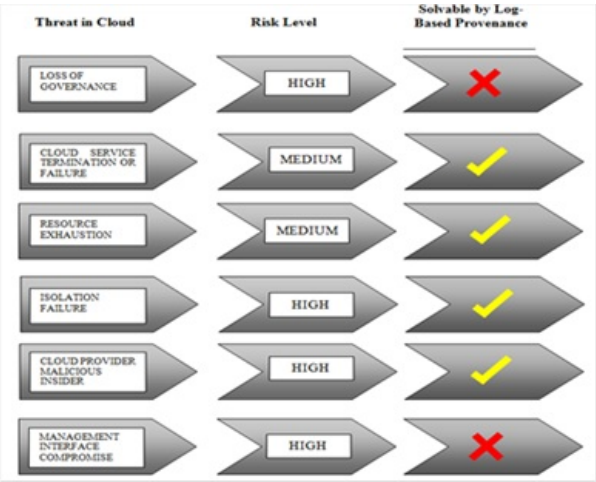


Figure 2.3: List of cloud computing risks and those solvable through provenance

vulnerability with a given likelihood. The likelihood of each incident scenario and the business impact was determined in consultation with the expert group contributing to this report, drawing on their collective experience. In cases where it was judged not possible to provide a well-funded estimation of the likelihood of an occurrence, the value is non-applicable. In many cases the estimate of likelihood depends heavily on the cloud model or architecture under consideration. The following shows the risk level as a function of the business impact and likelihood of the incident scenario. The resulting risk is measured on a scale of 0 to 8 that can be evaluated against risk acceptance criteria. This risk scale could also be mapped to a simple overall risk rating.

(a) ***Low risk:*** 0-2

(b) ***Medium risk:*** 3-5

(c) ***High Risk:*** 6-8

Cloud services are not only about convenient storage, accessible by multiple devices, but include important benefits such as more convenient communication and instant multi-point collaboration. Therefore, a comparative analysis needs

to compare not only the risks of storing data in different places (on premises v the cloud) but also the risks when on premises-data stored on premises, for example, a spreadsheet - is emailed to other persons for their contributions, against the security issues of a spreadsheet stored in the cloud and open to collaboration between those persons. Therefore, the risks of using cloud computing should be compared to the risks of staying with traditional solutions, such as desktop-based models. It is possible for the cloud customer to transfer risk to the cloud provider and the risks should be considered against the cost benefit received from the services. However not all risks can be transferred: if a risk leads to the failure of a business, serious damage to reputation or legal implications, it is hard or impossible for any other party to compensate for this damage. The level of risks is expressed from the perspective of the cloud customer. Where the cloud provider point of view is considered, this is explicitly stated. Following are a list of cloud risks and the risk levels provided by Gartner [11].

Chapter 3

Literature Review

In this chapter, the research issues in provenance detection, storage, secure access and user involvement have been highlighted. The chapter focuses on the existing research work that has been carried out regarding provenance. At the same time the scope of research in this growing field are highlighted. With regard to the rapid growth of cloud, effective provenance detection will ensure that the data stored on the cloud is reliable. Provenance data based on system and operation logs will ensure the integrity of that data. Identification of provenance attributes and obstacles to provenance detection in the cloud is necessary for user-friendliness and effectiveness of provenance detection. Also research in the field of provenance will ease cloud maintainability and at the same time develop user confidence. Regarding digital forensics, the loss of control caused by Cloud environments and vendors presents a huge challenge for investigators. Preliminary ndings of the computer forensic community in the eld of digital forensics have to be revised and adapted to then environment. Investigators need the possibility of reconstructing the corresponding environment for recreating scenarios and test hypothesizes. Hence the scope of work in this area is manifold.

3.1 Provenance Storage

The remote storage of data and remote computation is considered as a critical risk for both the cloud service provider and customer. Cloud computing carries risk for both the customer and provider. The customers provide their data and computation on machines that are located on a remote place which the customers cannot physically control. At the same time, cloud service providers agree to provide computation

services for operations which they are not aware. In case of a problem such as data leakage or incorrect computation, it is quite difficult to identify whether the customer or the provider has caused the problem. In cloud computing environment it is impossible to help a body responsible without proper evidence.The research proposed that accountability must be ensured for both the customers and service providers of cloud. In case of a problem, both parties must be made aware about the liabilities of the problem. Secure auditing, recording and evidence of information stored on the cloud was proposed as probable solutions to the issue. Technical requirements for an accountable cloud have been identified and the challenges to ensure this accountability but have not yet been met are also outlined.

From a customers perspective, a cloud can contain significant amount of risk since the customers must relinquish control over their data and computation. In the traditional model, the compute servers are located in data centers at the customers premises. The customer obtains direct access to the machines and monitors the status of those. The servers are managed by administrators whom the customers trust. In cloud computing, all the computations are carried out in the virtual machines and the customers cannot execute the above tasks on their own. The cloud provider is responsible for maintenance of the physical servers. Customers have control only over the virtual machines, which is remotely managed over a network connection. The potential problems that might arise when control is lost are given below.

1. The machines in the cloud can be defective and can cause corruption of the customers data on a continuous basis or cause the computation to give wrong results as output.

2. The cloud service provider can provide a wrong flavor to the customer which may consist of insufficient resources. This may lead to loss in performance and slow speed of the virtual machines.

3. The attacker can take over the virtual machines and cause Distributed Denial of Service (DDoS) attacks. Also the attacker can steal valuable data from the cloud storage, hence customer data may be stolen.

4. The cloud service might be down, hence the customer may not be able to retrieve or access the data at a convenient time which may lead to loss in business.

The absence of reliable fault detection does not only discourage customers, it may also lead to the prevention of provision of services on the cloud which requires compliance with laws and regulations. Even customers of reliable cloud service like Amazon require the removal of Protected Health Information (PHI) before uploading the data on the Amazon Web Services (AWS) cloud [5]. Accountability has been proposed to address the mentioned challenges. A system is said to be accountable if 1) faults are effectively detected and 2) each faults can be connected to at least one of the nodes which is not functioning properly. Accountability appears to be a promising approach to the problems defined above since customers will then be able to check that the cloud is performing as promised by the provider. If a problem occurs, customers and providers can then decide who is the faulty party on the basis of the accountability information that is detected, and in case of a dispute, the evidence gathered can be presented to a third party which is an independent body.

However, the importance of log based analysis of provenance data has been covered to a little extent. Log based provenance detection should be an ideal solution for the above problem statement as it can track which partys activity caused the problem. Also tasks like system and operation centric file provenance detection has been covered to a limited extent.

3.2 Privacy Aware Provenance

Recent technologies for provenance detection and scientific workflow systems are aimed for the group concerned with scientific workflow and provenance storage in database [12]. The objective of the research is to give an overview of the problem of managing provenance data for scientific workflows, illustrate some of the techniques that have been developed to address different aspects of the problem, and outline interesting directions for future work in the area. In particular, techniques for reducing provenance overload as well as making provenance information more fine-grained have been presented. Provenance that go beyond the ability to reproduce and share results was examined, and workflow evolution provenance which can be leveraged to explain difference in data products was demonstrated, exploratory computational tasks were streamlined, and knowledge re-use was enabled. New applications that are enabled by provenance were also discussed, such as social data analysis [12], which has the potential to change the way people explore data and execute scientific experiments. On the basis of the mentioned experiments it was shown that wworkflow systems help scientists conceptualize and manage the analysis process, support scientists by allowing the creation and reuse of analysis tasks, aid in the discovery process by managing the data used and generated at each step, and systematically record provenance information for later use. Three key components of a provenance management solution are provided below.

1. ***Mechanism:*** The mechanism for capturing provenance
2. ***Data model:*** The data model for representing provenance information
3. ***Infrastructure:*** Infrastructure for storing, accessing, and querying provenance

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. In the context of scientific workflows, provenance is a record of the derivation of a set of results. There are two distinct forms of provenance prospective and retrospective as defined in the research. Prospective provenance captures the specification of a computational task that is a workflowit corresponds to the steps that need to be followed (or a recipe) to generate a data product or class of data products [13]. Retrospective provenance captures the steps that were executed as well as information about the execution environment used to derive a specific data product and a detailed log of the execution of a computational task [13].

In accordance to the two types of provenance mentioned above, a common feature across many of the approaches to querying those was that the solutions were closely tied to the storage models used. A wide variety of data models and storage systems had been used ranging from specialized Semantic Web languages and XML dialects that were stored as files and to tuples stored in relational database tables. Hence, they require users to write queries in languages like SQL, Prolog and SPARQL [13].

A number of emerging applications for workflow provenance were analyzed and the challenges posed by those to database research were stated such as provenance and scientific publications in which there is a key benefit for maintaining provenance of computational results is reproducibility as a detailed record of the steps followed to produce a result allows others to reproduce and validate these results. Another challenge is that provenance and data exploration which ensures that provenance can also be used to simplify exploratory processes. In particular, mechanisms those allow the flexible re-use of workflows, scalable exploration of large parameter spaces and comparison of data products as well as their corresponding workflows have been

identified [13].

In addition to the above, use of provenance for social analysis of scientific data has been addressed in this research. Social websites and web based communities which facilitate collaboration and sharing between users, are becoming increasingly popular. Also a major challenge identified in the research involves provenance in education which stated that using a provenance enabled tool in class, an instructor can keep detailed record of all the steps which were tried while responding to students questions and after the class all those can be made available to students through the use of provenance.

The above research deals with the principal task of identifying the challenges and areas of application of provenance. The role and importance of provenance detection for successful administration of complex and distributed environments such as the cloud have been considered to a minimal extent [14].

Accountability as a Service (AccS) has been identified for cloud computing environments in [15]. The research presented a model in which cloud service providers are held responsible for accountability issues related to the services deployed in the cloud. In this paper, a novel design to achieve Trustworthy Service Oriented Architecture (TSOA) in the Cloud through enforcing strong accountability was proposed. In such system not only the root of a fault can always be concluded to the guilty participants, but also each conclusion is supported with non-disputable evidence [15].This is achieved by making the service provider accountable for the faults and breaches in the Service Level Agreement (SLA). Similarly, the Service Oriented Architecture (SOA) as a design paradigm allows for new value added services in terms of compositions of existing services from different service providers [15]. The adoption of the cloud and SOA result in a cloud computing environment that enables highly dynamic and effective organizational collaborations. In such collaboration, each of the participants

would behave according to the predefined and mutually agreed upon business logic and Service Level Agreement (SLA) [16].

As stated above, any deviation from this agreed upon SLA is regarded as violations, and a robust mechanism is needed for its detection, logging and resolution [16]. The detection and prevention of failures under a composed service is complicated by the fact that composed services usually span several administrative domains, each of which will have its own interests and priorities. Building on the notions of trust presented, a trustworthy system is defined as a system that is already trusted, and continues to warrants that trust, because the systems behaviours can be validated in some convincing ways [17]. In such a system, the root of a failure or misbehaviour can always be identified and associated with responsible entity, and supported by non-disputable evidence [17].

The methodology stated above allows fault detection of predefined requirements through binding web addresses. The research in [18] has modified the research in [16] through incorporation of cloud users beside providers and making cloud users equally accountable for any malicious activities. The fault detection framework of [17] has been improved in [18] 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.

3.3 Journal-Based Provenance

Log based provenance is the process in which system logs and file operations are monitored and provenance data collected and stored from those. Ko et al identified accountability of cloud services as a complex challenge to achieve by cloud developers [19]. This is due to the large scale distributed virtual and physical server environments where the following attributes must be achieved.

1. Real time identification of source and destination of duplicate file locations

2. Logging of a file life-cycle

3. Logging of modification of file information and access history to a specific file.

A recent survey revealed that 88 percent of the customers are worried about security, confidentiality and availability of the data stored on the cloud, and would like to have more information about what activities take place on the cloud servers [19]. Such surveys have not only identified trust and accountability as key barriers to the widespread acceptance of cloud computing, but also enhanced the urgency of researchers to quickly address obstacles to trust in the cloud such as provenance detection from log files [19]. The paper focused on detective controls for tracing data and file movements in the cloud. Detective controls involve identification of file changes without taking any preventive measures. Preventive approach involves taking measures to prevent a specific activity.

The research also outlined that current prominent cloud service providers are not providing full transparency for the tracking of file access logs and modification history of both the physical and virtual machines which are utilized [20]. Currently users can monitor system performances of a virtual machine which they use. Methodologies which identify accountability and audibility of cloud service providers, such as log based provenance detection, have been outlined to a little extent. Such methodologies will help cloud service providers reduce the following threats as identified by the Cloud Security Alliance (CSA).

1. Abuse and nefarious use of cloud computing

2. Insecure application programming interfaces

3. Malicious insiders

4. Data loss and leakages

5. Unknown risk profile

Through the description of real life scenarios which included storing images on cloud servers, the real time tracing of operations executed on the cloud were identified. The scenario illustrates customer storing sensitive data on the cloud storage within a virtual machine instance which is provided by the cloud service provider. The fail-safe mechanisms of the cloud will ensure data backup and load-balancing will be ensured through keeping redundant storage of data in virtual machine servers and physical server within the trusted domain of the service provider. From storage creation to virtual machine instance launching, a large number of file transfer operations take place across virtual and physical servers and several memory read-write operations are involved. At the same time file-life-cycle logging, file-change logging was identified to be primary challenges [20].

Malicious attackers try to intercept the file transfer and memory operations and transmit files on the cloud to servers outside using email services. If all such actions for file transactions and memory operations are logged, it is possible to access the log history and the duplications and modifications made to the files can be traced. Regulations which hinder the blocking of such transfer were also identified. These regulations include certification of trusted cloud providers, cloud trust track record and legislation which involve government bodies imposing penalties for breaches of the cloud service agreement [21].

The research does not provide a provenance detection scheme that identified both file-centric and system-centric operation on the cloud and can be successfully used to identify and block such attacks.

Technical and procedural approaches to ensure privacy is a key goal for software engineers when building a cloud for the production environment [21]. The research

focus on lower levels of privacy through the use of privacy management tools for system level operations. Use of pseudonymisation tools to hide the name of real users is proposed. These technologies include anonymous login, pseudonymous identity number and email addresses. However the use of log-based provenance to aid in data-forensics is discussed to a little extent.

Privacy aware provenance of scientific workflows is stated in terms of data module and privacy policy [22], [23]. The 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. The research comprised of providing both composite and simple workflows of provenance data as acyclic graphs and then obtaining inference from those.

A workflow application was represented as a directed acyclic graph, with nodes denoting modules and edges denoting assignment of modules. Modules can be labeled with names, keywords and descriptions; the description may include the name and type of input/output data. Workflow specification may be hierarchical, in the sense that the module may be composite and itself contain a workflow. Composite modules were frequently used to simplify workflow design and allow component reuse. Workflows that do not have composite modules were defined as simple workflow in the research [24].

Privacy concern of the workflows was tied to the components: data, modules and the structure of the workflows. The research stated that in data privacy, private intermediate data, that is, data flowing between modules in a workflow execution, must not be revealed to the users. In module privacy, the researchers targeted to ensure the functionality of the module is hidden from the users in a strong sense, namely, for any input to the module, a user should not be able to guess with any

degree of certainty the output of the module [24].

In the case of structural privacy, the process through which a particular information was generated in the execution of the workflow are kept hidden from the user, and critical proportions of the provenance graph were not revealed to the user [24]. From the broader sense, the fundamental question addressed in the research is given below.

1. How to provide provable guarantees on the privacy of the components in a workflow while maximizing utility with respect to the provenance queries [25]- in doing so, the researchers considered the following.

   (a) How privacy was measured?

   (b) How utility was measured?

   (c) What provenance information should be hidden from the user?

The methodology of the research involved a mechanism to identify the structural privacy through keeping the data private of some module M which contributed to the generation of data item d, produced through a second module M‘. In the execution of a particular workflow Wn, which reads data from both a private and public database, the information that updates the private database, and to the module Wn, can be kept private and the private provenance information are kept hidden from the user.

The structural privacy in the case of hierarchical workflows was determined through clustering a number of public and private databases within the same sub-workflow which introduced new nodes to the execution of the final workflow. Therefore the information of the clustered private database was kept hidden from the user. The expansion hierarchy contained data retrieved from the public databases.

Technique for keyword search within the workflow was also identified which consisted of comma-separated search terms which can match the names, descriptions and

keywords associated with the modules. The search results are a list of workflows each of which represent a minimal view of some specification in the repository that contains distinct matches of all search terms. Search results catch the dataflow among search term matches using the definition of view [25].

The results reflect that privacy guarantees in the design of the proposed provenance scheme is maintained in the case of scientific workflows. The privacy concerns of provenance data, module and structural privacy were identified and the questions stated above were answered. However, necessity of user defined provenance information and ensuring management of the captured provenance have been discussed to a little extent. At the same time the research was concerned to a little extent about system and file-centric provenance data.

A framework to ensure trust and security of cloud environments with respect to large volume of virtualization and distributed storage management is proposed in [25]. In the framework both technical and policy level approaches have been taken into account. The methodology involves capturing provenance information in three different layers; namely system, data and workflow [25]. Through the capture of the above types of data, the following can be accountability properties must be achieved.

1. ***Logging:*** Non-disputable operation history must be logged in real-time. History logs should be sufficient enough for any later disputes with respect to the predefined correctness of business processes [25].

2. ***Monitoring and Auditing:*** By analysing the operation log, the system's state is continuously monitored to provide underlying services monitoring information to assist their operations. Once an exception or violation is detected or reported, the root cause should be discovered in a provable manner and actions taken to reduce its impact [25].

3. ***Dispute resolution:*** In special cases when the source of failure cannot be bound to a specific entity, procedures will be carried out to decide the violating entity in a best effort manner. The system shall suffer minimal delay due to dispute resolution [25].

Based on the properties of the above framework, a model has been proposed to capture provenance information in accordance to the requirements of Trustworthy Service Oriented System (TSOA). It was seen that services were divided into two domains, the Accountability Service Domain (ASD) and the Operational Service Domain (OSD). In OSD, operational services (OS) from multiple clouds interact with each other to form a service composition. Each service in an OSD keeps a close association with the accountability services (AS) in ASD, so as to ensure that the business process is indeed accountable [25].

The overhead of storing large amount of provenance information and data-process dependencies have been discussed to a little extent. The experiment had been carried out on proprietary cloud services only with little focus on open source technology. However both the research have focused very little on the important need of maintaining the captured provenance and devising an easy to understand methodology of provenance presentation. Also the benefits of ensuring accountability through provenance detection were focused to a little extent and no results of the proposed model were provided.

3.4 Provenance Attributes

Provenance attributes involve the pre-requisites that are necessary for successful provenance detection in a distributed and ubiquitous computing environment like the cloud [26]. Muniswammy Reddy et al identified key attributes for provenance

detection on the cloud [26]. The key attributes include provenance data querying which is the mechanism for disallowing clients to use the old data to analyze new provenance information and vice versa. Another attribute is the multi-object causal ordering property which acknowledges the causal relationship among objects and ensures that an objects ancestors are persistent before making the object persistent itself [26]. A notable attribute is the data-independent persistence property which ensures that a system retains an objects provenance, even if the object is removed.

These attributes are discussed below to a greater detail.

1. ***Provenance Data Coupling:*** The data-coupling property states that an object and its provenance must match, that is, the provenance must accurately and completely describe the data. This property allows users to make accurate decisions using provenance. Without data-coupling, a client might use old data based on new provenance or might use new data based on old provenance. In both of these cases, the user relying on the provenance is misled into using invalid data. Systems that do not provide data-coupling during writes can detect data-coupling violations on access and withhold or explicitly identify objects without accurate provenance [27].

2. ***Multi-object Causal Ordering:***This property acknowledges the causal relationship among objects. If an object, O is the result of transforming input data P, then the provenance of O is the super-set of the provenance of P. Thus, a system must ensure that an objects ancestors are persistent before making the object itself persistent. Multi-object Causal Ordering violations occur when the system writes an object to persistent store before writing all its ancestors, and the system crashes before recording those ancestors and their provenance [26].

3. ***Data-Independent Persistence:*** This property ensures that a system re-

tains an objects provenance, even if the object is removed. As in the last section, assume that P is an ancestor of O. If P were removed, Os provenance still includes the provenance of P, so a system must make sure to retain Ps provenance, even if P no longer exists [26].

4. ***Efficient Query:*** Since provenance is created more frequently than it is queried, efficient provenance recording is essential. However, efficient query is also important as provenance must be accessible to users who want to access or verify provenance properties of their data. In scenarios where the number of objects are few or users already know the objects whose provenance they want to access, efficiency is not an issue.

At the same time cloud properties like extensibility, availability and scalability were identified as key challenges to the detection of provenance. The challenges are explained in detail in the following.

1. ***Extensibility:*** Most existing provenance systems assume the ability to modify system components. For example, PASS uses either a file system or an NFS service as the storage backend. However, modifying or extending existing services is not possible due to system lock-in property of traditional infrastructure [26].

2. ***Availability:*** One can imagine building a wrapper service that acts as a front to the cloud services and provides a cloud provenance storage service that satisfies the properties that were identified. For the approach to be viable, however, the wrapper service has to match the availability of the cloud. If not, the overall availability is reduced to the availability of the wrapper service. Building such a highly available wrapper service is counterproductive as it requires a great deal

of effort and infrastructure investment, defeating the very purpose of moving to the cloud [26].

3. ***Scalability:*** In order to make the provenance queryable, most systems store provenance in a database. To avoid corrupting the database, clients need to synchronize updates between each other. A single global lock is a scalability bottleneck, and a distributed lock service would introduce the potential for distributed deadlock. Due to the update granularity of cloud stores, clients need to download the database object for every update, which also does not scale [26].

This research identifies benefits of provenance in addition to the above. At the same time the challenges of provenance are discussed to a little detail in this research. A scheme that keeps the provenance data confidential from general users in case of sensitive records should also be provided, which will ensure successful maintenance of metadata to aid data forensics.

3.5 Provenance Detection Obstacles

Provenance detection presents a number of challenges and obstacles for data forensic experts since the cloud is a distributed environment and there are relationships between multiple physical servers and between physical and virtual servers also [28]. Hence detection of provenance for such an environment is a fairly difficult task that must be achieved. Ambrust et al have provided figures to quantify comparisons between cloud and conventional computing and identified the top six technical and non-technical obstacles of cloud computing those are business continuity and service availability, data lock-in, data confidentiality and audibility, performance unpredictability,

scalable storage, reputation fate sharing [28]. The identified obstacles are discussed to a greater detail in the following.

1. ***Business Continuity and Service Availability:*** Technical issues of availability aside, a cloud provider could suffer outages for nontechnical reasons, including going out of business or being the target of regulatory action [28].

2. ***Data Lock-In:*** Concern about the difficult of extracting data from the cloud is preventing some organizations from adopting Cloud Computing. Customer lock-in may be attractive to Cloud Computing providers

3. ***Data Confidentiality and Audibility:*** In special cases when the source of failure cannot be bound to a specific entity, procedures will be carried out to decide the violating entity in a best effort manner. The system shall suffer minimal delay due to dispute resolution [25].The security issues involved in protecting clouds from outside threats are similar to those already facing large datacentres, except that responsibility is divided among potentially many parties, including the cloud user, the cloud vendor, and any third party vendors whose value-added services have been bundled into the cloud offering [28].

4. ***Performance Unpredictability:*** The obstacle to attracting HPC is not the use of clusters; most parallel computing today is done in large clusters using the message-passing interface MPI. The problem is that many HPC applications need to ensure that all the threads of a program are running simultaneously and todays virtual machines and operating systems do not provide a programmer-visible way to ensure the above statement [29].

5. ***Scalable Storage:*** The opportunity, which is still an open research problem, is to create a storage system that would not only meet these needs but combine

those with the cloud advantages of scaling arbitrarily up and down on-demand, as well as meeting programmer expectations in regard to resource management for scalability, data durability, and high availability [29].

6. ***Reputation Fate Sharing:*** One customers bad behaviour can affect the reputation of the cloud as a whole. For instance, blacklisting of EC2 IP addresses [29] by spam prevention services may limit which applications can be effectively hosted.

To address the above issues, a methodology was adopted and it was shown that negative behaviors of attackers posing as customers can cause black listing of EC2 IP-addresses through span prevention which ultimately limits the application services of the cloud provider. Trusted email was suggested as a solution to the above problem, which leads to microcosm of the issue. However the use of log-based provenance detection to track the attackers has been discussed to a very little extent.

3.6 Provenance based Identity Detection

Rule based provenance detection, tracing data provenance to ensure actual detection of identity has been discussed in [30]. In the methodology, rule correlation engine was used which implements provenance algorithms on various existing cloud software, and performance in terms of data leakage were analyzed. The important of user-defined provenance and visual representation of provenance were highlighted in the research to a little extent.

With respect to the above issue, provenance detection for distributed computing has been analyzed in [30]. The research stated that the provenance detection scheme proposed in the research identified provenance data using Axis 2C, which is a service provided by Apache Axis and Mule that logs the actions to reproduce the results of

the operations. Also the research provided standards that can be followed to detect and provenance data. However the methodology did not provide an infrastructure of cloud which can be used to detect provenance.

The importance of provenance was addressed and it was stated that provenance will play a central role in emerging advanced digital infrastructures. In the paper, the current state of provenance research and practice has been outlined, hard open research problems involving provenance semantics are identified, formal modelling, and security, and a vision for the future of provenance have been articulated [31]. The paper address the fact that technologies offer dramatic advantages, however they also exacerbate the hazards posed by buggy programs and dirty data. Digital information is easy to copy, change, and misinterpret. Current software systems do not provide the levels of repeatability, reliability, accountability and integrity achieved by the paper-based technology such as books, academic journals and laboratory notebooks that those are predicted to replace [31]. The solutions to the identified problems include provenance detection on the basis of semantics, traces, causality and identity, details of which are described below.

1. ***Semantics:*** Many forms of provenance could be captured for a particular system. At a bare minimum, where a particular piece of data comes from should be known [31]. While the above forms of provenance have already been introduced and studied in the literature for specific settings and languages such as relational databases, workflows, or file systems, there is a need for generally applicable, formal foundations for provenance. Moreover, since it can only be expected that provenance would become ubiquitously available if the effort of adding provenance support to systems is relatively low, an effective methodology that allows this general theory to be easily applied to new settings and languages should be developed [31].

2. ***Traces:*** Many forms of provenance are motivated by a desire to cache intermediate results and support efficient recompilation when the input changes. Incremental recompilation is a classical problem encountered in many different guises throughout computer science. Thus, when the input is changed, the effects could be propagated by replaying just a part of the trace. Such trace information ought to be computable from a sufficiently rich form of provenance [31].

3. ***Causality:*** These concepts are often invoked as motivations for provenance, but they are nontrivial. It is far from obvious how to make sense of informal claims that a given provenance record correctly captures a causal relationship, increases trust, or justifies a knowledge or belief, and most research papers dont even try. However, recent work on mathematical models of causality, trust, and knowledge may provide a good starting point for answering these questions [31].

3.7 Issues in Provenance Research

The research is based on identifying provenance solutions for effective provenance detection and inference. However, the research focused to a little extent on the importance of providing a model on the basis of the solutions. If data forensics is to be successful, effective provenance detection models are required to identify disputed data in the cloud. Work on detecting provenance data for investigation of malicious activities on stored files and databases have been conducted to a limited extent [32]. Secure provenance which implies tracking and recording of ownership and process history of data, are essential for data forensics [32]. A scheme that keeps the provenance data confidential from general users in case of sensitive records should be provided, which will ensure successful maintenance of metadata to aid data forensics.

Provenance is an unexplored area in cloud computing, where a number of security challenges exist. The provenance data should be stored in a way so that original provenance information should not be forged by malicious users. The model of provenance design should be such that the provenance data can be revealed only by a trusted authority, and no one else. Provenance data may be platform specific or may not be reliant on a given platform. Developing provenance schemes to ensure interoperability between different types of data is an area of keen research interest. If platform independency and interoperability of metadata is achieved, the migration of provenance data from one system to another can be achieved [32].

Obtaining and representing provenance metadata without violating security or privacy of the users is an active research issue. In addition to these attributes, developing a model that provides unforgetability and interoperability is a highly sought-after architecture. A design with the mentioned capabilities will achieve the desired attribute of provenance data sustenance even when the artifacts are removed.

Although extended studies have been carried out in provenance, the main challenges and requirements to implement cloud provenance is an active area of research. Considering the dynamic and complex nature of cloud computing, a model that would link log and audit data collected from multiple infrastructures is a challenging task [32]. These challenges should be identified to a greater detail in order to ensure that they can be mitigated for reliability of cloud computing.

Cloud provenance collection must be real time as the data is accessed and manipulated [32]. Time tagged provenance detection is an issue, however ensuring time synchronization between the different physical machines like cloud and node controllers is another challenge.

Provision of a model for an API for collection of provenance data from the users and making that data available on the web for easy access is also a vital research issue.

The most recent research objective is to incorporate the users with the obtainment of provenance or attribution data. This will also enable the system administrator to check whether the user comply with the provided information. Hence this model is highly demanded.

A well designed user APU-GUI is needed to permit user annotation and application specific provenance. As a result, challenges of automated provenance collection can be mitigated and at the same time user involvement and increased semantic knowledge on provenance data can be ensured [33]. If the data was generated from non-provenanced sources, a good solution for incomplete provenance data is to allow signed user input, with the help of a well designed API. Assurance that provenance data is not forged by an adversary and mechanism to detect and prohibit suspicious user annotation is still an open research question [34]. The provenance model should be such that an administrator should be able to monitor provenance data. The model should also ensure that provenance data is encrypted and should have access control policies like the one described above.

A comprehensive provenance framework is essential for researchers to verify quality of data, reproduce scientific results in peer-reviewed literature and validate scientific processes [9]. This framework is required to ensure the pre-publication and post-publication states of data in translational research activities.

Provenance data without incorporation to real life web data is of little use. Approaches need to be defined for incorporating provenance data to the data on the World Wide Web. This is needed for quality assessment of data on the web since web data undergoes a large amount of replication, query processing, modification and merging [9].

The provenance data that are not very sensitive and needed by the users to verify the trustworthiness of data on the web should be made available to the users. Also,

the provenance metadata should be linked to the data which it describes. To make the provenance data part of the web data, both must adhere to the same publication principles [9]. A method that automatically synchronizes provenance data with actual data should be provided. The goal of this framework is to ensure the quality and trustworthiness of provenance data.

In case of cloud computing, lack of physical access presents a new challenge for provenance collection. Due to decentralized nature of cloud computing, traditional methods of collection metadata about data is no longer a practical approach [9]. Research needs to be done whether it is possible for cloud customers to carry out provenance detection on their data which is stored on the cloud from a technical standpoint. Methodologies to perform such investigations needs to be provided as it will give more control to the customers of cloud computing.

Similarity and overlapping between trusted computing and cloud computing provenance should be identified [35], [36]. Architectural idea for a trusted provenance system and the requirements for effective integration of trusted computing systems and provenance systems should be identified for effective provenance detection.

Mechanisms to develop provenance storage in the cloud are an area of significant research interest. Cloud storage is affordable and scalable. So if an effective system for storing provenance data on cloud storage can be engineered, the issue of data storage cost and continuous growth in volume of attribution data can be mitigated.

The methodology of using cloud as a standalone system to store provenance data is also an important issue of research. The issue is justified since it tries to propose a model of storing provenance data using a standalone cloud storage system, thus providing a new way of database-cloud storage co-ordination.

The target of the application is to identify the granularity for provenance data, and group those on the basis of the granularity. Despite the extensive research interest

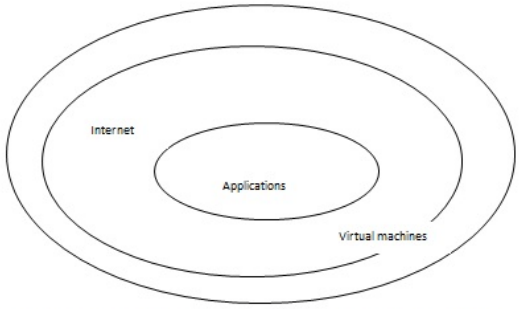


Figure 2.1: Category of provenance data on the basis of granularity

Figure 3.1: Category of provenance data on the basis of granularity

shown in the field of cloud computing provenance, a number of drawbacks exist. The provenance research works have only focused on database systems, operating systems and automated provenance detection [37]. Research on how cloud can replace existing methods like databases systems to store provenance data has been carried out to a limited extent.

Community standards such as the Open Provenance Model allow uniform interpretation and exchange of metadata, but do not prescribe query specifications to retrieve provenance [37]. A system may perform a pipeline of tasks that will execute in a specific environment due to platform dependency; however, little work has been done to devise a methodology for storing this data in a single repository and in real time as the pipeline of tasks is carried out [37]. There is scope for research in the mapping between the physical machines and virtual machines in cloud computing. Since in cloud computing, most of the applications are run on the virtual machines, the provenance data must be recorded even when the virtual machines are terminated. However, very little research has been done in this regard. Data reuse and transfer across multiple platforms must be accompanied by provenance data, otherwise tracking this data and ensuring the trustworthiness of data will become debatable. The research papers under consideration consider this requirement to a very little extent.

Chapter 4

ProvIntSec: Provenance Cognition Blueprint

As discussed in the previous chapters, a summary of the cloud provenance based trust issues can be drawn on the ground of the limited work on provenance detection for a highly complex and distributed environment such as open source clouds like OpenStack [38]. The chief reason for this is the dynamic nature of cloud environments. The lack of trust on the cloud service providers from the customers perspective poses a big risk to the success of the cloud [38]. The principal reasons behind this are the limited availability of journal based frameworks for file-centric and system-centric provenance tracking for open source cloud.

4.1 Proposed model for Provenance Cognition

Lack of accountability from the perspective of the cloud provider is another notable reason for some customers reluctance of moving to the cloud platform [39]. Hence the activities of cloud administrators and maintenance engineers need to be monitored through effective provenance detection schemes. Provenance storage for future retrieval is another issue regarding the performance hindrance for critical processes caused due to storage of high volume of provenance information in data banks. The issue must be addressed as to whether a proposed journal based provenance framework adheres to the requirements of the performance standards such as allocated bandwidth utilization and percentage of excess bandwidth consumption.

Many cloud providers are using open source cloud infrastructures like OpenStack for provision of vm instances to the customers. The reliability and accountability of such open source cloud must be ensured. OpenStack cloud is an open source cloud

environment that has gained considerable amount of popularity among the business world [39]. An extension to Openstack is needed that would include journal based provenance detection to aid cloud forensic experts and audit trials. As a result, such an extension will certainly increase the acceptability of OpenStack among the business community.

As mentioned above, once the journal based provenance detection framework is available for OpenStack, an organizational business model needs to be proposed that would identify the core members of the cloud team within a company who are involved or are affected by the provenance detection scheme [40]. This is necessary as it will help cloud providers identify the players involved in the implementation, operation and maintenance of the cloud service and also define the auditability and accountability relationships with other cloud providers and aid in the determination of Service Level Agreement (SLA) with customers as well.

A mathematical model with effective performance analysis results must be presented to identify the process trees in the cloud and evaluate modifications to system critical processes for better detection of forgery and malicious activities in open source cloud environments. Such evaluations have been done to a minimal extent for open source cloud, the reason being the organizational dependencies on proprietary cloud services [41]. However with the shift in the interest of business organizations towards open source cloud platforms [41], such models with analysis has become a must need. According to Cloud Security Alliance, top of the 4 threats out of 7 can be solved using journal based provenance detection [42]. Hence the framework can contribute to the mitigation of majority of threats in open source cloud environments.

4.2 Provenance Blueprint

As discussed in the previous section, the need for journal based provenance detection scheme is manifold. Hence a framework for provenance detection to monitor critical activities executed in open source cloud platform such as OpenStack is proposed in this research. The framework extends the existing OpenStack Essex cloud platform and ensures effective monitoring of critical processes running in OpenStack cloud.

Mathematical models showing how process trees are parsed to identify relocation of files have been proposed in this thesis. At the same time mathematical metrics are used to evaluate the performance of the proposed framework from multiple perspectives such as empirical performance analysis, network bandwidth usage and rate of malicious activities detected by the framework [43].

4.3 Description of the proposed blueprint

Figure 4.9 shows the events in which a process P1 monitors the journal events in Virtual Machine Environment 1 (VME1) and passes the collected data to another process P2. 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 VME1 is located at a specific physical machine, if the framework is implemented in the same machine, the transmission of packets is still required since VME1 is completely separate virtual entity. Process P7 is a process in the physical machines which compounds the free packets to obtain the buffer which is in turn unpacked using the header and footer information attached with those to obtain the data.

It is clear from the above information that the data which is packed and sent

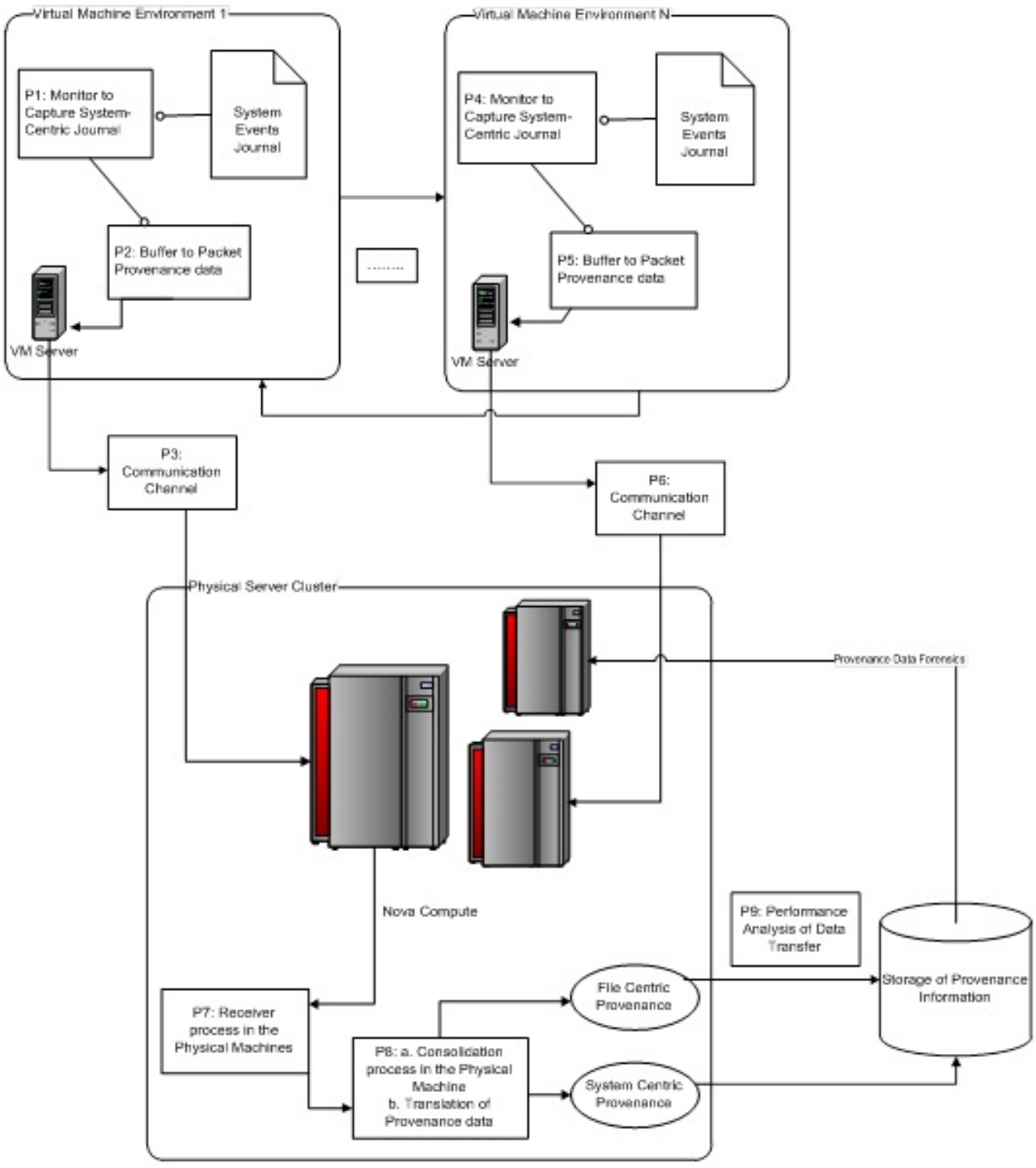


Figure 4.1: Model showing journal based provenance detection in vm-pm mapping on the cloud

from the physical machines are received at the receiver in intact format without and modifications. Process P8 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 critical process journal files are placed in the system centric provenance objects. After consolidation is completed, the information is analyzed using mathematical models and then translated through methods shown in the later sections.

As stated above, the translated data can be used by data forensic experts and cloud administrators for their specific purposes like the data forensic experts can analyze the user centric and system centric journals to ensure that a specific process was triggered by a certain user at a point in time, on the other hand cloud administrators can use the information to decide whether a virtual machine instances was launched successfully, and if not, what is the reason behind the failure. In order to ensure future retrieval of this useful provenance, it is critical that the provenance be stored in a data bank. The network consumption/usage/overhead caused when the provenance data are transferred to the cloud data storage server is an important issue of research [44].

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, both physical and virtual. Figure 4.10 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. When the file moves from one process to another within the same process tree, the system calls made on the file can be identified and evaluated for each process to identify whether a file has been copied

from the cloud environment to another physical machine outside the cloud. System critical files can also be renamed within the same process. If this occurs within the same machine, hence more than one process may be involved in it. The algorithm shows what may happen in that situation. 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. For OpenStack such files include the nova-compute.conf, nova-volume.conf, nova-scheduler.conf, nova-network.conf files.

Algorithm 1 and Algorithm 2 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. Since the experiments have been conducted in OpenStack Essex which is setup on Linux server, so the system calls for Linux file transfers have been considered since the master machines are always in Linux. The file actions which are critical to system administration and data forensics have been identified in chapter 2 on the basis of [45].

Algorithm 1: ProvIntSec Provenance detection at vm-instance

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
```

Algorithm 2: ProvIntSec Provenance detection of data movement at the receiving vm-instance

```
Data: at Receiver 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 [PIDy-1, PIDy] do
4          if Pfirst+1 ran the connect-from-INTERNET command and
           OpenFile command executed using scp or mv then
5              file received from remote 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
```

At sender all the processes from P1 to Pn within the same process tree are parsed. For each consecutive process pairs, if the CONNECT-TO-INTERNET system function is called and then the openFile function is called, then the algorithm returns that a file has been copied from one machine to another. At receiver, the processes in the same process tree are parsed and for any two consecutive process pairs, if CONNECT-FROM-INTERNET is called and openFile is then the system returns that the file has been received from a remote machine.

4.4 Extension of OpenStack model

The implementation and performance analysis of the proposed provenance detection model for system administration and data forensics have been implemented and tested on OpenStack Essex cloud running in Ubuntu 12.04 servers. Hence the OpenStack model has been extensively analyzed and a proposition to integrate the proposed provenance framework has been presented to increase the security of OpenStack and increase its acceptability to the business world.

The extension of the OpenStack model will work in line with a number of modules of the existing OpenStack structure which includes collecting journaling information from Openstack scheduler process, Image register and upload process, nova-compute and nova-network processes respectively. Information regarding file copy, rename or volume processes are also collected by the proposed framework. This will enable OpenStack cloud administrators to keep track of the cloud and detect malicious activities executed in the cloud computing infrastructure.

Figure 4.11 shows an overview of the extended OpenStack model and shows how the extended part of the framework gathers information from the existing cloud modules without hindering the activities of those. The following is a description of the processes with which the proposed framework communicates and collects journal events.

1. ***Nova-schedule:*** The nova-scheduler in OpenStack is used to schedule the events that take place in the cloud such as the event of launching a vm instance [45]. The framework collects information about the schedule and feeds the information to the cloud administrator thereby assisting the administrator in the decision making process that a vm-instances has been launched successfully.

2. ***Image upload and register:*** The process determines whether an image of a operating system has been uploaded and once uploaded, whether the image

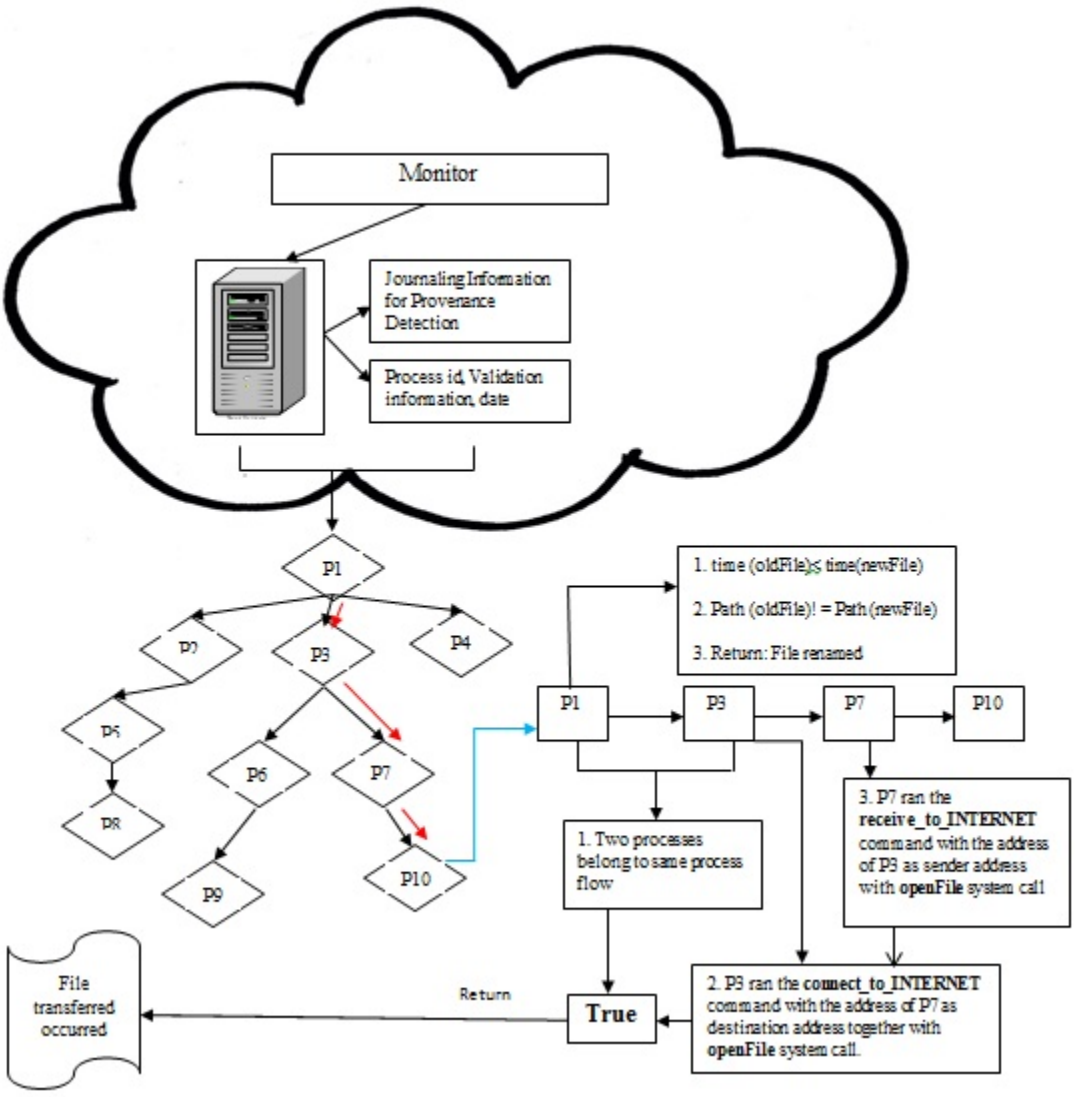


Figure 4.2: Detailed functionality of the provenance framework that detects critical system file movements across multiple machines

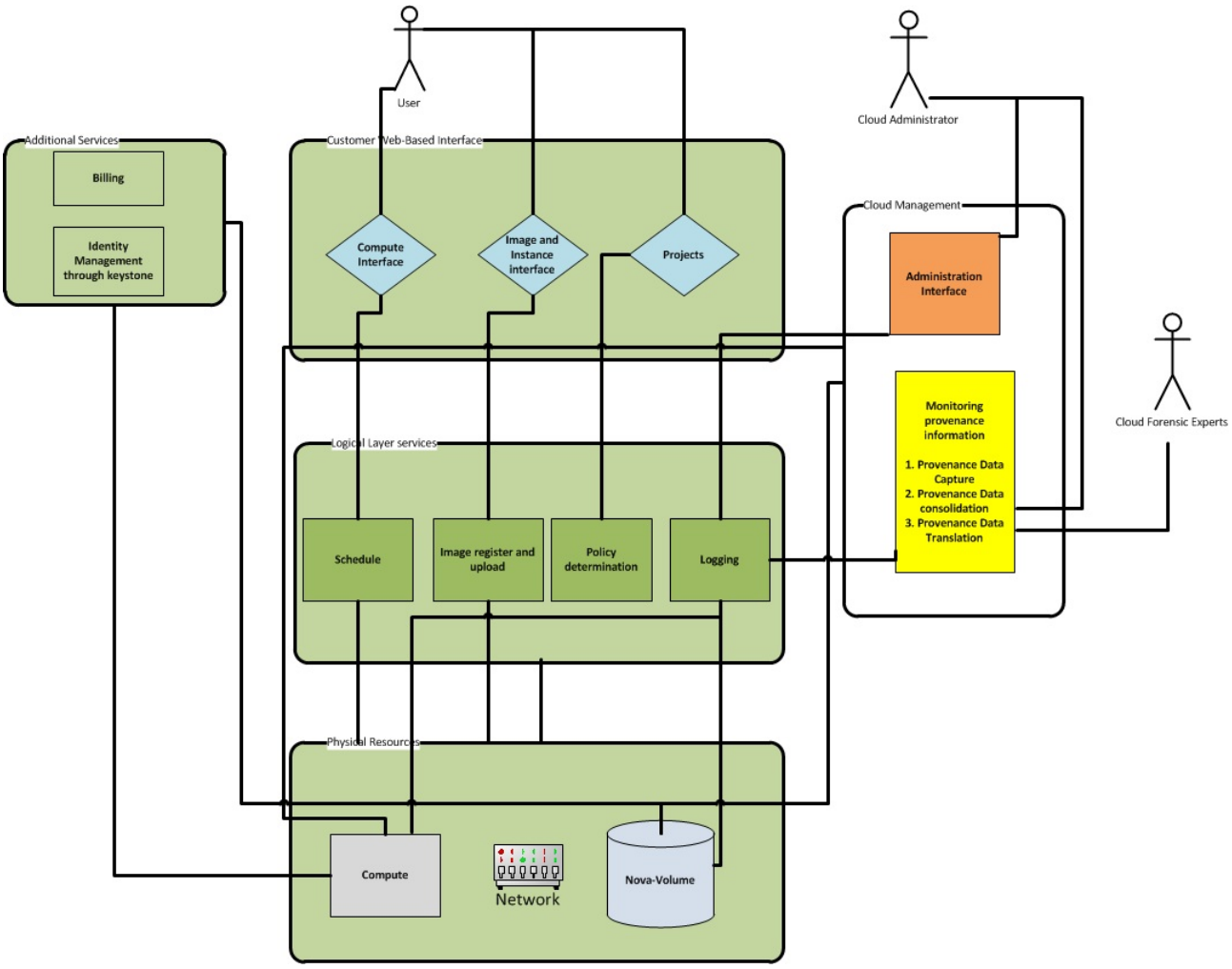


Figure 4.3: Extension of OpenStack Model

has been registered and a unique registration id has been provided that will enable the users to launch instances of that image [46]. The framework collects information from the process and enables the cloud administrator to identify the time, date, version and permissions of the uploaded and registered image.

3. ***Nova-compute:*** The journals of nova-compute are monitored by the proposed framework to provide information about critical proceedings on the cloud such as allocation of virtual network, the resource allocation from physical machines.

4.5 Solution of the addressed problems by the proposed model

The problems stated in earlier section of this chapter can be solved through the provenance detection scheme proposed in this research. The highlighted problems with their solutions are stated below.

1. ***Limited work on Provenance collection for open source cloud:*** It was stated in the related work chapter that research on journal-based provenance detection for open source cloud has been conducted to a limited extent [47], [48], [49]. The chief reason for this being that most research are funded by proprietary cloud service providers and hence the testbed are the proprietary cloud platforms. This research puts forward a journal based provenance detection model for OpenStack which is open source. Therefore the research contributes in the provenance detection field for open source cloud computing therefore helping open source cloud administrators and forensic experts better manage the open source cloud platforms and collect data for investigation.

2. ***Lack of accountability:*** The provenance detection model proposed in this research will work to increase accountability of the cloud administrators. Cloud forensic experts can use the provenance information to identify what important files have been removed from the cloud environment or renamed or even copied from the cloud master machine.

3. ***Lack of trust by cloud customers:*** Potential customers of open source cloud services do not get trust on the service mainly because there is no way that they can track what is done with their data which they have stored in remote open source cloud servers. A provenance detection model for open source cloud

infrastructure will certainly contribute to increasing the trust and reliability of the corporate houses to host their important data on open source cloud platform because then the open source service will not only be cost effective but also reliable.

4.6 Difference with existing research

Most research on provenance detection on cloud has been done with scientific workflows that are executed using the computation power of cloud [50], [51]. This research is different from that perspective that the provenance detection scheme is for the cloud itself rather than any service which runs on the cloud. The provenance research related to management of the cloud has been done mostly for proprietary cloud service providers such as Azure and AWS mainly because the research was financed by those companies. The research in [52] provides a security scheme for open source Eucalyptus cloud which is now being provided as a proprietary service from version Eucalyptus 3.0. However, this research was conducted specifically for OpenStack which is completely free and open source and on which very little research has been done. Hence this research is concerned with a highly promising open source cloud platform on which limited research has been carried out.

In addition to the above, the research has also put forward a business model that identifies the people who will be involved in the cloud team for maintenance and use of the framework once the model is implemented at the business level shown in Figure 4.12. The model also shows the processes involved in inter-organization communication between different cloud teams in the hybrid framework. The model also identifies which data needs to be tracked during such hybrid communication. The business model also identifies the members of the cloud team who will be in-

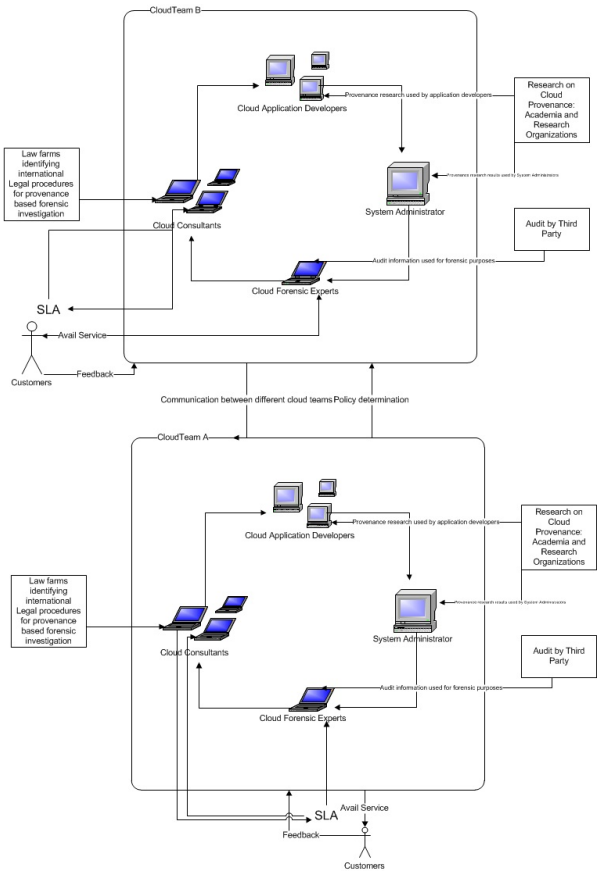


Figure 4.4: Business model to incorporate the proposed framework into business organizations

volved during the meetings with the customers in the determination of the Service Level Agreement (SLA) regarding the cloud service which incorporates the proposed accountability framework.

Chapter 5

Implementation and Result Analysis

The chapter provides the platform and experimental environment to execute complex research experiments and analyze the performance of those. The testbed principally allows implementation of the research and evaluation of the experimental results.

The testbed for the research under consideration involves physical servers of considerable capacity. The cloud environment setup in those consists of OpenStack cloud service which is open source. The latest version of OpenStack Essex release has been used in the latest Ubuntu 12.04 Long Term Service (LTS) servers for the experimental excellence. The cloud controller or compute service is the main process that controls launching and termination of virtual machine instances in OpenStack.

OpenStack Compute which is also called the cloud controller consists of several main components. A "cloud controller" contains many of these components, and it represents the global state and interacts with all other components. An API Server acts as the web services front end for the cloud controller [53]. The compute controller provides compute server resources and typically contains the compute service. The object store component optionally provides storage services. A volume controller provides fast and permanent block-level storage for the compute servers. A network controller provides virtual networks to enable compute servers to interact with each other and with the public network. A scheduler selects the most suitable compute controller to host an instance. The remaining sections of this chapter provide description of the above components to a greater detail.

5.1 Cloud Environment: OpenStack

OpenStack is a collection of open source technology that provides massively scalable open source cloud computing software. Currently OpenStack develops two related projects: OpenStack Compute, which offers computing power through virtual machine and network management, and OpenStack Object Storage which is software for redundant, scalable object storage capacity. Closely related to the OpenStack Compute project is the Image Service project, named Glance. OpenStack can be used by corporations, service providers, VARS, SMBs, researchers, and global data centers looking to deploy large-scale cloud deployments for private or public clouds [53].OpenStack offers open source software to build public and private clouds. OpenStack is a community and a project as well as open source software to help organizations run clouds for virtual computing or storage [54].

OpenStack Compute is built on a shared-nothing, messaging-based architecture. You can run all of the major components on multiple servers including a compute controller, volume controller, network controller, and object store or image service. A cloud controller communicates with the internal object store via HTTP (Hyper Text Transfer Protocol), but it communicates with a scheduler, network controller, and volume controller via AMQP (Advanced Message Queue Protocol) which is also called RabbitMQ server. To avoid blocking each component while waiting for a response, OpenStack Compute uses asynchronous calls, with a call back that gets triggered when a response is received. To achieve the shared-nothing property with multiple copies of the same component, OpenStack Compute keeps all the cloud system state in a database.

There are currently three main components of OpenStack: Compute, Object Storage, and Image Service. OpenStack Compute is a cloud fabric controller, used to start up virtual instances for either a user or a group. It is also used to configure networking

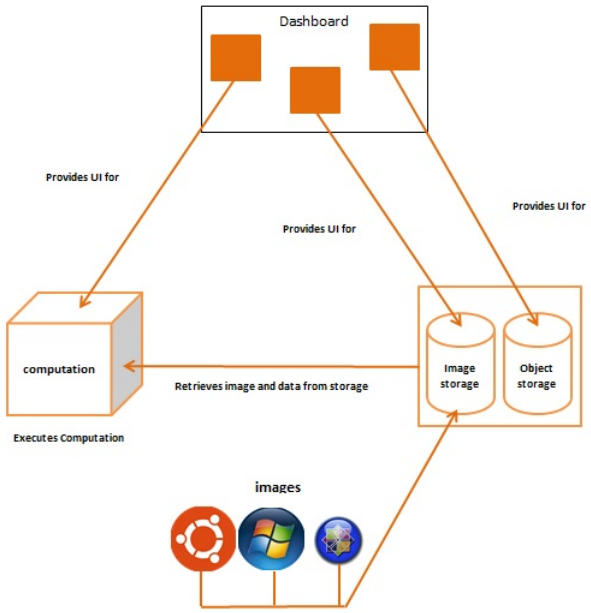


Figure 5.1: Communication between the various components of the cloud infrastructure used for the research

for each instance or project that contains multiple instances for a particular project.

OpenStack Object Storage is a system to store objects in a massively scalable large capacity system with built-in redundancy and failover. Object Storage has a variety of applications, such as backing up or archiving data, serving graphics or videos, storing secondary or tertiary static data, developing new applications with data storage integration, storing data when predicting storage capacity is difficult, and creating the elasticity and flexibility of cloud-based storage for your web applications [54].

OpenStack Image Service is a lookup and retrieval system for virtual machine images. It can be configured in three ways: using OpenStack Object Store to store images; using Amazon's Simple Storage Solution (S3) storage directly; or using S3 storage with Object Store as the intermediate for S3 access. Figure 4.1 below shows the communication between the principal components of OpenStack.

As seen in the above diagram, the services of OpenStack cloud can be divided

into two major components: Computation which involves providing virtual machines instances to the users in which computational operations are executed and results are obtained, and Storage which provides repository for cloud images and also customer data. The two services are described in the following to a greater detail.

5.2 OpenStack Compute

As stated above, the nova-compute process is primarily a worker daemon that creates and terminates virtual machine instances via Application Interfaces (APIs) of the hypervisor such as XenAPI for XenServer/XCP, libvirt for KVM or QEMU, and VMwareAPI for VMware. The nova-compute consists of a number of other processes like nova-volume, nova-network, nova-schedule, and a Database Management System (DBMS). The process by which this is done is fairly complex and the basics are described below.

1. The compute aims to accept actions from the queue and then perform a series of system commands like launching a Kernel Virtual Machine (KVM) instance while updating state in the database.

2. The nova-volume process manages the creation, attaching and detaching of persistent volumes to compute instance which is similar to the functionality of Amazons Elastic Block Storage (EBS).

3. The nova-network worker daemon is very similar to nova-compute and nova-volume with primary responsibility of accepting networking tasks from the queue and then performing those to manipulate the network.

4. The nova-schedule process takes a virtual machine instance request from the queue and determines where it should run, more specifically, in which compute

server host it should run on.

The SQL database stores most of the build-time and run-time state for a cloud infrastructure. This includes the instance types that are available for use, instances in use, networks available and projects. OpenStack cloud infrastructure can support any database supported by SQL-Alchemy but the only databases currently being widely used are sqlite3 (only appropriate for test and development work), MySQL and PostgreSQL. For the research on provenance detection, MySQL database is used.

Figure 4.2 below shows the important databases that have been made for the research purpose. The highlighted databases are namely keystone, glance and nova which are described in the following. As stated above, Nova cloud controller or compute process interacts with all of the usual critical cloud modules such as the following.

1. Keystone for authentication
2. Glance for image storage and retrieval
3. Horizon for web interface
4. The API process can upload and query Glance while nova-compute will download images for use in launching instances.

5.3 Storage

The storage infrastructure of the cloud used for the experiment is called swift architecture which is distributed in nature with an aim to prevent any single point of failure as well as to scale horizontally as described below.

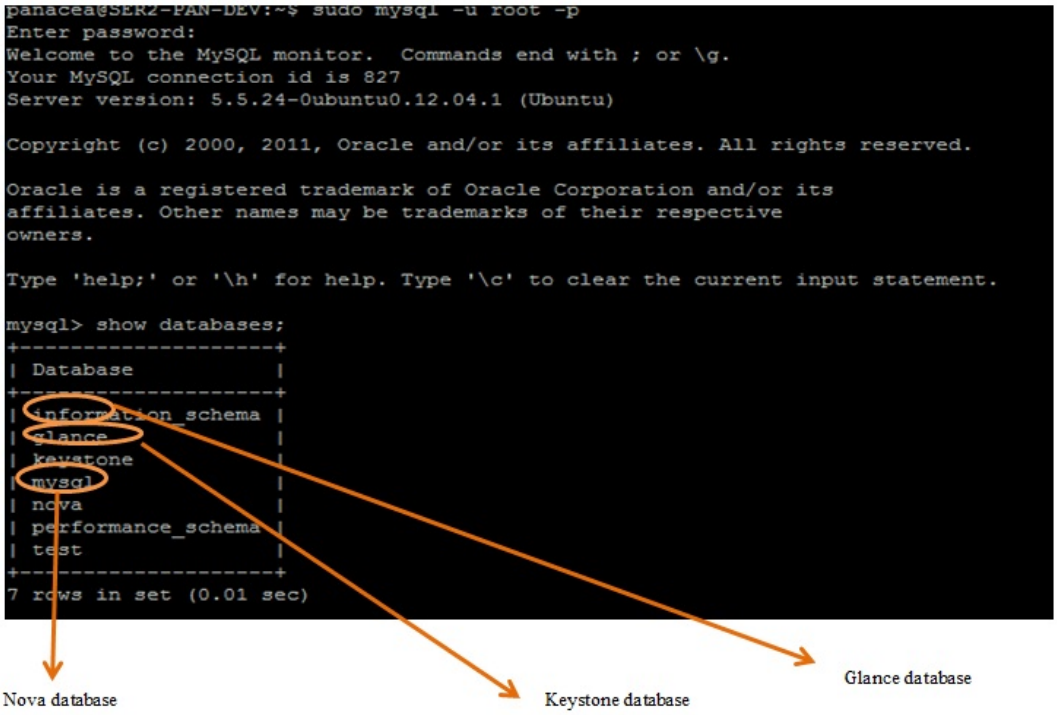


Figure 5.2: Important databases for the research

Proxy server accepts incoming requests via the OpenStack Object API or just raw HTTP. The proxy server may utilize an optional cache which is usually deployed with memcache to improve performance.

The swift service accepts files to upload, modifications to metadata or container creation. In addition, it will also serve files or container listing to web browsers. The swift service is responsible for maintenance of the account management, container management and object management which are responsible for authentication and folder maintenance on the cloud.

With respect to the services mentioned above, account servers manage accounts defined with the object storage service. Container servers manage a mapping of containers within the object store service. Object servers manage actual objects on the storage nodes. There are also a number of periodic processes which run to perform housekeeping tasks on the large data store. The most important of those is

Table 5.1: Hardware configuration requirement for the research

Server Processes	**Hardware**
Cloud Controller node which runs the following: **Network** **Volume** **API**: Provides user interface for enhanced cloud maintenance **Scheduler**: Determines job times and assignment of tasks to processes **Image service**	Following hardware requirement are necessary: **Processor**: 64-bit x86 **Memory**: 16 GB RAM **Disk space**: 1900 GB (SATA or SAS or SSD) **Volume storage**: 100 GB (SATA) for volumes attached to the compute nodes. **Network**: one 1 GB Network Interface Card (NIC) minimum

the replication services, which ensures consistency and availability through the cluster. Other periodic processes include auditors, updaters and reapers of the database.

The hardware and software environment of the research is important to this respect. The following sections provides a description for those.

5.4 Hardware and Software requirements

The research environment requires a number of hardware and software specifications, the details of which can be described as follows.

1. ***Hardware Specification:*** The hardware specification is described in the Table 4.1.

2. ***Software Configuration in the case of Post Installation:*** Configuring the compute installation involves many configuration files namely the nova.conf file, the api-paste.ini file, and related Image and Identity management configuration files. This section contains the basics for a simple multi-node installation, but Compute can be configured many ways. Networking options and hypervisor

Figure 5.3: Nova.conf file of the target research

```
nova-manage db sync
```

Figure 5.4: Database schema synchronization

options can be described.

3. ***Setting Configuration Options in the nova.conf file:*** The configuration file nova.conf is installed in /etc/nova by default. A default set of options are already configured in nova.conf when installed manually. Figure 4.3 identifies the configuration file components.

4. ***Setting up OpenStack Compute Environment on the Compute Node:*** Figure 4.4 is the command that was used to ensure the database schema adheres to all the current updates.

5. ***Software Configuration in the case of Post Installation:*** Configuring the compute installation involves many configuration files namely the nova.conf file, the api-paste.ini file, and related Image and Identity management configu-

```
sudo nova-manage network create private --fixed_range_v4=192.168.4.32/27 --num_networks=1 --bridge=br100 --bridge_interface=eth1 --network_size=32
```

Figure 5.5: Database population with network information

ration files. This section contains the basics for a simple multi-node installation, but Compute can be configured many ways. Networking options and hypervisor options can be described.

The requirement exists to populate the database with the network configuration information that Compute obtains from the nova.conf file. Following examples shows how it looks like when real values are entered as shown in Figure 4.5.

For this example, the number of IPs is /27 since that falls inside the /16 range that was set in fixed range in nova.conf. Currently, there can only be one network, and this set up would use the max IPs available in a /27. You can choose values that let you use any valid amount that you would like. When the nova-manage network create command was run, entries were made in the networks and fixed ip tables. However, one of the networks listed in the networks table needs to be marked as bridge in order for the code to know that a bridge exists. The network in the Nova networks table is marked as bridged automatically for Flat Manager.

5.5 Configuring Hypervisors

OpenStack Compute requires a hypervisor and supports several hypervisors and virtualization standards. Configuring and running OpenStack Compute to use a particular hypervisor takes several installation and configuration steps. The libvirt type configuration option indicates which hypervisor will be used. The authentication at the note of each hypervisor needs to be identified for this purpose.

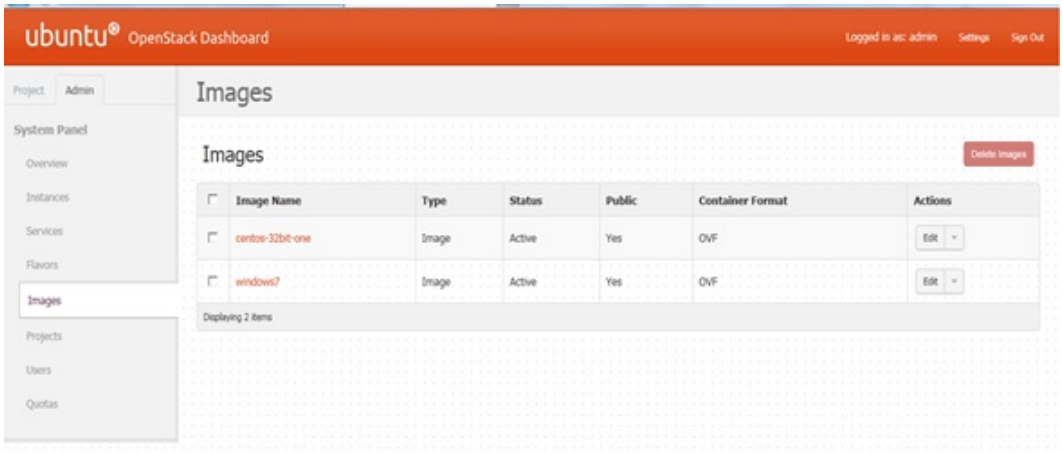


Figure 5.6: Image dashboards for the research

There are different methods of authentication for the OpenStack Compute project, including no authentication, keystone, or deprecated which uses nova-manage commands to create users [54]. With additional configuration, the OpenStack Identity Service, code-named Keystone was used for the target research.

5.6 Configuring Image Service and Storage for Compute

Compute of the target research relies on an external image service to store virtual machine images and maintain a catalog of available images. Compute is configured by default to use the OpenStack Image service, which is the only currently supported image service. A pictorial overview of the dashboard of image service is proved in Figure 4.6.

5.7 Understanding the Compute Service Architecture

The following are some basic categories that describe the service architecture and activities within the cloud controller prepared for the test environment.

1. ***API Server:*** At the heart of the cloud framework is an API Server. That is API Server takes command and control of the hypervisor, storage, and networking programmatically available to users in realization of the definition of cloud computing. The API endpoints are basic http web services which handle authentication, authorization, and basic command and control functions using various API interfaces under the Amazon, Rackspace, and related models. This enables API compatibility with multiple existing tool sets created for interaction with offerings from other vendors. This broad compatibility prevents vendor lock-in hence it was used for the target research.

2. ***Message Queue:*** A messaging queue brokers the interaction between compute nodes that are responsible for processing, volumes which are responsible for the block storage, the networking controllers which is the software that controls network infrastructure, API endpoints, the scheduler that determines which physical hardware to allocate to a virtual resource, and similar components. Communication to and from the cloud controller is by HTTP requests through multiple API endpoints. A typical message passing event begins with the API server receiving a request from a user. The API server authenticates the user and ensures that the user is permitted to issue the subject command. Availability of objects implicated in the request is evaluated and, if available, the request is routed to the queuing engine for the relevant workers. Workers continually listen to the queue based on their role, and occasionally their type

hostname. When such listening produces a work request, the worker takes assignment of the task and begins its execution. Upon completion, a response is dispatched to the queue which is received by the API server and relayed to the originating user. Next the database entries are queried, added, or removed as necessary throughout the process.

3. ***Compute Worker:*** Compute workers manage computing instances on host machines. Through the API, commands are dispatched to compute workers to execute the following actions in the research environment.

   (a) Running instances

   (b) Terminating instances

   (c) Rebooting instances

   (d) Attaching volumes

   (e) Detaching volumes

   (f) Get console output

5.8 Network Controller

The Network Controller manages the networking resources on host machines. The API server dispatches commands through the message queue, which are subsequently processed by Network Controllers. Specific operations include the following for the research environment.

1. Allocate fixed IP addresses

2. Configuring VLANs for projects

3. Configuring networks for compute nodes

Figure 5.7: Figure showing the launched instances in the research testbed

5.9 Volume Workers

Volume Workers interact with iSCSI storage to manage LVM-based instance volumes. Specific functions implemented to set up the testbed for the research include

1. Creating volumes
2. Deleting volumes
3. Establishing Compute volumes

Volumes may easily be transferred between instances, but may be attached to only a single instance at a time. Figure 4.7 shows multiple instances of different operating systems launched within the same physical node of the cloud infrastructure testbed prepared for the experiment. Similarly Figure 4.8 shows the access to the vm instance through the web based dashboard of the test environment. The launching of instances shows that the cloud test environment is ready for the research. The target now is to analyze the research environment and identify various schemes of provenance detection. Next the provenance detection scheme must be modeled and

Figure 5.8: Figure showing the launched instance accessed through the web

implemented. The modeled schemed should be evaluated for performance and results should be documented in the following chapters.

ProvIntSec satisfies the requirements of All Provenance Scheme (APS) as illustrated in [55] which requires the model to detect both socket provenance and malicious activity provenance. ProvIntSec is a provenance detection model, the implementation details of which have been discussed in the previous chapter. With regard to the process tree based provenance generation, the scheme is implemented and the provenance records are analyzed for file transfers and malicious activities in Linux cloud environment. Files of varying range of sizes are transferred across multiple virtual machine (vm) instances in the cloud where ProvIntSec is installed in each vm instance. Next provenance journals from ProvIntSec are analyzed to find clues for identification of the file operations based on multiple process trees across numerous vm instances [56]. Afterwards the precision or correctness of ProvIntSecs detection capability is analyzed and presented graphically [57].

Precision of ProvIntSec for detection of malicious activities in the cloud was also identified and performance evaluated for a list of malwares prevalent for Linux. The

malicious codes were executed in the cloud vm-instances and the journals of ProvIntSec are used to identify the presence of the malicious entities in the cloud. Next precision of ProvIntSecs performance was analyzed and the performance compared with the baseline to identify the overhead [57]. The results of the compiled experiments are presented using analytics and graphs.

It is seen that ProvIntSec is capable of detecting provenance for both socket based file operations in the cloud and also for the detection of malicious activities. Hence ProvIntSec satisfies the requirements of All Provenance Scheme (APS) as illustrated in [58]. The following sections provide ProvIntSecs accuracy in dealing with both malware and socket level provenance to a greater detail.

5.10 Metrics for malware detection using ProvIntSec's provenance information

As stated earlier, the primary task for detection of malicious activities by a provenance scheme is to prove that the data collected by ProvIntSec is of considerable integrity to be used in the context of cloud security [58]. Since network based communication between vm-instances are involved in the cloud, so the primary task is to determine the network oriented malware activities. A malware such as a worm consists of numerous components which are illustrated below.

1. The malicious code imposes on infecting the cm-instances as well the cloud master server.

2. The attack imposes on causing malfunctioning of the critical cloud processes.

3. The attack aims to persist as long as time permits,

Generally, the malicious code is a lading or a dropper with the sole aim of obtaining the lading and executing those on the server. The lading constitutes of multiple actions which can be performed in a system which has been infected, including creation of backdoors, enforcing load on the servers active processes or recoiling system critical data and files [59]. The chief attribute of all the malicious code tested in ProvIntSec is the ability to spread to other vm instances over the network and carry out the same cycle of harmful activities [60].

The malware generator codes were obtained from Symantecs list of Linux Worms for Research and the test standards were inherited from Symantecs definition of a standard malware [61]. According to Symantecs requirements, in order to be a standard Linux malware, a malicious code must exhibit the following attributes.

1. ***Persistence and Stealth:*** Involves the ability to the malware to persist in the system without triggering the security mechanism of the system [62].

2. ***Exfiltration:*** Includes the ability of the malware to pass critical system information to the attacker, obtain unauthorized access of the machine and commit itself to system memory [63].

3. ***Spread:*** Included the capability of the malware to impose itself and attack the remaining machines of the system within short period of primary attack [63].

The malwares which were run for the experiment had the ability to move from one machine to another, which provided to test the ability to detecting socket based provenance for ProvIntSec. This is critically importance since recording provenance information of files moving across multiple machines on the cloud will enable detection of critical file movement by cloud administrators.

The malware codes were executed in ten vm instances, with ProvIntSec installed in all of those. The vms were created using OpenStack cloud and all has Ubuntu

Servers installed as the base operating system. At the same time two master cloud machines were allocated for the experiments where the same malicious codes were executed to identify the effects on virtual machines created by those. Malicious agents were executed in one vm-instance, which spread to multiple vms using the network. Then ProvIntSec collected the provenance data, the journals of which were collected from specific file paths and analyzed manually to obtain signs of malicious activities identified in [64].

5.11 Implementation of ProvIntSec for Malware Detection

Ten malware codes [i.e, worms] were executed in this experiment and performance of ProvIntSec analyzed for those. Each malware, with the exception of ZipWorm and Remen were executed 200 times in the vm instances. The reason for the exception is stated later. The vm instances where the codes were executed are identified in column 4. At the same time critical path wards in the provenance journals defined in [65] were used to identify presence of infection of the system. As stated earlier the journals of ProvIntSec was filtered and then manually analyzed to identify the critical path words which are signatures for reach malware. Detection of malware standard test and performance of ProvIntSec in this regard are described in the following sections.

1. ***Persistence and Stealth:*** One of the primary actions taken by the malicious application is to ensure that the malware runs as long as time allows [66]. One of the most important aspects of ensuring persistence for a malware is to assure the stealth factor, which is the ability to remain undetected or unrecognized while executing the objective in the computing environment. Once the malware

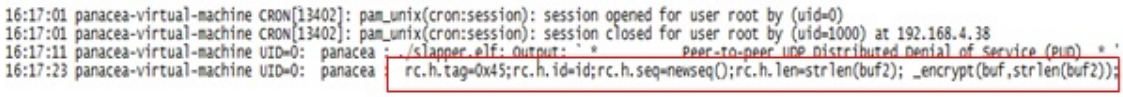


Figure 5.9: ProvIntSec provenance journal for Kippo worm tested in the experiment

achieves stealth, the system monitor will not identify the malware as it crawls through the system. In order to rerun when the system is rebooted, the malware should write itself in system hard drive so that it will be executed every time the system runs [66]. A goal of Linux malware is to successfully infect the init.d process, which runs every time a systems loads [66].

In this experiment of ProvIntsec in which 10 standard Linux worms are implemented [67], it is seen that Kippo, Satyr and Slapper worms infected the init process by affecting rc.local or rc.h files of the cloud vm-instances with ip-addresses 192.168.4.38 and 192.168.4.39 respectively. ProvIntSec collected journals of the attack shown in Figure 5.1.

As seen in the ProvIntSec provenance journal collected for Slapper worm in this experiment, clear signs are obtained that indicate the infection of the rc.h file of the cloud testbest used for this research has been infected by the worm.

Lupper worm implemented during the experimental implementation exhibits characteristics of ensuring persistence through execution of chron jobs on a specific port as identified in the following provenance journal.

As identified by the provenance collection of this experiment, persistence and stealth of Lupper worm is achieved when the malicious code adds CRON jobs through affecting system binaries and memory components and assuming that the daemon is executed each time the system reboots [67]. Certain worms implement more advanced techniques of ensuring persistence such as association with Apache Web Server to ensure that the malicious script is executed each

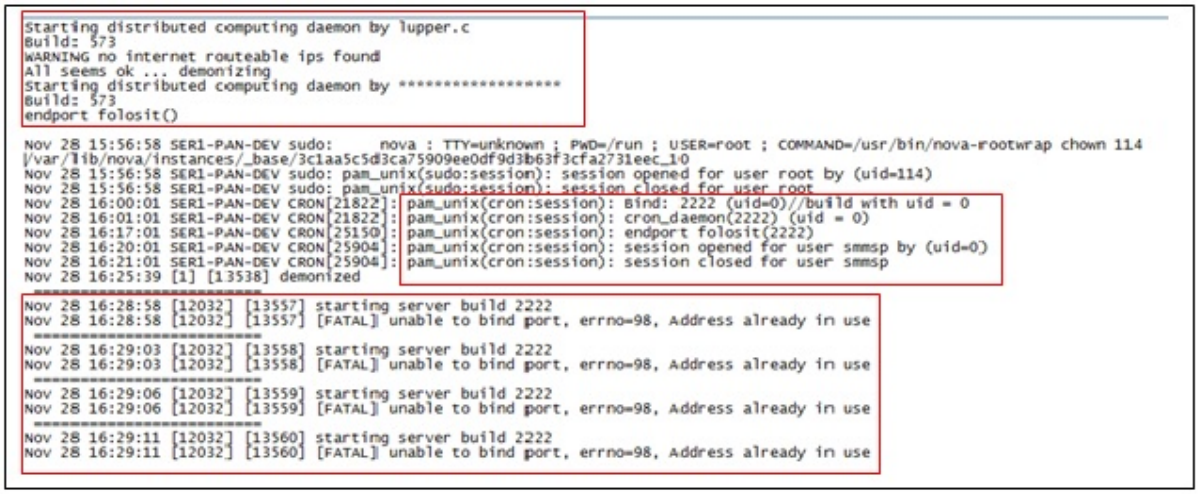
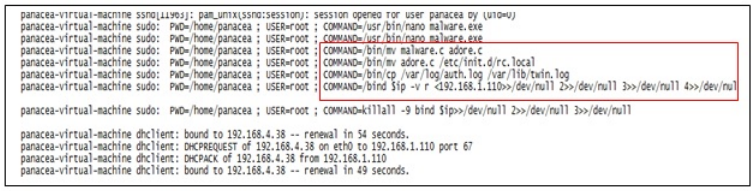


Figure 5.10: ProvIntSec provenance journal for Lupper worm tested in the experiment

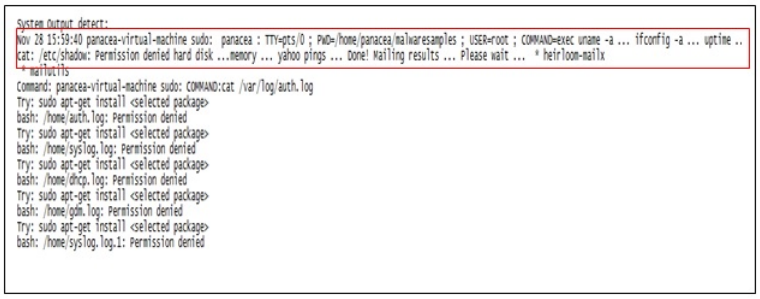
time the server runs [67]. In addition, worms like Adore and Millen make C header files which are attached to every C code file, as a result the malicious code is compiled and executed each time a C code is compiled [67].

2. ***Exfiltration:*** Exfiltration is the process of removing or copying critical system files or data using stealth and transferring those to the remote attacker [68]. In this regard, the Adore, Kippo, Satyr and SSHBruteForce show strong signs of exfiltration as they transfer data to remote host by renaming the file, changing the file permission and copying the file over the network. In this case the attack was made from a malicious vm instance to another vm with the sole aim of transferring the authentication logs of the target vm to the attacker through the Adore worm. The provenance journal of the proposed scheme detects the above activity as shown in the journal block shown in Figure 5.3.

   The SSHBruteForce exhibits exfiltration through ensuring that the critical system processes such as accessing web, checking ping status, checking mail, system uptime services are blocked for the authenticated user and provides malicious attackers access to the vms web access through which the attackers steals im-



(a) Adore



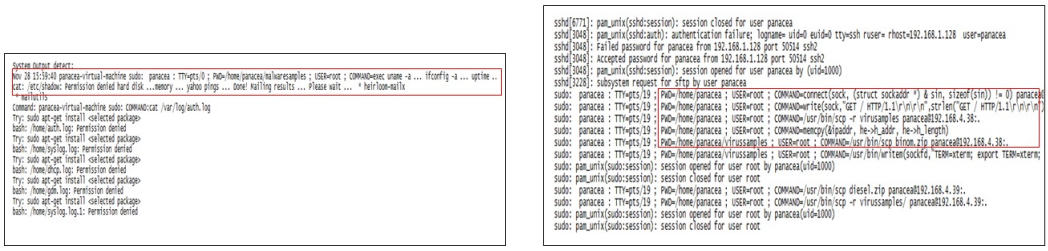
(b) SSHBruteForce

Figure 5.11: ProvIntSec provenance journal for detection of Malicious activity in the cloud

portant system data to a remote node using the web. This activity of SSHBrute-Force is captured in the journal block of ProvIntSec during this experiment is shown in Figure 5.4.

3. ***Spread:*** The experiments involved two ways to spread the malware across multiple virtual machine instances of the cloud. One way is to use Hyper Text Transmission Protocol (HTTP) using the Apache Web Service as captured by ProvIntSec in the case of Mighty worm the journal block of which is shown below [69].

   The Mighty worm connects 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. Once the request is accepted by the target vm, then the worm can crawl through the attacked vm instance and execute the characteristic infectious operations that hinder system performance [69]. In this way the mighty worm spreads across multiple instances in the cloud which



(a) Kippo (b) Satyr

Figure 5.12: ProvIntSec provenance journal for detection of Malicious activity in the cloud

is successfully captured by ProvIntSec.

In addition to the above, Linux worms possess the capability of spreading through the secured socket shell and transferring files through secure copy or scp [70]. The ProvIntSec journal for Kippo worm captures the stated activity.

ProvIntSec successfully detected through the provenance data that Satyr exhibits characteristics of binding with the logical memory and spreads whenever that memory block is copied to another vm-instance of the cloud. If the memory block is copied to the master machine, the master machine is infected. Following log identifies the memory write of Satyr through collecting provenance of memory file from system. Hence it is seen that ProvIntSec identifies critical information about malicious worm attacks on the cloud with regards to the standards specified by Symantec [70].

5.12 Precision and Overhead Measurement for Malware Provenance

Table 5.1 identifies the compiled results of the experiments. The data obtained are used to identify the precision of the proposed scheme which is determined by finding

the Rate of Detection (ROD) of the provenance model. For the Total Amount of tests for malware x, TAx, the RODx is determined in terms of the Times Detected variable, TDx.

$$RODx = \frac{(TDx)(100)}{TAx}, \forall x \in X \tag{5.1}$$

Overhead for any precision, Op of ProvIntSec can be used to determine if the obtained precision is favorable in terms of the baseline values, Overhead, Ox for RODx and Base Line value of x, BLx is determined by the following formula.

$$Ox = BLx - RODx, \forall x \in X \tag{5.2}$$

The results of this experiment which is tabulated in Table 5.1 satisfies for favorable condition f(x)as shown in the following equation, with the exception of ZipWorm and Remen.

$$f(x) = \begin{cases} +favorable & \text{if } Ox \text{ is less than or equal to 0} \\ -favorable & \text{if } Ox \text{ is greater than 0} \end{cases}, \forall x \in X \tag{5.3}$$

The results of this experiment are tabulated in Table 5.1.

Based on the table 5.1, the precision graphs of ProvIntSec for the 10 test malwares for this experiment is shown in Figure 5.5.

The graph above illustrates the fact that ProvIntSec is capable of assuring Precision of over 80 percent for the eight malwares out of 10 with ZIPWorm and Remen as exceptions. The reason for the exceptions is that the two worms are designed to affect Red Hat Linux systems. The test bed for this research is Ubuntu Server 12.04 Long Term Service as discussed in earlier chapters, hence ZipWorm and Remen worms are

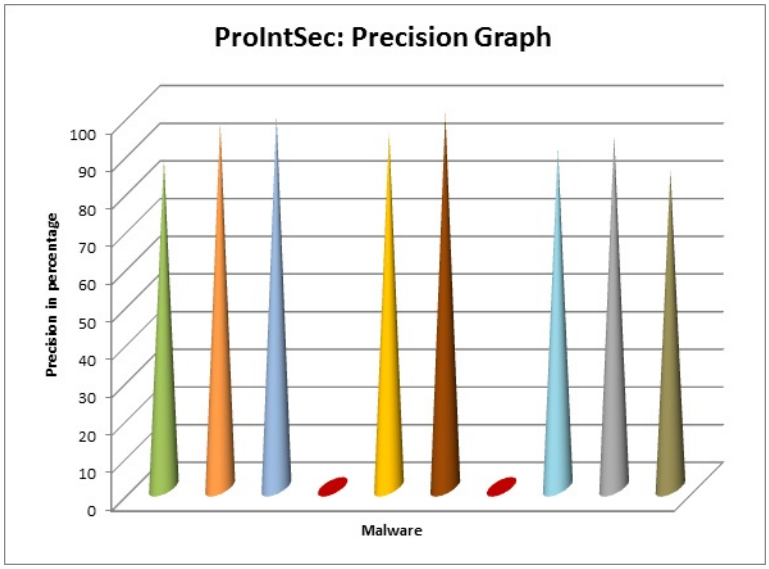


Figure 5.13: Precision Graph of ProvIntSec for the experimented results

not applicable in this regard.

The cones for the remaining 8 malwares applicable to the cloud environment are critical because it should the performance of ProvIntSec which is greater or equal to 84 percent atleast for all cases with respect to the base line which is 80 percent as defined by [71]. The negative overhead in 7 cases out of 8 show the capability of ProvIntSec that there is minimal overhead in terms of performance with respect to the analyzed samples. Figure 5.6 shows the graph of Total Tests and Accuracy of ProvIntSec.

It is clear from the above graph that the provenance detection scheme, ProvIntSec is such that the Accuracy curve has the same W appearance as the Match curve showing that Accuracy has been derived in terms of the latter.

With respect to equation 3, the overhead curve will give an idea of the correctness of results as negative overhead implies that the provenance scheme is favorable for detection of the malicious activity which is shown in Figure 5.7.

Figure 5.7 shows that ProvIntSec performs the best for Kippo worm, and worst

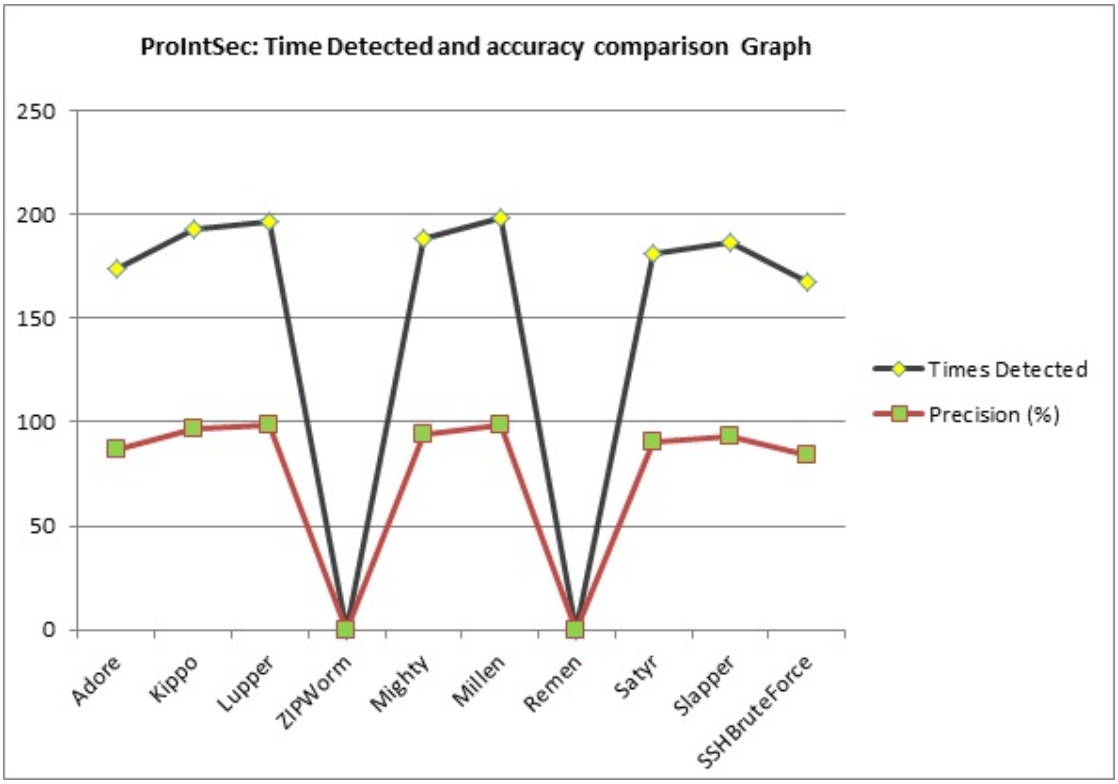


Figure 5.14: Time Detected and accuracy comparison Graphs

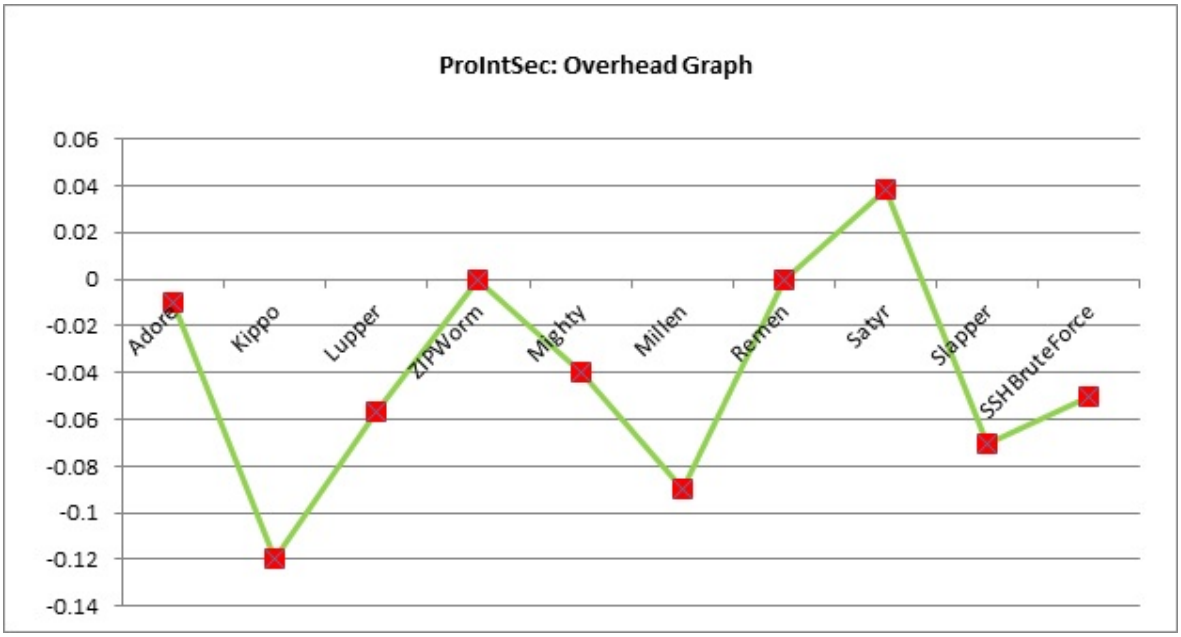


Figure 5.15: ProvIntSec Overhead Graph

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 5.16: ProvIntSec Risk Matrix for 10 Malicious Linux Worms

for Satyr worm with a positive overhead. So it is most positively favorable for Kippo worm and most negatively favorable for Satyr worm with respect to equation 3. Based on the proposed model and with respect to requirements, a risk matrix identifying the probability of occurrence of the 10 malware and their threat on the cloud infrastructure if ProvIntSec is installed in those is presented in Figure 5.8.

It is clear from Figure 5.8 that Satyr is the most dangerous worm attack among all the 10 test worms for a system which is protected by ProvIntSec as a provenance detection scheme. Remen and ZipWorm have minimum threats and Millen, Mighty and SSHBruteForce have the greatest frequency of occurrence and Adore has maximum threat level with reduced frequency.

5.13 Implementation of ProvIntSec for Socket provenance Detection

As seen in the previous section ProvIntSecs ability for detection of malicious entities shows effective provenance detection with desirable performance. Similar results are obtained for socket based provenance detection using ProvIntSec. The proposed

Malware Name	Symantec Malware Characteristic			
	Persistence	Stealth	Exfiltration	Spread
Adore	0	0	1	1
Kippo	1	1	1	1
Lupper	1	1	1	0
ZIPWorm	0	0	0	0
Mighty	0	0	1	1
Millen	1	1	1	0
Remen	0	0	0	0
Satyr	1	1	1	1
Slapper	1	1	0	0
SSHBruteForce	0	0	1	1

Figure 5.17: Malware Characteristics for the 10 worms tested in the experiment

solutions need to retrieve provenance for network socket activities arises from its requirement for monitoring file transfers of various sizes across multiple vm instances and from master to vm instances in cloud computing environment [72]. To allow interoperating with existing cloud infrastructure which does have provenance detection service installed, activities concerning file copy or movement across systems have been considered for evaluation which is illustrated in the implementation [73].

The aim of ProvIntSec implementation is to identify provenance journals for file transfers which constitute of the entire process tree across multiple vms involved in the transmission process, time of action, command executed and intermediate child processes that participate in the transmission of data [74]. Hence file transfers across multiple vm instances can be identified and verified manually [75].

Implementation of socket provenance to detect file transfers involve transfer of variable sized data files over 800 times for each type of file size taken under consideration. The requirement of size range for provenance detection consists of a 1 Kilobytes to 10 Gigabytes [75]. Based on the standard range of files, each type is

Table 5.2: Results of ProvIntSec in Malware Detection provenance

Malware Name	Type	Rounds	VMs	Detected	Precision	Baseline
Adore	Worm	200	Experiment-1	174	87.0	86.0
Kippo	Worm	200	Experiment-1	193	96.4	82.4
Lupper	Worm	200	Experiment-1	197	98.7	93.0
ZIPWorm	Worm	000	Experiment-1	000	0000	na
Mighty	Worm	200	Experiment-1	189	94.0	90.0
Millen	Worm	200	Experiment-1	199	99.0	90.0
Remen	Worm	000	Experiment-2	000	0000	na
Satyr	Worm	200	Experiment-2	181	90.0	93.8
Slapper	Worm	200	Experiment-2	197	93.4	86.0
SSHBruteForce	Worm	200	Experiment-2	168	84.0	89.0

Table 5.3: Results of ProvIntSec in Socket provenance detection

File Size	Total	Detected	Missed	Accuracy	Baseline	Overhead	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

transferred across the cloud network 800 times which included multiple vm-instances and master servers of cloud environment. Files transferred with secure copy (scp) or move (mv) command in Linux is detected through inspection of the provenance journals across multiple vm instances in the cloud. The number of successful detection from ProvIntSec and total number of file transfers is identified and cumulative results are tabulated in Table 5.2.

As seen above smaller file sizes in the first two rows shows positive overhead, indicating that system performance is unfavorable. However, performance significantly increases for larges file sizes as ProvIntSec captures provenance of file movements with accuracy that is well above the required base line [75]. Performance of ProvIntSec with the concerned area of socket provenance is illustrated in the following section.

The performance of ProvIntSec in terms of precision and overhead is provided in the tables above. At the same time the average times taken by file transfers of different file sizes are also presented in Table 5.2. It is seen that performance improves in proportion to the time taken for file transfer. So relationship between the trasfer time and precision of ProvIntSec can also be derived from the table.

5.14 Precision and Overhead Determination for Socket Provenance

Figure 5.9 shows the accuracy level of ProvIntSec for the different file sizes expressed as percentage. As seen in the figure, small file sizes shows less accuracy in provenance capture using ProvIntSec which is due to the inability of Linux inotify to capture multiple file transfers within 0.018 seconds. Since Linux inotify has been used in this experiment as the base of ProvIntSec to determine provenance, small file sizes pose a comparatively greater overhead.

Significant performance improvements have been identified for large file sizes as performance is unhindered because Linux inotify is capable of detecting files which are transferred at a rate that is greater than 0.018 seconds. The results show that ProvIntSec performs better for larger data files. This is critically important for the detection of drones in Distributed Denial of Service (DDoS) attacks since the drones are scripts that put payload on the system and files of those are relative large in size [76]. The tabulated results of this experiment shown in Table 5.2 also depict the baseline which is 80 percent for all file sizes as stated in [76]. Comparison with other provenance detection schemes have not been carried out since the base line is used for performance analysis which acts as a benchmark [77].

Figure 5.10 identifies the amount of detection out of the total of 800 rounds

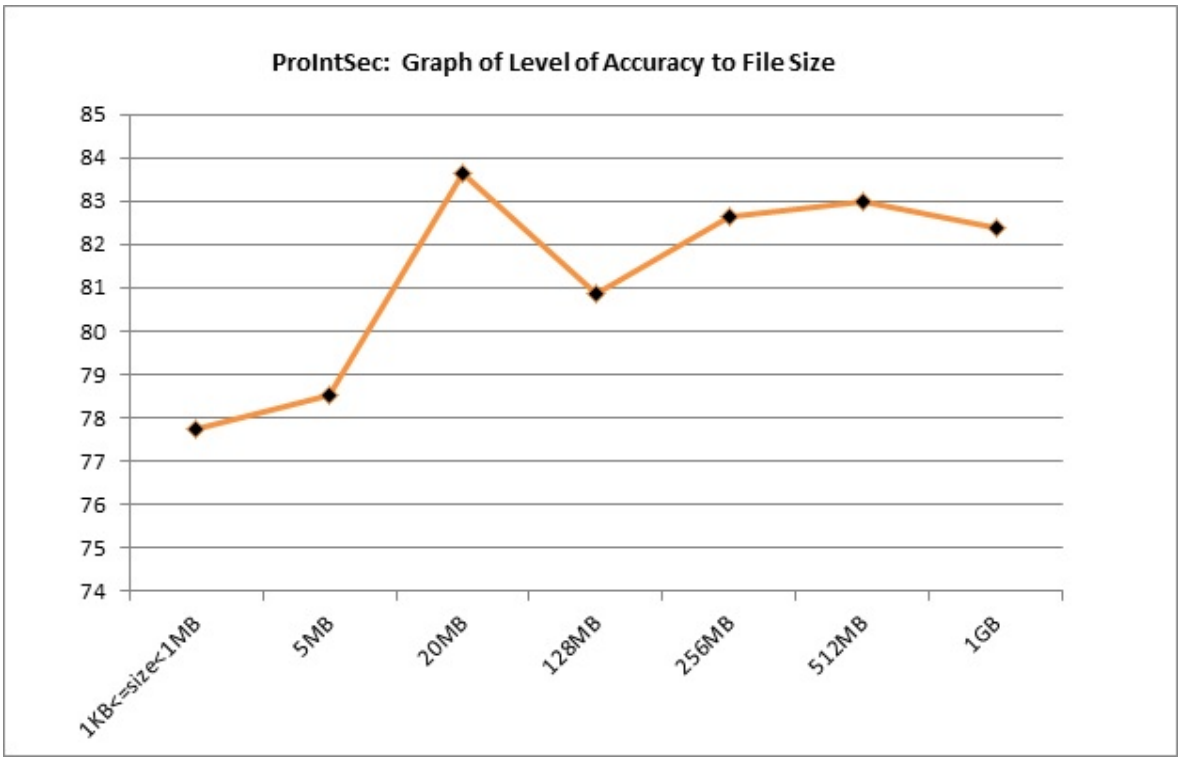


Figure 5.18: ProvIntSec precision graph of Level of Accuracy to File Size

of operating executed for each file size which accumulates to 5600 rounds out of which 4550 have been detected with a cumulative overhead of -8.77. The negative overhead implies that the system does not have significant factors which can degrade the performance of ProvIntSec.

Finally Figure 5.11 identifies the scattered overheads of respective test cases of different file sizes. As discussed earlier, overhead for smaller file sizes is positive but those of large file sizes are negative which is favorable for ProvIntSec.

For both Malware provenance and Socket provenance detection, ProvIntSec has shown favorable performance with respect to the base line benchmarks chosen for this research. The fact must be addressed that the ability of ProvIntSec to detect both types of provenance data for open source cloud will enable Cloud Administrators identify nefarious activities in the cloud and also unauthorized or non- permitted

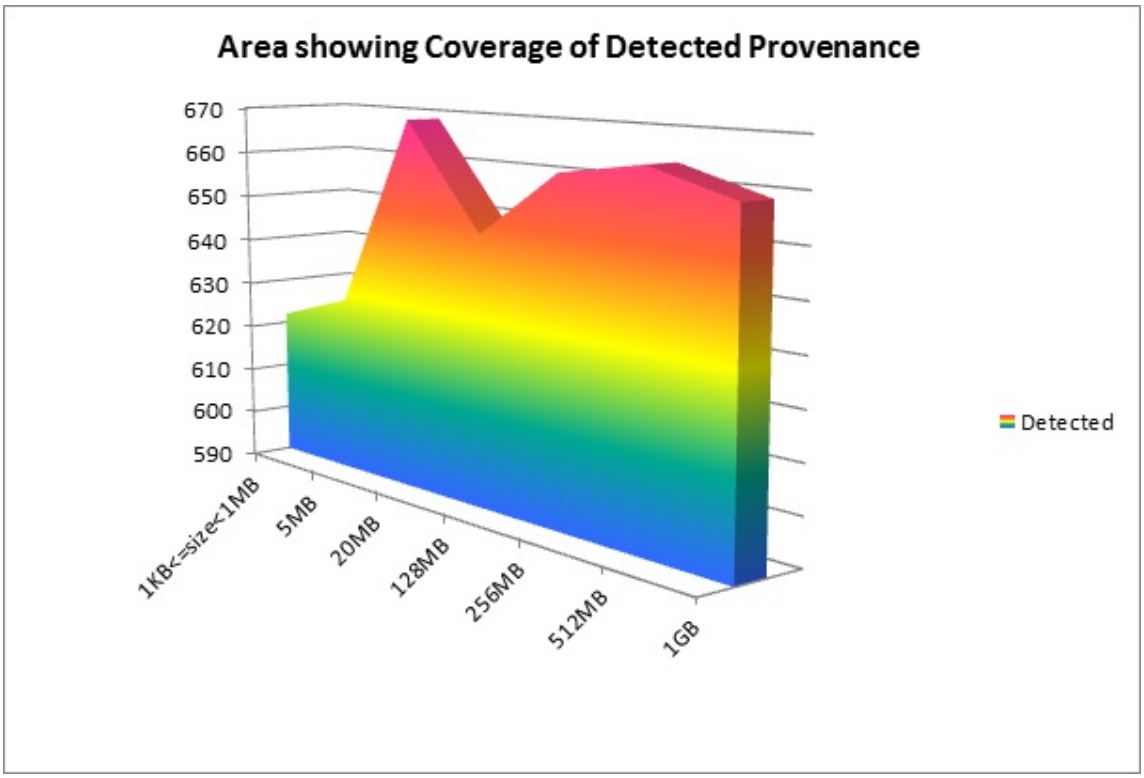


Figure 5.19: Area curve showing the extent to which file movement in the cloud has been detected using ProvIntSec

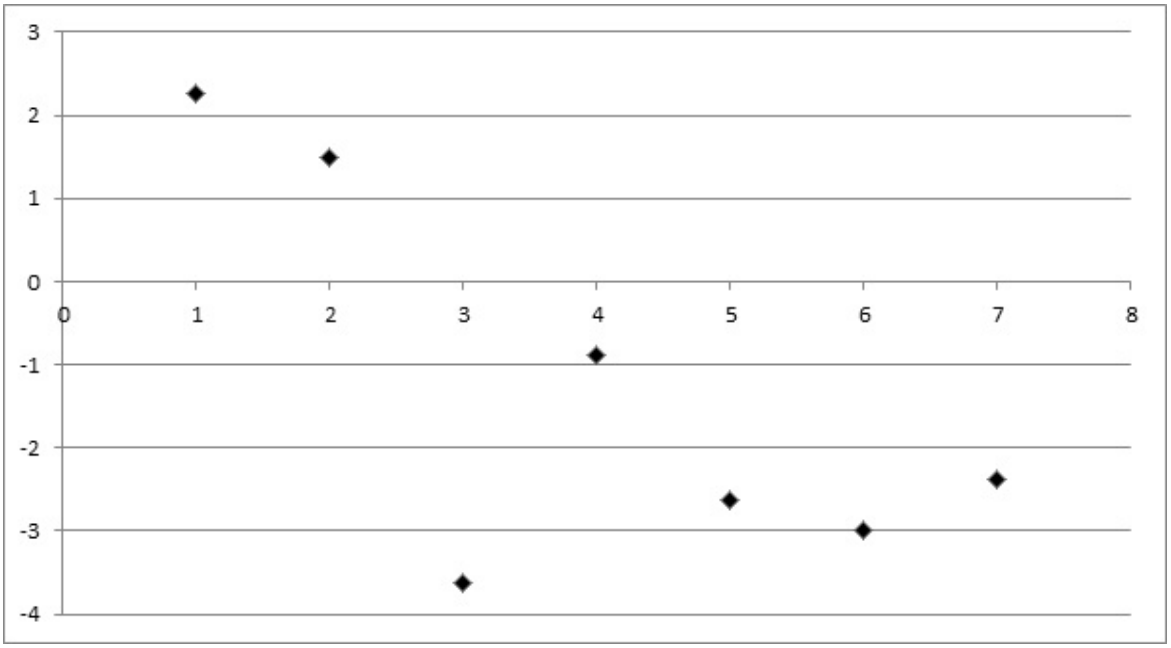


Figure 5.20: Scatter diagram showing the positions of overheads of different test cases used in the experiment

file transfer across multiple vm instances and in between physical cloud controllers and nodes. The ability of ProvIntSec to detect provenance for both two types of operations, socket and malicious, make ProvIntSec complete All Provenance Detector (APD) [78].

Chapter 6

Conclusion and Future Work

The research aimed to ascertain a solution to the bottleneck of capturing provenance data in a widely decentralized system such as cloud computing. The research succeeded in devising ProvIntSec, which is a blueprint to the solution of the stated problem and provides a journal based provenance detection technique for open source cloud computing.

6.1 Discussion

In the experimentation, ProvIntSec successfully analyzed system critical files of vm-instances as well as cloud controllers and nodes. The detected data was used to render useful provenance information that enable cloud administrators detect malwares and unauthorized file transfers in open source cloud more effectively. As a result, the research show that ProvIntSec enable cloud administrators manage the open source cloud infrastructure more effectively.

Empirical investigation was carried out to evaluate the performance of ProvIntSec in open source cloud environments, in which a precision rate of 92.81 percent was obtained for detecting malware provenance and 81.24 percent was obtained for detecting socket provenance. Upon comparison of obtained test results with pre-defined baseline and benchmark values, desirable overheads of -0.399 and -8.777 were obtained for malware provenance and socket provenance respectively. This shows that from the performance perspective, ProvIntSec has acceptable precision rate to be implemented in a commercial cloud infrastructure based on open source cloud.

Compared to traditional provenance detection techniques which function in stan-

dalone systems, ProvIntSec is capable of detecting provenance in a highly virtualized environment like the cloud, which is a strict requirement [79]. Distributed provenance schemes deal with detection of malicious activities for a small number of two to three malware [80]. However, the results of ProvIntSec were obtained through analysis of 10 Linux worms running in cloud environment with a total number of 1600 runs, and thousands of file transfer test cases across vm-instances to detect socket provenance. Hence the research was result-oriented and aimed to provide a generalized solution of provenance detection for open source OpenStack cloud.

6.2 Future Work

As stated earlier, ProvIntSec detects provenance for the identification of Linux worms infecting open source cloud and also socket provenance for unauthorized file transfer over the cloud network. However, performance evaluation for other malicious entities like rootkits and key loggers were considered to a minimal extent. In the case of socket provenance, ProvIntSec performs considerably well for transfer of large sized files using secure copy and move commands in Linux. However, ProvIntSec has limited capability in provenance capturing of file transfer scenarios where a new file is created and the contents of the old file are copied to the new file in a remote location of the distributed cloud environment.

With respect to the above limitations, scope of future research lies in performance evaluation of ProvIntSec for Linux malwares not tested up until now. Performance of ProvIntSec for provenance detection of mass file copying and copy of a large number of very small files are issues of future research. Programs analysis for ProvIntSec should be carried out to obtain the time and memory complexities of the proposed algorithms and aiming to improve the duration and storage multiplicities of those.

Establishment of a new open source cloud computing paradigm by the name of Open Cloud Integrity as a Service (OCIaaS) using ProvIntSec is an area of future research interest.

Appendix A

Appendix: OpenStack Cloud Compute setup

sudo apt-get install bridge-utils

sudo apt-get install ntp

sudo nano /etc/ntp.conf

server ntp.ubuntu.com

server 127.127.1.0

fudge 127.127.1.0 stratum 10

sudo service ntp restart

sudo apt-get install mysql-server python-mysqldb

sudo nano /etc/mysql/my.cnf

bind-address = 0.0.0.0

sudo restart mysql

sudo mysql -uroot -pepz1971

CREATE DATABASE nova;

CREATE USER 'novadbadmin'@'192.168.1.226';

GRANT ALL PRIVILEGES ON nova.* TO 'novadbadmin'@'192.168.1.226';

SET PASSWORD FOR 'novadbadmin'@'192.168.1.226' = PASSWORD('epz1971');

CREATE DATABASE glance;

CREATE USER 'glancedbadmin'@'192.168.1.226';

GRANT ALL PRIVILEGES ON glance.* TO 'glancedbadmin'@'192.168.1.226';

SET PASSWORD FOR 'glancedbadmin'@'192.168.1.226' = PASSWORD('epz1971');

CREATE DATABASE keystone;

CREATE USER 'keystonedbadmin'@'192.168.1.226';

GRANT ALL PRIVILEGES ON keystone.* TO 'keystonedbadmin'@'192.168.1.226';

SET PASSWORD FOR 'keystonedbadmin'@'192.168.1.226' = PASSWORD('epz1971');

sudo apt-get install keystone python-keystone python-keystoneclient

sudo nano /etc/keystone/keystone.conf

connection = mysql://keystonedbadmin:epz1971@192.168.1.226/keystone

sudo service keystone restart

sudo keystone-manage

keystone tenant-create –name admin

+———-+————————-+ — Property — Value — +———-+————————-+ — description — None — — enabled — True — — id — ff210aa4948e447b8a22927ef3e48f45 — — name — admin — +———-+———————————-+

keystone tenant-create –name service +———-+————————-+ — Property — Value — +———-+————————-+ — description — None — — enabled — True — — id — 66a3e88dbe2d4919a6f570d2f4dda48a — — name — service — +———-+————————-+

sudo apt-get install sendmail

keystone user-create –name admin –pass admin –email admin@panaceasystemsbd.com

keystone user-create –name nova –pass nova –email nova@panaceasystemsbd.com

keystone user-create –name glance –pass glance –email glance@panaceasystemsbd.com

keystone user-create –name swift –pass swift –email swift@panaceasystemsbd.com

———————————————————————— keystone role-create –name admin

keystone role-create –name Member

keystone tenant-list

keystone user-list

keystone role-list ———————————————————————— +—

———-+————————————+ — Property — Value — +————-+—

————————————+ — adminurl — http://192.168.1.226:8773/services/Admin

— — id — 68817d5a74194fcbbf8e0c77a9d3a21e — — internalurl — http://192.168.1.226:8773/servic

— — publicurl — http://192.168.1.226:8773/services/Cloud — — region — myregion

— — serviceid — 546b26fcc6f145e8a234dbb0a5dbb51a — +————-+————————

————————+

+————-+————————————-+ — Property — Value — +————-

+————————————-+ — adminurl — http://192.168.1.226:35357/v2.0 — —

id — 1366a071abe64227bd51798a344aa723 — — internalurl — http://192.168.1.226:5000/v2.0

— — publicurl — http://192.168.1.226:5000/v2.0 — — region — myregion — —

serviceid — 35fc871c6a1b46bb9dadf76eee027dcf — +————-+————————

———-+

+————-+————————————-+ — Property — Value — +—

———-+————————————-+ — adminurl — http://192.168.1.226:8080/v1

— — id — 297c96ceb0834a03882d870a43510c86 — — internalurl — http://192.168.1.226:8080/v1/A

//192.168.1.226 : 8080/$v1/AUTH$(tenantid)s — — region — myregion — — service

id — d6de43fa8bc6487092fa8edf9e9ff57b — +————-+————————————

————-+

Bibliography

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

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

[3] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee, 2008.

[4] S. Miles, P. Groth, S. Munroe, S. Jiang, T. Assandri, and L. Moreau. Extracting causal graphs from an open provenance data model. *Concurrency and Computation: Practice and Experience*, 20(5):577–586, 2008.

[5] W.C. Tan. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull*, 30(4):3–12, 2007.

[6] 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.

[7] 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.

[8] C. Wang, S. Chen, and J. Zic. A contract-based accountability service model. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 639–646. IEEE, 2009.

[9] 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.

[10] M. Mowbray and S. Pearson. A client-based privacy manager for cloud computing. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, page 5. ACM, 2009.

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

[12] S. Pearson and A. Benameur. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 693–702. IEEE, 2010.

[13] M.A. Vouk. Cloud computing–issues, research and implementations. *Journal of Computing and Information Technology*, 16(4):235–246, 2004.

[14] 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.

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

[16] 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.

[17] W.Z.P. Ning, R. Wang, Z. Zhang, G. Ammons, and V. Bala. Always up-to-date–scalable offline patching of vm images in a compute cloud. In *ACSAC*, volume 10, pages 6–10, 2010.

[18] W. Zhou, M. Sherr, T. Tao, X. Li, B.T. Loo, and Y. Mao. Efficient querying and maintenance of network provenance at internet-scale. In *Proceedings of the 2010 international conference on Management of data*, pages 615–626. ACM, 2010.

[19] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1–11, 2011.

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

[21] D. Lekkas and D. Zissis. Leveraging the e-passport pki to achieve interoperable security for e-government cross border services. *Global Security, Safety and Sustainability & e-Democracy*, pages 96–103, 2012.

[22] W. Zhou, L. Ding, A. Haeberlen, Z. Ives, and B.T. Loo. Tap: Time-aware provenance for distributed systems. In *Proc. USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, 2011.

[23] W. Zhuo, Q. Fei, S. Sun, T. Tao, A. Haeberlen, Z.G. Ives, B.T. Loo, and M. Sherr. Nettrails: a declarative platform for maintaining and querying provenance in distributed systems. 2011.

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

[25] R.K.L. Ko, P. Jagadpramana, and B.S. Lee. Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 765–771. IEEE, 2011.

[26] 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.

[27] M. Sakka, B. Defude, and J. Tellez. Document provenance in the cloud: constraints and challenges. *Networked Services and Applications-Engineering, Control and Management*, pages 107–117, 2010.

[28] 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.

[29] S. Sakr, A. Liu, D.M. Batista, and M. Alomari. A survey of large scale data management approaches in cloud environments. *Communications Surveys & Tutorials, IEEE*, 13(3):311–336, 2011.

[30] P. Alvaro, T. Condie, N. Conway, K. Elmeleegy, J.M. Hellerstein, and R. Sears. Boom analytics: exploring data-centric, declarative programming for the cloud. In *Proceedings of the 5th European conference on Computer systems*, pages 223–236. ACM, 2010.

[31] J. Zou, Y. Wang, and K.J. Lin. A formal service contract model for accountable saas and cloud services. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 73–80. IEEE, 2010.

[32] K.E. Pavlou and R.T. Snodgrass. Temporal implications of database information accountability. In *Temporal Representation and Reasoning (TIME), 2012 19th International Symposium on*, pages 125–132. IEEE, 2012.

[33] C. Wang, S. Nepal, S. Chen, and J. Zic. Cooperative data management services based on accountable contract. *On the Move to Meaningful Internet Systems: OTM 2008*, pages 301–318, 2008.

[34] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, 2007.

[35] S. Pearson. Taking account of privacy when designing cloud computing services. In *Software Engineering Challenges of Cloud Computing, 2009. CLOUD'09. ICSE Workshop on*, pages 44–52. IEEE, 2009.

[36] A. Khajeh-Hosseini, I. Sommerville, and I. Sriram. Research challenges for enterprise cloud computing. *arXiv preprint arXiv:1001.3257*, 2010.

[37] S. Biggs and S. Vidalis. Cloud computing: The impact on digital forensic investigations. In *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*, pages 1–6. IEEE, 2009.

[38] S.M.S. da Cruz, M.L.M. Campos, and M. Mattoso. Towards a taxonomy of provenance in scientific workflow management systems. In *Services-I, 2009 World Conference on*, pages 259–266. IEEE, 2009.

[39] M. Anand, S. Bowers, T. Mcphillips, and B. Ludäscher. Exploring scientific workflow provenance using hybrid queries over nested data and lineage graphs. In *Scientific and Statistical Database Management*, pages 237–254. Springer, 2009.

[40] L. Ding, J. Michaelis, J. McCusker, and D.L. McGuinness. Linked provenance data: A semantic web-based approach to interoperable workflow traces. *Future Generation Computer Systems*, 27(6):797–805, 2011.

[41] S. Rozsnyai, A. Slominski, and Y. Doganata. Large-scale distributed storage system for business provenance. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 516–524. IEEE, 2011.

[42] 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.

[43] I.M. Abbadi and J. Lyle. Challenges for provenance in cloud computing. In *in 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP11). USENIX Association*, 2011.

[44] R.L. Grossman. The case for cloud computing. *IT professional*, 11(2):23–27, 2009.

[45] M.D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali. Cloud computing: Distributed internet computing for it and scientific research. *Internet Computing, IEEE*, 13(5):10–13, 2009.

[46] J.J. Tran, L. Cinquini, C.A. Mattmann, P.A. Zimdars, D.T. Cuddy, K.S. Leung, O.I. Kwoun, D. Crichton, and D. Freeborn. Evaluating cloud computing in the nasa desdyni ground data system. In *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*, pages 36–42. ACM, 2011.

[47] A. Imran, A.U. Gias, and K. Sakib. An empirical investigation of cost-resource optimization for running real-life applications in open source cloud. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 718–723. IEEE, 2012.

[48] G. Chang, E. Law, and S. Malhotra. Demonstration of lmmp automated performance testing using cloud computing architecture. In *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*, pages 71–71. ACM, 2011.

[49] W. Dawoud, I. Takouna, and C. Meinel. Elastic vm for cloud resources provisioning optimization. *Advances in Computing and Communications*, pages 431–445, 2011.

[50] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[51] B. Sotomayor, R.S. Montero, I.M. Llorente, and I. Foster. Virtual infrastructure management in private and hybrid clouds. *Internet Computing, IEEE*, 13(5):14–22, 2009.

[52] K. Kambatla, A. Pathak, and H. Pucha. Towards optimizing hadoop provisioning in the cloud. In *Proc. of the First Workshop on Hot Topics in Cloud Computing*, 2009.

[53] M. Taylor, J. Haggerty, D. Gresty, and D. Lamb. Forensic investigation of cloud computing systems. *Network Security*, 2011(3):4–10, 2011.

[54] 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.

[55] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. 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.

[56] J. Lyle, A. Martin, et al. Trusted computing and provenance: Better together. In *TaPP10: 2nd Workshop on the Theory and Practice of Provenance*, 2010.

[57] D. de Oliveira, E. Ogasawara, F. Baião, and M. Mattoso. Scicumulus: A lightweight cloud middleware to explore many task computing paradigm in scientific workflows. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 378–385. IEEE, 2010.

[58] I. Souilah, A. Francalanza, and V. Sassone. A formal model of provenance in distributed systems. *TAPP*, 9:1–11, 2009.

[59] M. Christodorescu, R. Sailer, D.L. Schales, D. Sgandurra, and D. Zamboni. Cloud security is not (just) virtualization security: a short paper. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 97–102. ACM, 2009.

[60] Y. Chen, V. Paxson, and R.H. Katz. Whats new about cloud computing security? *University of California, Berkeley Report No. UCB/EECS-2010-5 January*, 20(2010):2010–5, 2010.

[61] S. Ramgovind, M.M. Eloff, and E. Smith. The management of security in cloud computing. In *Information Security for South Africa (ISSA), 2010*, pages 1–7. IEEE, 2010.

[62] D.G. Feng, M. Zhang, Y. Zhang, and Z. Xu. Study on cloud computing security. *Journal of Software*, 22(1):71–83, 2011.

[63] H. Sato, A. Kanai, and S. Tanimoto. A cloud trust model in a security aware cloud. In *Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on*, pages 121–124. IEEE, 2010.

[64] A. Bisong, M. Rahman, et al. An overview of the security concerns in enterprise cloud computing. *arXiv preprint arXiv:1101.5613*, 2011.

[65] L.M. Vaquero, L. Rodero-Merino, and D. Morán. Locking the sky: a survey on iaas cloud security. *Computing*, 91(1):93–118, 2011.

[66] F. Sabahi. Cloud computing security threats and responses. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 245–249. IEEE, 2011.

[67] N. Gruschka and M. Jensen. Attack surfaces: A taxonomy for attacks on cloud services. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 276–279. IEEE, 2010.

[68] M. Jensen, J. Schwenk, N. Gruschka, and L.L. Iacono. On technical security issues in cloud computing. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pages 109–116. IEEE, 2009.

[69] N. Provos, M.A. Rajab, and P. Mavrommatis. Cybercrime 2.0: when the cloud turns dark. *Communications of the ACM*, 52(4):42–47, 2009.

[70] N. Hawthorn. Finding security in the cloud. *Computer Fraud & Security*, 2009(10):19–20, 2009.

[71] L. Lu, V. Yegneswaran, P. Porras, and W. Lee. Blade: an attack-agnostic approach for preventing drive-by malware infections. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 440–450. ACM, 2010.

[72] F. Lombardi and R. Di Pietro. Secure virtualization for cloud computing. *Journal of Network and Computer Applications*, 34(4):1113–1122, 2011.

[73] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl. Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In *USENIX Security*, volume 8, 2011.

[74] F. Lombardi and R. Di Pietro. Transparent security for cloud. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 414–415. ACM, 2010.

[75] L. Martignoni, R. Paleari, and D. Bruschi. A framework for behavior-based malware analysis in the cloud. *Information Systems Security*, pages 178–192, 2009.

[76] C.A. Martínez, G.I. Echeverri, and A.G.C. Sanz. Malware detection based on cloud computing integrating intrusion ontology representation. In *Communications (LATINCOM), 2010 IEEE Latin-American Conference on*, pages 1–6. IEEE, 2010.

[77] I. Barash, G. Guseinov, A.S. Khetarpal, B. Liu, and S. Zilber. Systems and methods for operating an anti-malware network on a cloud computing platform, June 29 2010. US Patent App. 12/826,583.

[78] 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.

[79] X. Zheng and Y. Fang. An ais-based cloud security model. In *Intelligent Control and Information Processing (ICICIP), 2010 International Conference on*, pages 153–158. IEEE, 2010.

[80] M. Schmidt, L. Baumgartner, P. Graubner, D. Bock, and B. Freisleben. Malware detection and kernel rootkit prevention in cloud computing environments. In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, pages 603–610. IEEE, 2011.