# Incident Handling in IaaS Cloud Environment using Digital Forensic Practices

**THESIS**

Submitted in partial fulfilment

of the requirements for the degree of

**DOCTOR OF PHILOSOPHY**

by

**BKSP KUMAR RAJU ALLURI**
**ID. No. 2013PHXF0411H**

Under the supervision of

**Dr. G. Geethakumari**


BITS Pilani

Pilani | Dubai | Goa | Hyderabad

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**
**2018**

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

# CERTIFICATE

This is to certify that the thesis titled **Incident Handling in IaaS Cloud Environment using Digital Forensic Practices** and submitted by **BKSP KUMAR RAJU ALLURI** ID No **2013PHXF0411H** for award of Ph.D. of the Institute embodies the original work done by him under my supervision.

Signature of the Supervisor

Name in capital letters **DR. G. GEETHAKUMARI**

Designation Asst Professor, Dept. of CSIS

Date:

# Acknowledgements

Foremost, I would like to express my deepest thanks to my supervisor Dr. G.Geethakumari for all her suggestions and constant support during this research. Her valuable guidance and encouragement throughout the period were critical factors which contributed towards completion of the work. Through her untiring efforts, she helped me to critically analyse the problems in a systematic manner and consider innovative approaches to evolve practical solutions.

I would like to thank MeitY, Govt. of India, for sponsoring my work. I would like to express my gratitude to the PRSG Chairman Prof. PJ Narayanan, Director, IIIT Hyderabad, for his valuable suggestions throughout the course of my Ph.D. and project work.

I would also like to thank Prof. R. Gururaj and Prof. Thatagata Ray, members of my doctoral advisory committee for their constant review and invaluable suggestions in steering the work. I would also like to express my gratitude to other members of the faculty in the Department of Computer Science and Information Systems Prof. Chittaranjan Hota, Prof. Bhanu Murthy, Dr. Aruna Malapati, Mr. KCS Murti, for all their suggestions and encouragement during various presentations and whenever I interacted with them. I would also like to thank each researcher in the department for all the wonderful time we shared during our work.

Finally, my sincere acknowledgement of the sacrifices and support made by each member of my family during this period. They were my pillars of strength, always understanding and encouraging me. Without their support, this work would never have been completed.

BITS Pilani, Hyderabad Campus — BKSP Kumar Raju Alluri

April, 2018

# Abstract

Cloud computing, as a computational paradigm, has enticed the information technology community to facilitate various services with less operational and maintenance costs. However, the occurrence of various cloud incidents is affecting the trust of users on the cloud environment. The scope of our work is to handle security incidents occurring at the Infrastructure as a Service (IaaS) cloud systems. Incident handling in cloud is relatively new and involves various technical, organizational and legal challenges. Traditional incident handling approaches cannot be directly applied to the cloud environment due to its unique aspects like multi-tenancy, physical inaccessibility, lack of transparency and rapid elasticity. In this thesis, we handle cloud incidents using the stages of digital forensics as this would increase the availability of evidences of the occurred incident which in turn would be the key factor in effective incident handling.

We acquired various cloud specific evidences (vRAM, Service logs, Snapshots and vDisk) at the IaaS user level and proposed the corresponding analysis approaches to handle cloud incidents. Since the integrity and availability of the vRAM evidence acquired at the virtual machine level is questionable, we proposed a trigger-based introspection model to capture reliable and relevant vRAM events without compromising on its transparency. Cloud systems introduce additional incident handling challenges with new evidences like service logs. We identified the role of service logs for effective incident handling and proposed a model which can allow the incident handler to analyze the service logs effectively. Virtual Machine (VM) snapshots in the cloud are generally used for backup and restoration purposes. We made use of these snapshots to handle cloud incidents by proposing a provenance system. Finally, we came up with a methodology for correlating multiple evidences, which can help the incident handler arrive at quick logical findings about the occurred cloud incident. The proposed models for cloud incident handling are validated using an Openstack cloud test bed.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

*We do not need a hard disk in the computer if we can get to the server faster and carrying around these non-connected computers is byzantine by comparison.*

-Steve Jobs

Incident Handling is an integral part of security incident management of an organization. It starts with preparing the target environment to reduce the risks of the incident and ends with eradicating similar incidents in future [1]. The technological advancements in the information technology changed the directions in the way data is stored, processed and analyzed. This benefits the end user to receive highly valued services with less cost, less time and with more accuracy. On the other side, this opens up opportunities to the adversaries to intrude the network and compromise the target system(s) in the organization's environment.

Over the last few years, security threats have increased drastically in terms of numbers, level of sophistication and disruption. Some reported incidents include: in the year 2016, Yahoo announced that 500 million user accounts were hacked and their data was available for sale in the dark web [2]. A hacker group from Russia compromised the Oracle MICROS point of sale systems which is among the top three gateways globally [3]. In 2014, eBay was affected by a data breach in which more than 230 million users sensitive data was stolen [4]. Alone in 2014, Computer Emergency Response Team (CERT) of Malaysia received complaints about more than 4000 incidents [5].

Risk assessment strategies do exist to reduce the effect of the occurred incident. Since new incidents are being reported frequently, it is not possible to completely prevent them. Incident handling capabilities are therefore required to identify and analyze all the threat aspects and reduce its impact on the organizational assets and services. Incidents differ based upon the type and the targeted domain. Generalizing, the adversary can perform incidents like unauthorized access, inappropriate usage, malicious code injection, resource exploitation and multi-component threat strategies.

Handling the incidents in a timely fashion would benefit the organization in terms of identifying relevant incident response strategy, quickly recovering from the incident and improving the existing security policies.

## 1.1 Incident Handling

### 1.1.1 What is an incident ?

There are several definitions for ”Incident” and it varies from one organization to the other. Generalizing all of them, SANS (System Administration, Networking and Security Institute) gave a definition for incident as, *an adverse event in an information system which is an attempt to harm the system.* According to CERT (Computer Emergency Response Team) guidelines, the organization can conform the occurrence of an incident if atleast one of the following with respect to the violation of the security policies happens:

- An attempt to obtain unauthorized access
- Modifications without the actual owner instruction or knowledge
- Denial of resources
- Unauthorized use

### 1.1.2 Phases in the Incident Handling

Effective incident handling can mitigate the occurred incident and can aim to stop similar incidents in the future. Incident Handling consists of four stages:

1. Preparation: The effect of the incident could be minimized by (i) establishing checklists for incident handling, (ii) notification process for the incident, and (iii) activating risk assessment approaches for identifying incident response strategies.
2. Detection and Analysis: Each and every incident cannot be prevented using the preparation phase. The detection phase starts immediately after a suspicious activity has been reported by the people or by any tool. The analysis is performed to conform the claims made during the detection phase. Additionally, this also identifies the impact of the incident on the organization’s assets and services.
3. Incident Response (Containment, Eradication, and Recovery): The initial step over here is to reduce the extent of incident spread in the target environment. The attack vector is closed to eradicate the same or similar incidents. Finally, the affected system is rolled back to its latest safe state.

4. Post Incident: In this, postmortem analysis is conducted to understand the necessity of changing/updating the existing security policies. Thus, the chances of the intruder attacking the target system can be reduced.

Incident Handling approaches for various emerging domains like, Mobile Computing, Cloud Computing and IoT is being explored in the literature [7-10][14]. We classify the incident handling strategies according to the domains i.e. Traditional and Emerging (Figure 1.1). In this thesis, we focus on addressing the cloud incident handling issues due to its unique challenges in terms of multi-tenancy, lack of control and rapid elasticity. Also, the fact that other emerging technologies like, Mobile Computing and IoT directly or indirectly use cloud services. This motivated us to take up Cloud Incident Handling as our problem statement.

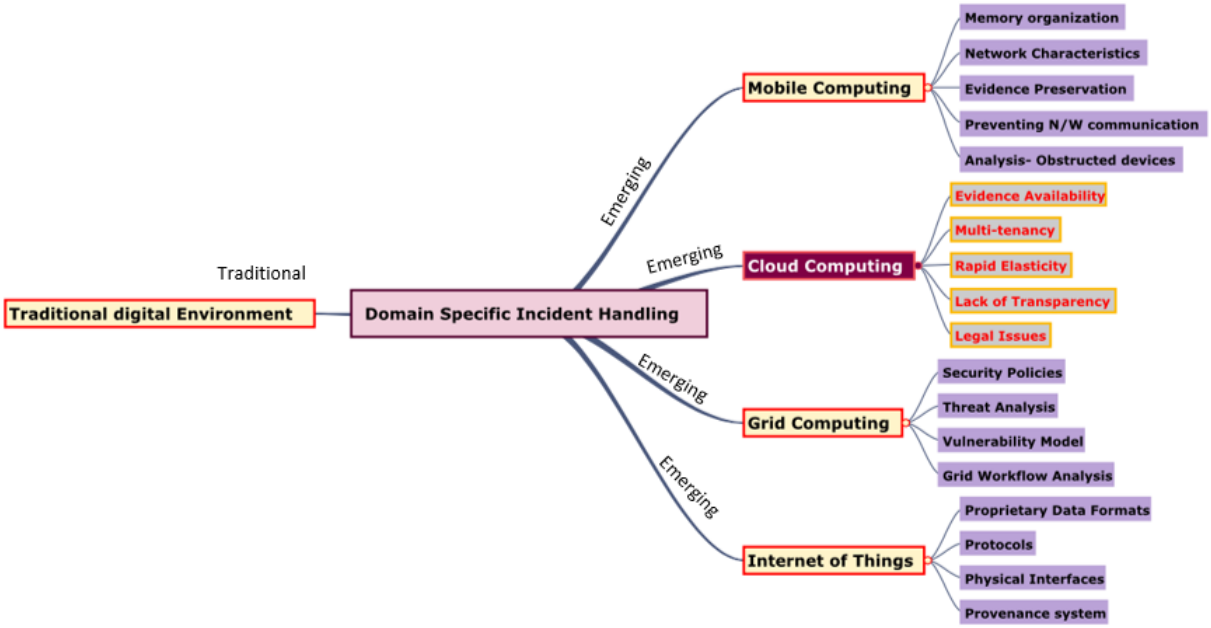



Figure 1.1: Domain Specific Incident Handling [7-10]

## 1.2 Incident Handling in the Cloud Environment

### 1.2.1 Cloud Computing

Technically the cloud paradigm evolved from grid computing. Several computers work parallely in the grid to solve an individual problem whereas in cloud, the Cloud Service Provider (CSP) delivers the end user with a unified service by leveraging multiple computing resources. Multiple definitions of cloud computing have been introduced in the literature [11][12] and the most widely accepted definition is given by the National Institute of Standards and Technology (NIST) [13].

**Definition**

*Cloud computing is a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.* [11]

The cloud model is composed of five essential characteristics:

1. On-Demand self service: The capabilities like, processing power, storage, and network usage can be unilaterally provisioned without human intervention.

2. Broad network access: All the computing capabilities can be consumed by the user via a network through heterogeneous thin or thick clients.

3. Resource pooling: The multi-tenant model of cloud can facilitate many cloud consumers to use its virtually available infinite computing resources through dynamic provisioning.

4. Rapid elasticity: The resources can highly scale outward and inward to satisfy the dynamic demands of the cloud consumers in terms of storage and computational capabilities.

5. Measured service: The metering capability can monitor, control and measure the cloud resources consumed by the user.

**Cloud Service Models and Deployment Models**

The cloud environment in general is referred to as XaaS (Anything as a Service) provider. The major service models encompassed in cloud include [12]:

- Software as a Service (SaaS): The applications at the cloud provider can be utilized by the cloud user without much concern about the maintenance overheads. The data, runtime environment, middleware, operating system, virtualization, servers, storage, networking, and individual application capabilities are controlled and managed by the cloud service provider (CSP). Examples include Concur, Google Apps, Citrix GoToMeeting, Workday, Cisco WebEx, and Salesforce.

- Platform as a service (PaaS): The consumer can create applications using the libraries, languages and tools supported by the CSP or can deploy the developed applications. The consumers of this service can have a control on the deployed application and data. The rest of the layers is managed by the CSP. Examples include Windows Azure, AWS Elastic Beanstalk, Google App Engine, and Heroku.

- Infrastructure as a service (IaaS): The resources like, processing, network, and storage can be provisioned by the consumer to install and run arbitrary softwares. The CSP

manages the cloud virtualization, servers, storage, and networking. The consumer can control the applications, data, runtime environment, middleware and operating system. Examples include Amazon Web Services (AWS), Microsoft Azure, Openstack, Cisco Metapod, Google Compute Engine (GCE) and Joyent.

The level of access varies based on the service model (Figure 1.2). Each service model can be configured with any one of the following deployment models [12]:

- Private cloud: It is generally provisioned and maintained by a single or group of organizations. Multiple business units of the organization can exclusively use the cloud infrastructure.
- Community cloud: Consumers with shared requirements in terms of security, policy and compliance prefer to use the community cloud. The cloud can be owned and managed by one or multiple organizations, a third party or some combination of them.
- Public cloud: The cloud is owned and maintained by a single or group of entities (academics, business or government). The cloud infrastructure is open to general public and capable enough to exhibit high rapid elasticity.
- Hybrid cloud: More than one cloud infrastructure is used to form hybrid cloud and accordingly it preserves the proprietary technology to achieve data and application portability.

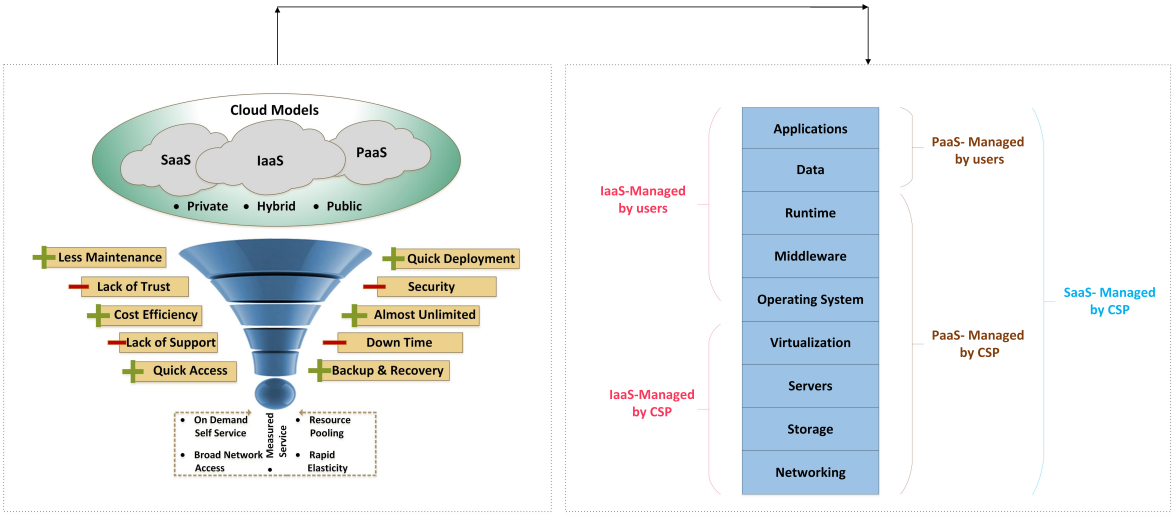



Figure 1.2: Cloud Service and Deployment Models

### 1.2.2 Significance of Cloud Incident Handling

Cloud computing is one of the most profound developments in the field of information technology. With the advent of this technology, the computational cost, software integration,

information access, and scaling of services were made easier and faster. A cloud service pools the resources and provides them across multiple tenants, delivers the resources without human intervention, monitors and measures the consumers resource usage accurately [1]. Moreover, it is accessible from heterogeneous thin or thick clients.

Market research media predicted an annual growth rate of 30 % for cloud computing and that this technology is expected to produce $270 billion by 2020. According to RightScale's cloud statistics of early 2016, almost 95 % of companies/respondents are using the cloud services for their business operations [15]. Moreover, the popular analyst firm Forrester estimated that for the year 2017, $463 billion will be spent on softwares among which $119 billion is solely spent in the domain of cloud computing [23].

On the contrary, cloud services are constantly being compromised by the intruders and the recent cloud attacks prove the same. A few examples of reported cloud incidents are: Amazon cloud was affected with a botnet in 2009 [17]. In 2013, hackers used the dropbox platform to perform advanced persistent threats (APTs) [18]. In October 2015, students at Worcester Polytechnic Institute hacked an instance in AWS which led to a data breach [19]. In 2016, Amazon was hit by Denial of Service and it was achieved by compromising the Domain Name System [20]. According to a survey conducted by [16], 56 % of cloud incidents are non-transparent. Similar attacks may continue in the future as well if the underlying isolation and cache management policies are not improved at both the hardware and software levels.

From the above, it is evident that the cloud services are not always secured. In reality, not every incident can be prevented. The incident handling in the cloud environment should be performed at the tenant level in a timely fashion such that the target cloud user's trust on the cloud environment should be improved. It is important note that, cloud incidents can happen at the cloud level and cloud tenant level and in this thesis, we focus on handling the cloud incidents at the IaaS tenant level.

### 1.2.3 Why Cloud Incident Handling is more challenging ?

In a traditional environment, the Incident Handler can reach the physical location for detecting, analyzing and eradicating similar incidents in the future. On the contrary, most of these activities are not directly applicable for Cloud Incident Handling (CIH) due to its distributed and rapid elasticity properties. This introduces many new challenges for cloud incident handling [21][22]. They are:

- Evidence Availability: During Cloud Incident Handling, the availability of the evidences is not always ensured as it is reactive in nature. Without the evidence, it is highly impossible to arrive at accurate logical findings about the occurred incident. This in turn would lead to inefficient containment and eradication strategies.

- Multi-tenancy: As the cloud systems facilitate the services to multiple tenants from the same infrastructure, it would be difficult to identify the events specific to the target

user. The evidence heterogeneity in the cloud would further complicate the incident handling.

- Admissibility of the evidence: The evidence collected should be legally admissible in the court of law. To achieve this, the evidence should be complete and its integrity should not be violated. This may not be always ensured due to the high dynamic nature of the cloud environment.

The other reasons to conclude that cloud incident handling is difficult are briefed below [21][24]:

- Service Logs which act as one of the major cloud evidences do not have unified format and this introduces difficulties in acquiring and analyzing them.
- Deleted data is a good source of evidence for Cloud Incident Handing. It is difficult to recover the deleted data from the cloud environment due to its dynamic resource allocation and reallocation feature. Additionally, attributing the deleted data to a specific user is still a challenge.
- Selective data acquisition is difficult as the Cloud Incident Handler may not know the complete details of the occurred incident.
- Cloud confiscation and resource seizure may affect the business continuity of the cloud.
- Cloud incident handling is highly challenging especially when the source of attack instance is terminated.
- Due to the rapid elasticity property of cloud, it is laborious to find the traces of an incident, especially when the process of Incident Handling is not initiated in a timely manner.
- In general, the size of the cloud evidences starts from gigabytes and so, locating the suspicious events related to the incident requires more time and human effort.
- Incident Handling on a live system may alter the state of the existing evidences and in effect, it makes the target entity to arrive at inaccurate logical findings about the occurred incident.

*Deduction:* We address the afore mentioned challenges of cloud incident handling using digital forensic science practices. In this thesis, the term *Incident Handling/Handler* refers to address various forensic challenges associated with the occurred cloud incident at IaaS VM or user level.

## 1.3 Using Digital Forensic Science Approaches for Handling Cloud Incidents

We use the Digital Forensic (DF) aspects to address the challenges of cloud incident handling. Also, using DF practices for CIH would increase the evidence availability for the incident handler.

### 1.3.1 Background on Digital Forensics

With the growing number of devices and threats, conducting forensic investigation gained the attention of security professionals as well as incident responders. The process to conduct digital forensic investigation was initially devised by FBI (Federal Bureau of Investigation) and other law enforcement agencies in early 1984. Later, many digital forensic models were developed from which an appropriate model could be chosen based on the investigator's requirement [25][26].

#### Definition

*Digital Forensics can be defined as, the use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation, and presentation of digital evidence derived from digital sources for the purpose of facilitation or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations [27].*

The common phases followed for performing forensic investigation in the digital environment are shown in the Figure 1.3 [28].

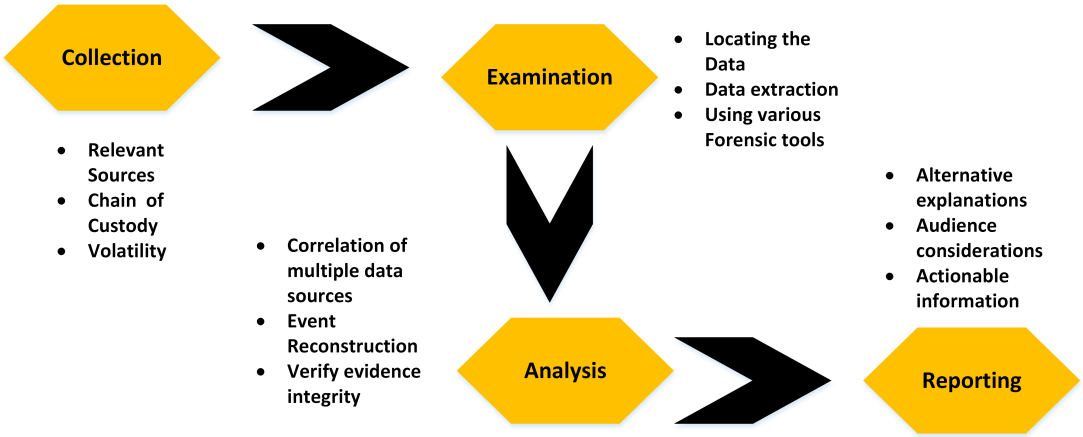



Figure 1.3: Digital Forensic Framework [28]

- Collection: The most relevant data sources pertaining to the incident should be identified and preserved without violating its integrity. Moreover, chain of custody should be followed to ensure legal admissibility of the acquired evidences.

- Examination: From each acquired data source, files/events/processes of interest are identified. This involves various activities like, using file viewers, uncompressing files, displaying directory structure, identifying known files and accessing file metadata.

- Analysis: Logical findings are drawn from the collected evidence. The data from multiple sources are correlated for quick interpretation. The integrity of the evidences should be verified at the end of the analysis phase.

- Reporting: The results from the above phases are presented in the form of a document. It also includes actionable information using which the investigator can acquire new evidences. Preparation of the report should be done keeping in mind the objective of the investigation and the purpose for which the analysis results are generated.

### 1.3.2 Using Digital Forensic approaches for Cloud Incident Handling

Existing works on cloud incident handling do not focus on evidence collection and analysis due to which several questions remain unanswered like: how the incident occurred ?, from where it originated ?, and what data was accessed/tampered with?.

There are a few research papers which discussed the integration of digital forensic aspects for improving traditional incident response capabilities [28][29][30]. Taking these as base, we suggest that the aspects of digital forensics can be used for handling cloud incident challenges. In CIH, evidence availability and its admissibility cannot always be ensured. So when the digital forensic aspects are used for cloud incident handling then the chances of evidence availability would increase due to the forensic readiness aspect in the digital investigation domain. The same is pictorially shown in the Figure 1.4. The forensic aspects which are used for cloud incident handling are discussed as below:

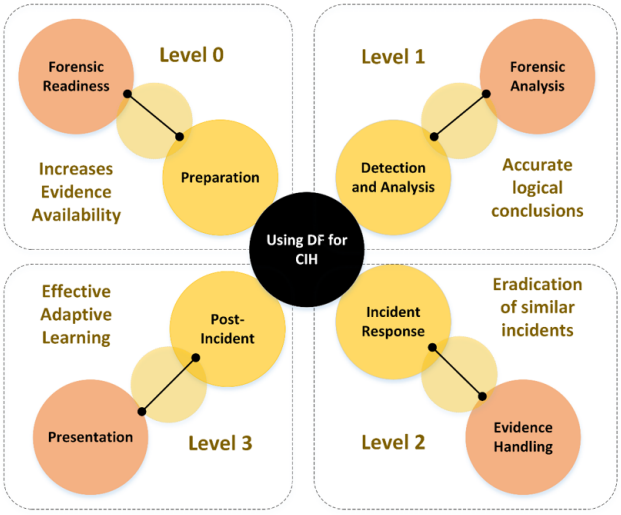



Figure 1.4: Using Digital Forensic (DF) Techniques for Cloud Incident Handling (CIH)

- Level 0 (Preparation phase of CIH uses Forensic Readiness aspect): Taking the forensic readiness perspective of digital forensics and applying that to the cloud incident handling would maximize the chances of evidence availability and helps the incident handler to improve the risk management strategies.
- Level 1 (Detection and Analysis of CIH uses Forensic Analysis practices): Immediately after the incident is detected, the incident handler will start through the selective evidence acquisition process such that its integrity is ensured. The forensic based incident analysis on the collected evidences is started so as to evaluate the impact of the occurred incident.
- Level 2 (Incident response of CIH considers Evidence Handling aspects): The assets and services of an organization that are affected with the incident are recovered. Inclusion of the evidence handling aspects makes the incident response effective.
- Level 3 (Post incident activity of CIH includes presentation phase of DF domain): The findings from the above three hybrid phases are included in the form of a report and presented to the court of law. The incident handler would go through the adaptive learning process based on the logical findings made from the previous incidents. This would finally improve the existing security policies of the target organization.

In this thesis, we address the challenges in Level 0 and Level 1. The reason is, addressing the technical aspects of these two levels would automatically help the incident handler to reduce the difficulties in Level 2 and Level 3. It is important to note that, we use forensic aspects for cloud incident handling to mainly increase the evidence availability. This can even be achieved by other proactive solutions. We only chose forensic aspects for performing incident handling because of its legal admissibility.

## 1.4 Challenges in using Digital Forensic Principles for Cloud Incident Handling

We identified that, applying digital forensic practices for cloud incident handling involves a lot of challenges. We organized them into three categories:

- Architecture (Diversity, Complexity, Data segregation, Multi-tenancy, etc.) [31][32][33]: This should deal with the variability in cloud architectures, building and maintaining provenance systems such that the chain of custody is preserved, compartmentalization of the on-demand resources, seizure of the cloud evidences without disrupting other tenants, etc. The approaches to handle each of these needs to be designed ensuring admissibility of the evidence.

- Analysis (Reconstruction, Anti-Forensics, Logs, Timelines, Correlation, Metadata, etc.) [34][35][36]: We need to design correlation approaches for analyzing the data or events of multiple artifacts of the same or different tenant(s). Handling anti-forensic techniques (countermeasures employed by the intruder to divert the forensic investigator) in cloud, reconstructing the events/data of the acquired artefact, analyzing cloud specific evidences like, service logs and snapshots, handling the issue of time synchronization especially when VM resources are spread across multiple locations are the issues which need to be considered for effective evidence analysis.

- Legal (Service level agreements, Subpoenas, Jurisdictions, Privacy ethics, etc.) [37][38]: There are various legal issues to conduct cloud forensic based incident handling. A few of them are, lack of international cooperation, missing terms in Service Level Agreements (SLAs) and lack of cooperation among the cloud providers. Also, without the proper knowledge of location of the physical resources, it is difficult to issue subpoenas.

Among the above, we focus on the major challenges in Analysis and Architectural categories. The multi-jurisdiction issues for cloud incident handling through forensics is not in the scope of our work.

## 1.5 Contributions of the Thesis

### 1.5.1 Objectives of the research

- Handling volatile evidences in cloud VM instances with trigger based introspection approaches

- Devising event reconstruction and provenance approaches for various IaaS VM artifacts for handling cloud incidents

- Designing event correlation approaches for various cloud artifacts

### 1.5.2 Scope and Assumptions

- Our emphasis is on solving major analysis and architectural challenges of forensic based incident handling at the IaaS tenant level.

- We test and validate the proposed cloud incident handling approaches using private IaaS cloud test bed.

- The cross border legal issues which might emerge during the forensic based incident handling are out of scope for this work.

- We assume that the Incident Handler (IH) gets the forensic support from various cloud actors like the Cloud Service Provider (CSP).
- We also assume that the target cloud environment is enabled with the Cloud Forensic Readiness (CFR) model(s).

### 1.5.3 Thesis contributions

The focus of this thesis is on addressing the major challenges in *Data Analysis* category by considering *Architectural* issues for handling the incidents at the IaaS user level.

1. Handling the volatile evidences: The thumb rule in forensic based incident handling is that the evidence acquisition should start from volatile artifacts. In reality, when we image the target virtual machine memory from the user level, it may undergo many changes which in turn can lead to inaccurate logical findings. The incident handler should perform live forensics without altering the evidences in the vRAM (virtual RAM) of the virtual machine which is possible through a technique called as Virtual Machine Introspection (VMI). VMI monitors the state of the virtual machine from the hypervisor level [39]. The CSP may not allow the incident handler to perform VMI on the target virtual machine as it requires hypervisor access. We handle this by proposing a model named ALTRA (Addressing Lack of Transparency) which addresses the issue of lack of transparency between the incident handler and the CSP.

   The incident handler can perform VMI to acquire memory events of the target virtual machine. It is important to note that, VMI is not a new technique to capture memory events and can have many delivery models to achieve the same [41]. We propose a trigger based introspection model for efficient capture of the target VM memory events. Finally, we propose improvisations for interpretation of the introspected data.

2. Devising approaches for event reconstruction and provenance: The evidences identified are segregated and transferred to the incident handler's environment for analysis. The incident handler selects the analysis technique based on the artifact and the incident type. In this work, we propose event reconstruction and provenance based analysis techniques and applied on cloud service logs and VM snapshots respectively.

   - Cloud Service logs: The cloud is managed by many inter-operable services. For each service there is an associated log created at the cloud node(s). We propose a model called SEASER (Applying Segregation, Aggregation on Cloud Service logs) using which one can arrive at effective hypothesis without violating the privacy of other users. The model addresses three major issues of service log analysis i.e.
     - Service logs contain events of multiple users and we propose approaches to automatically identify the events pertaining to the target user/instance.

  - All the service log events may not be of interest to the incident handler. We apply various supervised machine learning algorithms to automatically identify the events of interest to the incident handler.
  - In general, every service log contains millions of events and it is extremely time consuming to analyze all of those events. We address this problem through aggregation. We propose two new aggregation algorithms namely Leader-Follower Version 1 ($LF_{v1}$) and version 2 ($LF_{v2}$).

- Virtual machine Snapshots: Snapshots are the richest source of evidence for the forensic based incident handling. We address the major challenges involved in snapshot acquisition and analysis.
  - In reality, snapshots for a VM may not always exist. To increase their availability, we suggest the use of Cloud Forensic Readiness (CFR) models in which the snapshots are collected before the actual incident.
  - The captured snapshots have to be transferred to the isolated incident handler's environment via network and it may lead to the problem of data gravity (i.e. network and analysis overhead involved in cloud incident handling is referred as data gravity) [40]. We resolve this problem by designing a framework named SNAPS (Snapshots based Provenance System) which is derived from the existing spatio-temporal models and is then customized to suit the requirements of Cloud Incident Handling (CIH).
  - The acquired snapshots need to be analyzed and the proposed framework SNAPS achieve the same using provenance. Also, SNAPS can be used to address various forensic based incident handling challenges and we illustrate the same using VM snapshots acquired from the Openstack Cloud Environment.
  - On the other hand, using deleted objects for incident handling may give clues about the occurred incident. Retrieving them in the cloud environment is challenging and we address this using Natural Language Processing (NLP) techniques.

3. Performing Event Correlation: The term *event correlation* indicates or exposes the relation between two or more events. It can be used to gain higher level of knowledge from a huge number of events, to identify the faults and filter out the redundant, irrelevant and spurious messages, and make predictions about the future trends [41]. In comparison with the existing work on event correlation, our proposed correlation approach differs in the following aspects:

   - The first and obvious difference is, our event correlation approach is specific to the cloud environment and uses cloud specific artifacts like service logs for incident handling.

- We correlate events effectively from all the major evidences available in the target VM and the specific artifact to be investigated vary depending on the case under consideration.
- Most of the existing correlation approaches deal with identifying the relations among the events present in the same artifact. We extend the correlation capabilities to identify the associations among the events present in multiple artifacts. We also discuss the correlation approaches in both incident - known and unknown cases.

The proposed approaches for handling cloud incidents would address many other cloud architectural and analysis challenges. The same is shown in Figure 1.5.

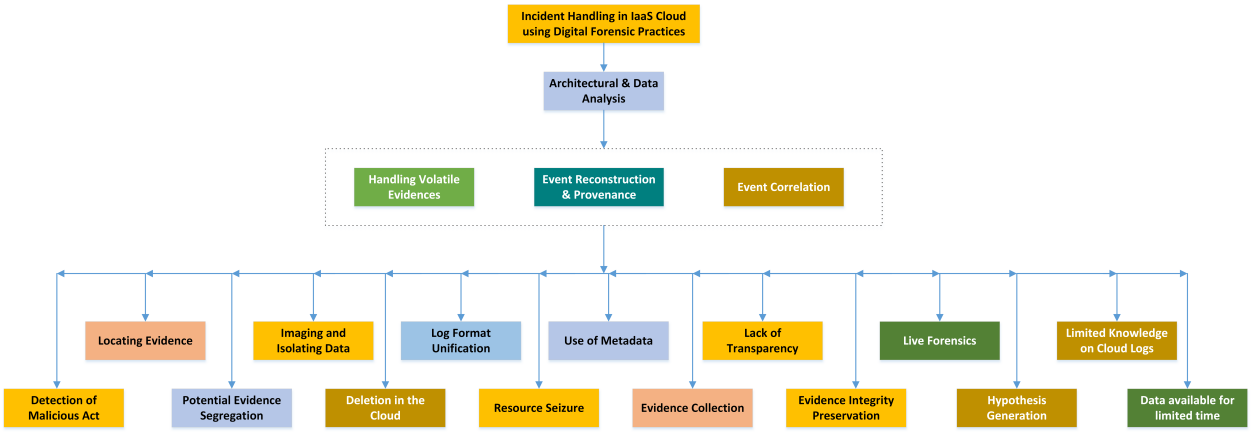



Figure 1.5: Objectives of the Research

## 1.6 Organization of the Thesis

The thesis comprises seven chapters and the outline of each is described below.

- Chapter 2 discusses various incident handling and digital forensic models and their current research thrust areas. We discuss the challenges, current solutions and open issues of using digital forensic science practices for cloud incident handling.
- In chapter 3, first we describe the proposed hybrid introspection approach which can handle volatile evidences of the target instance. The hybrid approach is proposed by taking the analogy of logic analyzer. To identify the root cause of the incident from the captured introspection events, we use Complex Event Processing (CEP). We add a new module to the introspection flow process and this populates the events of forensic relevance which can help the incident handler reduce the time spent on analyzing the volatile evidences.

  The second part deals with the proposed model which can address the lack of transparency between the incident handler and Cloud Service Provider (CSP). We discuss

the proposed approaches - SeMS (Sequence Mining on user Sequences) and CoPS (Conditional Probability on user Sequences) to detect suspicious events from the target application logs. Finally, we validate both the approaches using a typical investigative scenario.

- Chapter 4 describes the identified role of cloud service logs during incident handling. We then discuss various approaches for segregating the target user events from other users. We detect the events of incident handler relevance using various supervised machine learning algorithms. We discuss the proposed aggregation approaches for service log analysis and we validate them using Openstack cloud logs. Finally, we discuss the identified application of the proposed approaches.

- Chapter 5 discusses the proposed approach to increase the availability of the virtual machine snapshots to the incident handler. Further, we handle the issue of data gravity using spatio-temporal models. We discuss the proposed provenance system for snapshot analysis. Finally, we recover the deleted objects from the VM snapshots using NLP techniques.

- Chapter 6 presents a framework for performing cloud event correlation. Event Correlation can be homogeneous (same artifact is used) or heterogeneous (different artifacts are used) and each of these can occur with the incident-known and incident- unknown cases. Based on this, we derived four cases. In this work we address the following three cases: Homogeneous-incident unknown based correlation, Heterogeneous-incident known based correlation and Heterogeneous-incident unknown based correlation.

- Chapter 7 summarizes the entire work with a discussion on the future scope.

An integrated model of our contributions is shown in Figure 1.6.

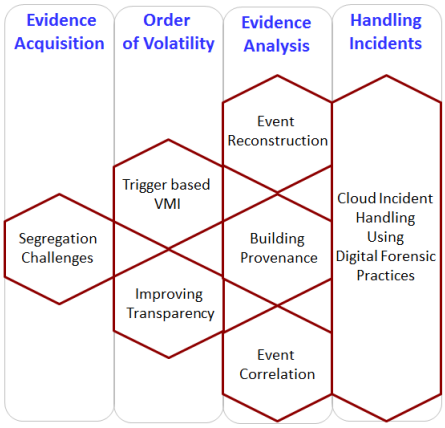



Figure 1.6: Integrated model of our contributions

## 1.7 Summary

The way data is stored and processed has changed with the advent of cloud computing. It facilitates virtually infinite computing resources using which an intruder can compromise the resources of other tenants/users. Evidence availability plays a significant role in effective incident handling. Using digital forensic science principles for handling cloud incidents ensures evidence availability. Performing forensic investigation in the cloud environment involves various Architectural, Analysis and Legal challenges. In this work, we address the major issues in the Architectural and Analysis categories.

In this chapter we introduced the challenges of cloud incident handling and gave an insight into the cloud incident handling using digital forensic practices. In the next chapter, we discuss various incident handling and digital forensic models and highlight the open research issues for cloud incident handling even after using digital forensic practices.

# Chapter 2

# Background and Related Work

*"Going Forward, It's a mobile first, cloud-first world"*

-Microsoft

As discussed in chapter 1, Cloud Incident Handling using the forensic aspects helps in better containment, eradication and recovery of the incident. In this chapter, we discuss the capabilities of Incident Management and Incident Handling in the traditional digital environment. We then describe the challenges and existing solutions involved in IaaS based cloud incident handling. Finally, we discuss the open forensic problems for the same.

## 2.1 Incident Handling in Traditional Digital Environment

Incident Management comprises many activities like, handling vulnerabilities, conducting training programs to increase security awareness, managing events and generating alerts [45]. The taxonomy for security incident management is shown in Figure 2.1. Incident Handling (IH) falls under the category of security incident management.

Incident Handling can be performed in multiple ways based on various guidelines and standards given by the most popular security agencies/organizations (Table 2.1) and the same are briefed below:

- *Computer Emergency Response Team/Coordination Centre* (CERT/CC): Detailed guidelines to handle security incidents are discussed by [44]. It contains various phases namely, incident alert/report, triage, incident response, analysis. All these phases include 14 sub-phases for incident handling.

- *National Institute of Standards and Technology* (NIST) is a renowned standards laboratory and it suggested a four phase Incident Handling process which includes preparation, detection and analysis, incident response and post incident activity [43]. The

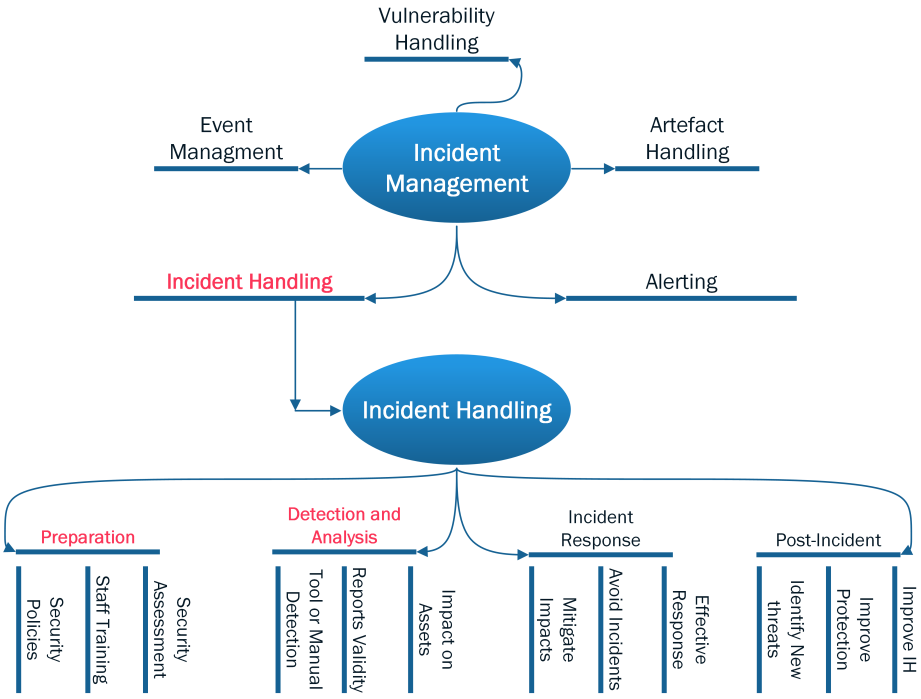



Figure 2.1: Taxonomy of Security Incident Management and Incident Handling (Compiled from [42][43])

second and third phases are conducted iteratively for better containment and eradication of the occurred incident.

- *International Organization for Standardization* (ISO): This suggested five phases for Incident Handling which comprises planning and preparation, detection and reporting, assessment and decision making, response and lessons learned [47]. These phases can also be used for managing the vulnerabilities and security events.

- *European Network and Information Security Agency* (ENISA) : This is a European agency which guides the organizations to meet the network security and information requirements [46]. It suggested a six phase process for incident handling which includes phases namely, incident report, registering the report, triage, incident resolution, incident closure, and post analysis.

In this thesis, we follow the process suggested by NIST and apply it for cloud incident handling. It is important to note that, incident handling models which we find today have close similarity with the NIST incident handling guidelines [49][50].

## 2.2 Cloud Incident Handling and its Research Challenges

Governments of many countries have started focusing on cloud incident handling as they realize about the dangers of intruder actions [52]. Till now, Cloud Incident Handling is not fully understood and explored due to its inherent characteristics like distributed, rapid

Table 2.1: Summary of Incident Handling Models

| Incident Handling Standards and Guidelines | | |
|---|---|---|
| **CERT/CC** [44] | **BIP 0107:2008** [42] | **ENISA** [46] |
| Reporting and detection | Incident detection and recording | Incident report registration |
| Triage | Classification and initial support | Triage |
| Analysis | Investigation and diagnosis | Incident resolution |
| Incident response | Resolution and recovery | Incident closure post-analysis |
| **ISO/IEC 27035:2011** [47] | **SANS (Kral, 2011)** [48] | **NIST SP 800-61** [43] |
| Plan and prepare | Preparation | Preparation |
| Detection and reporting | Identification | Detection and analysis |
| Responses | Containment, Eradication and Recovery | Containment, Eradication and Recovery |
| Lessons learned | Lessons learned | Post-incident activity |

elasticity and multi-tenancy [53]. This inturn makes the incident recovery difficult and time consuming. For instance, on February $28^{th}$ 2017, AWS suffered an outage incident that lasted for many hours and they could not immediately find the appropriate reasons behind it. Incidents in cloud are frequently reported and unfortunately, 56 % of them are non-transparent [55].

In the traditional incident handling, the experts can shutdown the system immediately after the incident. In cloud, this is not possible and requires a new strategy to handle the incidents. There are only a few papers which explored the possibilities for cloud incident handling and the same are briefed below:

- The challenges of cloud IH are presented using five phases namely, detection, analysis, containment, eradication and recovery, and continuous improvement [1].

- The authors in [21] concluded that, at the high level, the cloud IH is same as traditional IH. The way we perform each phase of the incident handling in the cloud is different when compared with the traditional IH process. This is due to its unique challenges like, multi-tenancy, lack of transparency and having no physical access.

- The authors discussed the new challenges in the domain of cloud security and then suggested that, there should be more cooperation among various cloud actors to deploy policies and procedures as a monitoring solution. Since there exist no proper policies for cloud incident handling, it can lead to privacy violation and thereby affect the availability of the cloud services [56]. For example, the FBI during its investigation on fraud detection powered off the whole data center in Texas and it affected the data availability of many users [51].

The extensive literature review on cloud incident handling has helped us compile a comprehensive taxonomy of the challenges as depicted in Figure 2.2 [57][58][59].

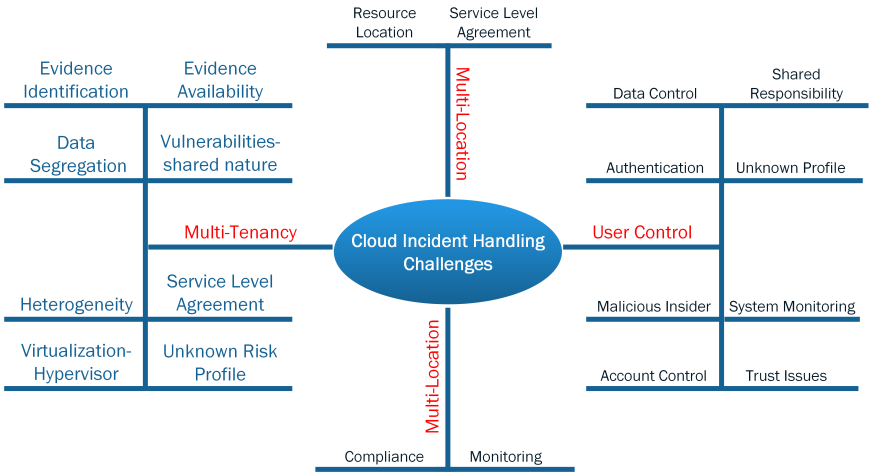



Figure 2.2: Cloud Incident Handling Challenges

*Deduction*: Cloud Incident Handling in its present form responds to the incidents without emphasizing on the relevant evidence collection. Without this due consideration, recurrence of the incidents cannot be prevented. We use the digital forensic practices for cloud incident handling to address these aspects.

## 2.3 Using Digital Forensic aspects for Effective Cloud Incident Handling

Since we use Digital Forensic (DF) principles as a methodology for effective cloud incident handling, we initially discuss the existing DF models. We discuss the challenges of applying the digital forensic practices for cloud incident handling.

### 2.3.1 Digital Forensic Models

In Digital Forensics (DF), computer and other digital media crimes are investigated through a scientific process to answer several forensically relevant questions pertaining to the occurred incident. Apart from identifying and imaging the digital evidence, it can attribute the digital evidences to exhibit and produce them in the court of law.

In the early $21^{st}$ century, various cyber-security bodies have proposed several guidelines to investigate the digital crimes. Some of the popular digital forensic models are summarized in Table 2.2. Using these models and forensic tools, the FBI is successfully investigating thousands of digital cases reported every year in the USA [60].

Table 2.2: Summary of Digital Forensic Models

| Guidelines for Performing Digital Forensic investigation | | |
|---|---|---|
| **Martini** [61] | **Choo et al.** [62] | **Quick et al.** [63] |
| | Commence Prepare and response | Commence Prepare and response |
| Evidence source identification and preservation | Identification and collection | Evidence source identification and preservation |
| Collection | | Collection |
| Examination and analysis | Preservation Analysis | Examination and analysis |
| Reporting and presentation | Presentation Feedback Complete | Reporting and presentation Feedback |

The investigative agencies solved many interesting cases over the years using digital forensic principles and practices like, Dennis Reader [64], Scott Tyree [65], Brad Cooper [66] and James [67]. Recently, Clinton's *Bombshell* case was defused using digital forensics [68]. Also, Cyber Forensics division in India solved many interesting cases using the standard digital forensic models [69]. In this thesis, we take the DF model proposed by Quick et al., as a base and use it to address the challenges of the cloud incident handling.

It is important to note that, using digital forensic aspects for handling cloud incidents would not solve all the challenges of it (CIH). The same are discussed in the subsequent sections.

### 2.3.2 Cloud Incident Handling Challenges: A Forensic Perspective

Handling cloud incidents using digital forensic science differs from one service model to another as the level of penetration and evidence availability significantly vary.

#### Cloud Incident Handling in (SaaS) vs (PaaS) vs (IaaS) - A forensic perspective

Forensic based Cloud Incident Handling for each service model of cloud is briefed below:

- *SaaS* (Eg - Salesforce, Cisco WebEx, Citrix GoToMeeting, etc.)[13][70]: The Cloud Provider owns all the layers (Servers, Storage, Networking, Virtualization, Operating System, Middeleware, Runtime, Data, Application) due to which customers' maintenance cost is significantly reduced. SaaS applications leave very limited traces at the client end. The incident handler should rely on the logs stored at the cloud side and that detailed information acts as a valuable source of evidence during forensic based incident handling.

- *PaaS* (Eg - Apprenda, Heroku, Google App Engine, etc.)[13][71]: The customers use the middleware components to develop their own applications. A full control on the

application layer will be with the customer including the application dependencies. Due to this, the customer can create and maintain a detailed application log which can definitely aid the incident handler.

- *IaaS* (Eg - Amazon Web Services, Microsoft Azure, Openstack, etc.)[13][72-77]: In IaaS cloud, the virtual machines will be managed (location, scaling, security etc.) by the cloud provider to satisfy the consumer requirements. The layers - Application, Data, Runtime, Middleware, Operating System will be under the control of consumer; the virtualization and hardware will be controlled by the cloud provider.

From the above discussion, it is evident that, the level of access control to the target user differs from one service model to the other and among which, IaaS users will have more penetration to the cloud resources. This may in turn increases the possibility of performing an incident/suspicious activity by the intruder. In this thesis, we focus on addressing the challenges in forensic based incident handling in IaaS environment.

### 2.3.3 Incident Handling in IaaS Cloud Environment using Digital Forensic Practices

It is important to note that, even though cloud forensic investigation follows the same sequence of phases as traditional digital forensics, the process behind each forensic phase differs in cloud due to its unique characteristics. This introduces several new challenges during forensic based incident handling and we brief all of them phase wise as below:

*Identification Phase:* In this phase, the incident handler searches for *likely value* evidences pertaining to the occurred incident and the challenges involved in it are:

- Logs as a source of Evidence: In general, the traces of cloud incident spreads across multiple logs (Service logs, OS logs, Firewall logs, Network Logs, etc.) and then automatically identifying the exact source of evidence still remains a challenge [78]. In [79], the authors suggested logging guidelines (what, where, and when to log) to aid the cloud incident handler. In [80], the authors developed their own logging mechanism in Eucalyptus cloud to easily identify forensically relevant events. In [81], the authors devised an encrypted logging model and the logs generated using this model are transferred to the central server in the encrypted format. All these logs will be completely under the control of the consumer. The common drawback while implementing the above cloud logging solutions is, the existing IaaS environment of the CSP should undergo significant changes and it may not be feasible in reality.

- Volatile evidences [61][82]: Focusing on volatile evidences like, vRAM can help the incident handler find forensically valued information (user names, passwords, encryption/decryption keys, etc.) and then properly performing volatile memory analysis gives quick logical findings about the occurred incident. The vRAM evidence

may undergo a few changes during its imaging from the cloud virtual machine and in this case, the modified evidence diverts the incident handler to generate inaccurate inferences.

*Data Collection and Preservation Phase:* Once a VM in IaaS cloud gets compromised, its identified evidences should be acquired without violating their integrity so that the chances of legal admissibility would increase. From the literature, we identified that there are five possible ways of acquiring the evidences from the IaaS cloud infrastructure [78][79] i.e.

- Use the existing tools [83]: The standard traditional forensic tools like, Encase, FTK, and Memoryze can be used to acquire vDisk/vRAM of the target virtual machine. The main drawback with this approach is, remote acquisition through these tools may not be trustworthy and cannot be legally admissible. Moreover, these tools cannot acquire the cloud service logs (without violating privacy) and snapshots of the target cloud instance (virtual machine).

- Use Trusted Platform Modules (TPM) [84]: TPMs can be installed on the cloud nodes so that the hardware in each node can know the health status of the hosted guest virtual machines. Once a TPM based hardware finds the suspicious activity in a certain VM then its evidences can be preserved securely. The main drawback associated with the TPM based acquisition is, there are chances that a VM process can be altered without being known to the TPM modules. Moreover, till now, no cloud provider hosted their virtual machines in the TPM based hardware. In the future, we expect that customers may show keen interest on a cloud provider who has TPM modules integrated in to the hardware.

- Management plane based acquisition: Cloud services are received by the end users through the management plane. Using this, the incident handler/user can acquire the evidences pertaining to the target VM [85]. The drawbacks associated with this type of acquisition are, all the possible evidences cannot be acquired through management plane. Also, this may open up a new attack surface and may not always ensure reliable evidence collection.

- Forensics-as-a-Service (FaaS) [86]: The cloud provider collects the required evidences and gives them to the target user/incident handler. Since the cloud provider controls the complete infrastructure, the evidences gathered through this may be legally admissible. The drawbacks associated with this type of acquisition are, there is no standard protocol devised for facilitating FaaS to the end user, the response time is high and the CSP may not monitor the target VM events at low level and so preserving evidences is not possible in all the cases.

- Legal Solutions: Issuing legal warrants to the CSP and asking him to provide the evidences of the affected virtual machine only works when laws satisfying multiple jurisdictions are framed and executed [83]. Till date, no cloud provider has announced its support for the law enforcement.

Depending on the context (type of incident, cloud provider policy, legal, acts etc.,), any one of the above acquisition approaches can be chosen. Before which the following acquisition related issues need to be considered and handled by the incident handler i.e.

- Preserving privacy [83][87]: Due to the multi-tenancy nature of cloud, it can host evidences of several users. During evidence collection, the evidences pertaining to the target user should be identified and isolated. If the evidence segregation is not done properly then it leads to the privacy violation of other users. In [87], the authors proposed three isolation techniques namely Instance relocation, Server farming and Sandboxing. These approaches lacks practical feasibility.

- Ensuring data integrity [89]: Preserving the integrity of the evidence increases the admissibility of the findings made through the forensic examination and analysis. There is a likelihood of integrity errors being generated due to the involvement of multiple cloud actors in the forensic investigation.

- Lack of thorough knowledge [92][93]: Since cloud is relatively a new and emerging area, the digital forensic experts may not have detailed knowledge about the data organization in the cloud, its architecture, policies, etc. This indeed leads to improper negotiation with the CSP and needs special training.

- Chain of custody [91]: It is the process describing how the evidence is collected, preserved and analyzed so that the admissibility of the collected evidences will be strengthened in the court of law. In cloud, ensuring chain of custody is challenging due to its multi-layered and distributed architecture.

*Analysis Phase:* Conducting forensic based incident analysis on the cloud evidences is very challenging. The two major reasons for it are - (i) Volume of the cloud evidences is generally very high when compared with the traditional digital environment and each evidence contains a vast number of objects/events. Analyzing all those is challenging especially in time critical investigations. (ii) The data exported to the cloud is internally stored in cloud's native format and it is difficult for the incident handler to analyze them. The other issues to be handled during this phase are:

- Lack of specialized tools: Due to rapid elasticity of cloud,its distributed nature and lack of transparency properties, the existing digital forensic tools cannot be applied to the cloud environment. There is a huge demand for dedicated cloud forensic tools to enable the incident handler perform forensic investigation [92]. In [94], the authors

conducted a survey to know the expectations from cloud forensic tools and 58 % of respondents replied that, automatic evidence analysis approaches (traditional/cloud) should be developed. The applicability of various forensic tools to the cloud environment is assessed and the same is shown in the Table 2.3.

- Correlating the evidences gathered from multiple sources [95][97]: An incident trace may spread across multiple artifacts and analyzing all of them to understand the associations among the events is challenging in terms of time and human effort involved. There are no proper evidence correlation approaches devised to handle the cloud incidents. The event correlation among the evidences will give a comprehensive picture of the occurred incident and may lead to quick appropriate findings. Unfortunately, the challenges of performing effective cloud event correlation still remain unaddressed.

- Maintaining provenance for cloud evidences [88]: Provenance stores the history of each object present in the target evidence. It is a proactive approach and intruder can easily disable the provenance system deployed at VM level. Moreover, for a basic user, storing provenance at object level adds to the space and cost overheads.

- Time Synchronization [90]: In forensic based incident handling, time related metadata can act as a crucial source of evidence. Generally, the cloud evidences spread across multiple systems which are having different time zones. The issue of time synchronization is especially challenging in the public cloud environments.

- Reconstructing the crime scene [91]: Knowing the sequence of suspicious activities performed by the intruder is possible through Event Reconstruction. Reconstruction of the crime scene is not always possible especially when the intruder restarts/terminates the virtual machine after the incident. In [96], the authors suggested that, event reconstruction in this case could be possible by considering snapshots of the virtual machine. Incident reconstruction may not be effective when the events in the evidences are incomplete and the chances of that happening in the cloud are more.

Table 2.3: Summary of the existing forensic tools

| Existing Forensic Tools | | |
|---|---|---|
| Tool | Use | Cloud/General Based |
| Encase Remote Agent [83] | Remote Acquisition | General |
| FTK Remote Agent [83] | Remote acquisition | General |
| X-Ways [98] | Live system based acquisition for Windows and Linux systems | General |
| Sleuthkit [99] | Can analyze forensic images of the hard disk | General |
| E-discovery Suite (Encase) [100] | Can investigate the network or computer offline | General |
| FTK imager [101] | Can image the memory and disk of the target system | General |
| Snort [80] | Network intrusion detection system for Windows and Linux systems | General |
| Wireshark [102] | Analyzes the to and fro network packets of the target system | General |
| OWADE [103] | User visited websites and can know whether they contain any data stores in the cloud | Cloud Based |
| FROST [104] | Acquisition tool integrated into the Openstack management plane | Cloud Based |

## 2.4 Current Solutions and Open Issues for Cloud Incident Handling using Digital Forensic Practices

Focusing on the challenges in each phase of cloud incident handling using forensics principles, researchers proposed several solutions and the same have been summarized in the Table 2.4.

Table 2.4: Summary of the existing cloud forensic solutions

| Existing Solutions | | |
|---|---|---|
| **Phase and challenges** | **Proposed Solution/Approach** | **Ref.** |
| **Identification** | | |
| Access to the evidence | Eucalyptus framework (OS and logs) | [80] |
| | Extraction of the status of the relevant data | [105] |
| | A log based model | [106] |
| | An advanced logging model embedded with encryption | [24] |
| Dependence on the CSP | | |
| Trust Issue | Layers of trust model | [83] |
| Data acquisition | TrustCloud | [107] |
| Compliance | Cloud Management Plane | [85] |
| Logs | Service Level Agreements (SLAs) | [108] |
| Volatile Data | Cloud Persistent Storage | [105] |
| | API based continuous synchronization | [109] |
| **Preservation and Collection** | | |
| Data Integrity | Trusted Platform Modules | [83][84] |
| Time Synchronization | Unified time system | [105] |
| Cloud literacy of incident handlers | Improving the skills of incident handlers | [93] |
| Chain of Custody | Trained staff | [92] |
| Ephemeral nature | Snapshot acquisition and analysis | [38] |
| Distributed storage | VM location identification through instance tagging | [72] |
| **Analysis and Examination** | | |
| Lack of standard forensic tools | FROST, OWADE | [103][104] |
| Lack of log framework | Suggested complete log management system | [83][110] |
| Evidence Time lining | A partial solution by AWS Trail | [111] |
| Encrypted Data | Cloud key management infrastructure | [112] |
| **Presentation** | | |
| Technical comprehension of jury | Training | [89] |
| Jurisdiction | Cross border Laws and Legal Agreements (Eg: MLAT) | [113] |
| Chain of custody | Well defined principles and guidelines (Eg: ACPO- UK) | [114][115] |

**Open Issues in Forensic based Cloud Incident Handling**

Handling cloud incidents using digital forensic science aspects is relatively a new methodology and has lot of issues which still need to be addressed and the same is shown in Table 2.5. Each of those issues is discussed below:

Table 2.5: Major unaddressed Issues in Cloud Incident Handling using Forensic Principles

| S. No. | Gaps Identified for Cloud Forensic Investigation in IaaS |
|---|---|
| 1 | Lack of standard and dedicated tools |
| 2 | Evidence correlation and Timeline analysis from multiple sources |
| 3 | Cross border Issues |
| 4 | Automatic analysis of cloud evidences |
| 5 | Handling the volatile evidences of the target virtual machine |
| 6 | Crime scene reconstruction |
| 7 | Technical comprehension of the jury |

1. Lack of standard and dedicated tools [92][103][104]: From the Table 2.3, it is evident that only two tools can perform forensic investigation in the cloud environment. There are a lot of drawbacks for these tools i.e.

   *FROST:* This is a forensic tool that was integrated into the Openstack cloud environment. It can acquire the evidences from the content management plane without interacting with the guest OS. The drawbacks associated with FROST tool are, it only works in Openstack cloud and cannot be applied to other cloud platforms, it requires significant changes in the Openstack cloud, it does not provide analysis capabilities for the acquired evidences, and it assumes the trust from multiple entities (CSP, provider's infrastructure etc.).

   *OWADE:* This tool was developed by the researchers in Stanford university and was launched in BlackHat Conference. It can store the websites visited, can track most of the internet activities, can search the online identities, etc. OWADE is still in development and only the basic version was released with the minimal features. The major drawback associated with this tool is, it can only track the web related data and cannot acquire the cloud specific artifacts like, Service logs and Snapshots. In addition to this, the tool capabilities for analysis need to be improved.

   From the above, it shows the necessity for developing a comprehensive cloud forensic tool which should identify the distributed evidences, acquire all of them by employing proper preservation techniques and analyze (basic, advanced) the preserved evidences.

2. Evidence correlation and Timeline Analysis [95][97]: Timeline analysis can sequence the occurred events using which the incident handler can know various details like, when, where and how an event had generated. If the events from multiple sources are considered for timeline analysis then several issues need to be handled like, how to interpret the events and how the events with different logging structure can be represented in a timeline without much data loss. Additionally, the cloud incident handler should correlate the events in several evidences and identify the events pertaining to the target incident. This correlation helps reduce the time for detecting and analyzing the occurred incident.

3. Cross Border Issues [37][38]: Due to the distributed nature of cloud, the evidences of victim/intruder virtual machine may spread to multiple locations. This gets worse, when multiple evidences from different jurisdictions need to be acquired. Cross border issues hinder the process of developing new approaches for cloud forensic investigation. This shows the importance and necessity for framing global laws for conducting forensic investigation in the cloud environment.

4. Automatic analysis of Cloud evidences [34][35][36]: The acquired cloud evidences will have its own format. Consequently, this introduces new difficulties during the analysis phase. For instance, the vDisk acquired from the Openstack cloud environment is in QCOW2 format and it cannot be analyzed by the existing digital forensic tools. In addition to that, the cloud evidences will be of huge size and time taken to analyze them is generally very high. New cloud forensic techniques should be developed to automate and reduce the time and human effort required for cloud evidence analysis.

5. Handling volatile evidences of the target instance [82][109]: The basic principle in forensics is to start investigation in the order of volatility. Acquiring volatile evidences is always not possible as the intruder may restart/terminate the cloud virtual machine after the occurrence of the incident. New approaches need to be devised to capture the events from volatile memory of the target instance. Moreover, these approaches should have analysis capabilities using which the incident handler can quickly know the intricacies of the occurred incident.

6. Crime scene Reconstruction [91][96]: Reconstructing the crime scene can make the incident handler to know about how, when, and where an incident occurred. There are no existing approaches or algorithms for performing cloud event reconstruction. Additionally, this introduces new difficulties for the incident handler when the events from any of the evidences are missing.

7. Technical comprehension by the jury [89]: The findings from the cloud forensic examination and analysis should be presented as a report in the court of law. The report

should contain many details like, crime scene location, resources used by the victim/intruder, and legally admissible analysis format. To know all these information from the cloud is would be challenging owing to its complex architecture. Comprehension of the technical findings by a non-technical jury member adds much more complexity to the entire presentation stage.

## 2.5 Summary

In this chapter, we initially discussed various Incident Handling and Digital Forensic models. We discussed all the major forensic challenges in IaaS environment. We then described the existing solutions and open issues for handling IaaS cloud incidents using forensic aspects. In this thesis, among all the above open forensic problems, we address the following issues:

- Handle volatile evidences of the target cloud VM
- Devise effective event reconstruction and provenance approaches for various cloud instance artifacts
- Perform correlation among the cloud virtual machine evidences

Chapter 3 discusses the proposed approaches for handling volatile evidences by addressing the lack of transparency between incident handler and the CSP.

# Chapter 3

# Handling Cloud VM's Volatile Traces by Improving their Availability

*"Cloud is about how you do computing, not where you do computing"*

- VMware

## 3.1 Introduction

The thumb rule in forensics based incident handling is to start investigation in the order of volatility. One of the most volatile evidences in the virtual machine/instance is vRAM. It contains valuable information like running process information, encryption keys, open files for each process, unpacked versions of a program, memory resident malwares, user names and passwords, and network connections. The main challenge while considering the vRAM for incident handling is:

*"After the incident, it is difficult to have the traces in the cloud virtual machine's vRAM as the intruder may restart/terminate the virtual machine"*.

Our objective is to ensure the seamless availability of the vRAM evidence and analyze it for handling cloud VM incidents. Accordingly, the chapter is divided into two parts and the same are briefed below:

1. Proactively capture vRAM events from the target user cloud virtual machine and we use Virtual Machine Introspection (VMI) technique to achieve this. VMI is a process used to monitor and analyze the state of the virtual machine from hypervisor level. We discuss the approaches proposed for analyzing the captured vRAM events.

2. Capturing events through VMI requires the incident handler to access the hypervisor which also contains the content of other users. Intentionally or unintentionally, the incident handler should not access the data of the non-target users as it violates the privacy of those users. It is the responsibility of the existing cloud actors (specifically, CSP) to make sure to track the activities being performed by the incident handler at the hypervisor level. Accordingly, the motivation for the second part is as follows.

   We propose a model called ALTRA (Addressing lack of Transparency) which can record all the activities performed by the Incident Handler. If the incident handler performs any suspicious activity then the ALTRA model will automatically alert the Cloud Service Provider (CSP). The CSP can take necessary action on the incident handler.

*Organization of Part 1:* In Section 3.2, we propose a design for effective introspection by taking the analogy of logic analyzer. In Section 3.3, we discuss various introspection rules from the identified categories along with their structural representations. In the same section, we propose a mechanism based on static call graphs to increase the incident detection capabilities from introspection data. We propose an approach for root cause identification using Complex Event Processing (CEP) and the same is discussed in Section 3.4. In section 3.5, we present the experimental results.

## 3.2 Proposed Trigger based Introspection Model for Cloud Instance Incident Handling

There are two main challenges in capturing the virtual memory of a cloud instance from virtual machine level. (i) After the incident, the target virtual machine may not be in the *on* state and so the virtual memory cannot be imaged (ii) Even though, the virtual machine is in the running state, imaging the memory or collecting the information about the target modules can change the state of the existing system and so the acquired evidences may not be legally admissible.

To handle the above, a new method of monitoring and analyzing the state of a virtual machine from hypervisor is introduced and it is named as Virtual Machine Introspection (VMI) [116]. A major advantage of VMI is, the contamination of the acquired virtual memory events is very less [117]. Moreover, VMI is a proactive technique which can ensure the availability of vRAM data and can increase the reliability of the overall forensic based incident handling process [118][119]. There are three types of delivery models for VMI namely *in-band delivery*, *derivation based* and *out-of-band delivery* [120].

- In-Band Delivery: Guest software in the virtual machine provides the semantic knowledge to the introspection module at VMM.

- Derivation-Based Model: Semantic data generation component derives the information based on the hardware architecture.
- Out-of-band: The semantic knowledge of the guest OS is supplied to the introspection module from external entity.

There are certain libraries to perform VMI. For example, LibVMI [121] supports Xen and also KVM. VMI-PL [120] is another monitoring language that has additional features in comparison with the LibVMI like, introspecting the data stream events. This library suffers with performance and stability issues.

Since the monitoring is done outside the virtual machines, the semantic knowledge pertaining to each VM may not be known and this can badly affect the data collection and extent of analysis [122][123]. This indeed leads to various advantages and disadvantages during the process of introspection. The same are summarized and shown in Table 3.1.

It is evident from Table 3.1 that, a single delivery model may not be sufficient in all the cases [124]. For example, if a target virtual machine is affected with memory resident malware then the data given by in-band delivery may not be comprehensive and reliable as it modifies the events in memory. Thus hybrid introspection approaches have been projected as a viable alternative. The technique, though, comes with its own issues.

### 3.2.1 Drawbacks of existing hybrid introspection approaches

Our intention is to take the aid of introspection for performing memory forensics based incident handling on virtual machines. The hybrid introspection techniques in the current state cannot serve our purpose as they are [124]:

- Unaware of the scenarios for proper capture of the events in accordance with time
- Unable to identify the effective combination(s) of hybrid model based on the context

Our emphasis is on the first issue. For this, we start with redesigning the concept of introspection technique. Based on the extensive interdisciplinary study we have done, the broad functionality of *logic analyzer* suits our requirement. We discuss the use of logic analyzer for designing an effective introspection approach in the following subsection.

### 3.2.2 Proposed trigger based introspection model using logic analyzer

Logic analyzers are electronic devices used for digital measurements containing numerous signals [125]. It is used for debugging the hardware in prototype digital systems. When a logic analyzer is connected to the digital circuit, the *trigger module* in it continuously monitors the logic state of the signal and sends out an alert when the defined signal pattern has been observed. This motivated us to use the concept of logic analyzer for devising

Table 3.1: Benefits and drawbacks of various introspection models (Out-band, In-band, Derivation)

| **S. No** | **Delivery Model** | **Advantage(s)/Disadvantage(s)** |
|---|---|---|
| 1 | Out-of-Band | *Advantages*<br><br>• Compromised VM will have less affect on the out-of-band based introspection.<br>• The guest OS may not need to run during the time of interpretation by introspection module.<br>*Disadvantages*<br>• The non-binding nature of the out-of-band delivery model cannot have precise analysis when the guest software was changed or replaced. |
| 2 | Derivation Based | *Advantages*<br><br>• The model can work independent of the guest OS since the binding is from the hardware level.<br>• Malicious entity cannot change the virtual hardware architecture, so the external view generation function is not affected.<br>*Disadvantages*<br>• It is difficult to retrieve more relevant information by monitoring the hardware state. |
| 3 | In-Band Based | *Advantage*<br><br>• There is no semantic gap at all<br>*Disadvantages*<br>• When the guest OS functionality is compromised then the semantic knowledge given by it will not act as a reliable source of information<br>• The semantic data generation component cannot function when the virtual machine is powered off<br>• Some contents gets modified/erased by which the reliability of the evidence(s) is questionable. |

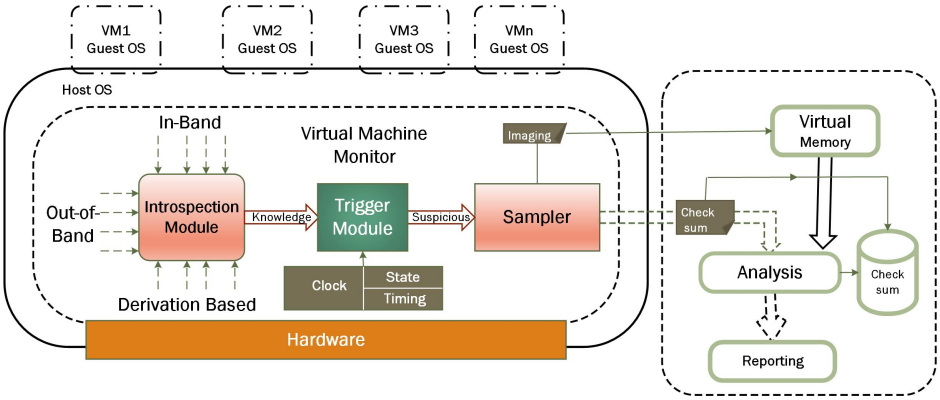



Figure 3.1: A trigger based introspection process in cloud

trigger based introspection approach. The modules of the proposed model are shown in Figure 3.1 and each of them is briefed below:

*Introspection module:* This module is a facilitator for all the three delivery models. It is the incident handler's choice to pick up the required delivery model(s) to form the hybrid model using which the data from the targeted virtual memory is collected.

*Trigger Module:* The trigger module is the main component of our proposed model. The introspected virtual memory events will be piped through this module. The events which satisfy the introspection rules mentioned in the trigger module will be stored and analyzed first. In the following sections, we will elucidate the challenges involved in generating introspection rules for memory events.

*Sampler*: After the specified triggers are satisfied, the basic functionality of the sampler would be to take the image of the target virtual memory using VMI and perform the analysis. Instead of taking the entire image, selective acquisition of virtual memory regions based on the occurred incident can also be done. For example, if the trigger module at the hypervisor identifies some suspicious events of a Linux VM after it is connected with the external drive then the Common Internet File System (CIFS) regions can only be captured and analyzed instead of capturing the whole VM memory.

*Analysis and reporting:* Correlating the data from the collected artifacts and knowing what exactly has happened to the targeted virtual machine is the primary goal of this module.

The trigger module enhances the relevant data collection pertaining to the incident by invoking the corresponding rule set. We do not discuss explicitly about the functionality of the introspection module as there is a lot of work which describes the internals of this module [122][123]. On other side, adding our trigger module to the introspection process will have significant benefits which are listed and justified accordingly in Section 3.5. We have used the Out-of-Band and In-Band Models of hybrid introspection and put emphasis on the virtual memory events like, process events and system call events.

## 3.3 Rule and Graph based Approaches for Trigger Module

Irrespective of the delivery model(s) chosen, we identified that the VMI should handle the following issues:

- The rate of introspection events of virtual memory is very high, especially in highly virtualized environment like cloud and finding the events of interest is difficult and time consuming.
- In most of the cases, the generated introspection data is difficult to interpret as it is not semantically organized to reflect the real incidents.

We address the first issue with the aid of our trigger module whereas the data interpretation is done using other components-CEP engine and break points (we will discuss all these components in the rest of the chapter).

### 3.3.1 Rule based approach for introspection

The main idea in addressing the first issue is adding the rules to the trigger module. When a defined rule gets satisfied with the captured introspection events, then those events can be treated as suspicious and correlating them would be sufficient to have a logical conclusion about the occurred incident. For representation of the rules we use more formal notations [126] and they are summarized in the Table 3.2.

Any combination of the above operators can occur like, events occurring at the same location and one after the other is denoted as $<>$; and events occurring at different locations and concurrently is denoted as $><||$, etc. Existing rule language framework(s) do not focus on events occurring at the same location. We even consider this in our rule generation process. We have formulated a few rules based on the literature and the important ones are described below:

*Rule-1:* UPX $<>$; Section_headers = 0

This rule comes under the spatio-temporal category which indicates that an obfuscated malware may use some packers like UPX and can remove the section header details. These events happen one after the other and on the same process by which we used $<>$; operator.

*Rule-2:* a. Access to GOT $<>$ strycpy / memcpy $<>$ Overwrite Symbol Addresses

b. Access to GOT $<>$ setuid/setgid $<>$ Escalating privileges

The above rule (a) says that when there is an access to Global Offset Table (GOT) such that there is a change in the symbol address with unprivileged access then the trigger module at hypervisor confirms that a malware is trying to overwrite the symbol addresses. Rule (b) says that, when there is an access to GOT with a modification of UID then it indicates that malware is trying to escalate privileges. Similar things can happen when there are modifications in the Process Linkage Table (PLT).

*Rule-3:* Access to /sys $<>$; Access to /kernel $<>$; ASLR=0

Table 3.2: Notations used for introspection rule generation

| S. No | Category | Notion | Comment |
|---|---|---|---|
| 1 | Spatial | $E_x <> E_y$ | Events happening at the same location |
| 2 | Spatial | $E_x >< E_y$ | Events occurring at remote or distinct locations |
| 3 | Temporal | $E_x$;$E_y$ | Events occurring one after the other in sequence |
| 4 | Temporal | $E_x \| \| E_y$ | Events occurring at the same time or concurrently |
| 5 | Temporal | $A(t_x, E_y, t_z)$ | Occurrence of an event $E_y$ will be triggered during the time period of $t_x$ and $t_z$ |
| 6 | Temporal | P($E_x$, T($E_y$), $E_z$) | Occurrence of $E_y$ for every time period of T and this starts when $E_x$ has occurred and stops until $E_z$ happens. |
| 7 | Temporal | P*($E_x$, T($E_y$), $E_z$) | Once $E_x$ has occurred then the readings are taken for every time period of T and the cumulative results are submitted once $E_z$ has happened. |
| 8 | Temporal | $\neg\{t_x, E_a, t_y\}$ | Non-occurrence of an event between the time intervals of $t_x$ and $t_y$ are recorded. |
| 9 | General | $\bigwedge$ , V | Conjunction and disjunction operators |
| 10 | General | ANY(n, $E_x$, $E_v \cdots E_t$) | Occurrence of n events from t events i.e $\|t\| > = \|n\|$ |
| 11 | General | ALL($E_x$, $E_v \cdots E_t$) | Triggering will happen when all the t events occurred |

There will be a huge and wide variety of information in *procfs* and *sysfs* file systems and intruders try to generate rootkits in these file systems. These rootkits can violate the integrity of the system as the data from these file systems are used by many utilities.

This rule falls under the category of Spatio-Temporal rule set and it says that, all the three events are accessing and modifying the entities in the same file system */proc* by following a certain sequence i.e. initially, rootkits access the */sys*, then the */kernel* file system is accessed and finally the Address Space Layout Randomization (ASLR) is made zero using which the trigger module confirms the suspicious activity.

*Rule-4:* P(select kernel process, $t_x$[DTB], Identify malicious process)

This rule helps the trigger module to identify whether the selected process is a malicious user process or not. For this, it checks the Directory Table Base (DTB) value and if this value is non zero for the kernel process then it indicates that the process is actually user based but trying to blend itself as kernel process.

*Rule-5:* Any (1, (E1) V (E2) V (E3)) where

- E1: Getting the details and modifying the shadow files (contains passwords hashes)
- E2: Malicious process after stealing of keystrokes has to be stored in some files like key loggers before they are actually sent to the target system through the network.
- E3: Also, malware can use the xmodulepath package to gain the root privileges

Any access/modification to confidential files acts like an input to the trigger module as they aid in the process of incident handling.

*Rule-6:* $Absence\_INTERP, \neg\{Dynamic\_Linker\_Funct\}, Packer$

The trigger module alerts the investigator when there is an absence of INTERP header and presence of the packer during which if dynamic linker is malfunctioning then it can be treated as suspicious under obfuscated malware.

*Rule-7:* Access_to_VM(e1) $><$; mount_to_host(e2) $><$; Any_newConnection

Whenever there is an entity downloaded from the virtual machine and mounted on the host machine then that is treated as suspicious by the trigger module especially, if it is behaving unusually like, establishing a new network connection and sending details to the target system/effecting the host system.

*Rule-8:* Access(ANY [(tmpfs) V (/dev/shm)]) $<>$ Modification/ Deletion

There are certain memory resident malwares using which the artifacts are made visible only in temporary file system (/tmpfs) of virtual memory. Intruders prefer to hide the data by storing the related content in the /tmpfs and the trigger module focuses on this too.

These rules are specific to VMs with Linux guest OS. The rules can still be applied to other guest operating systems with few modifications. We use complex event processing engine to generate automatic rules using both the positive and negative traces (discussed in the subsequent sections). The advantage of these sort of rule generation is the accuracy involved in detecting the specified introspection events. Irrespective of any number of rules

generated, a variation of those suspicious events cannot be detected by the trigger module. To overcome this, we add the feature of pattern matching to the trigger module and the details are explained in the following subsection.

### 3.3.2 Graph based approach for effective interpretation of introspection events

We identified that detecting the incidents with the rules especially in the context of introspection is difficult as the intruder can use $n$ possibilities. In reality, knowing all those possibilities and writing the rules accordingly may not result in increasing the detection accuracy if:

- We are not able to capture some events during introspection
- Variation of suspicious events are captured in comparison with the ideal possible suspicious sequences

To address these issues, we draw a static call graph. It is a graph depicting all the possible paths for the occurrence of an incident. For example, we identified different paths for performing shell code injection. The corresponding static call graph is drawn and the same is shown in Figure 3.2. To accomplish shell code injection, the intruder follows four generic steps:

- Handle is created with the target process (using ptrace).
- Within the target process, both writable and executable memory regions have to be identified (one of the ways is to use mmap)
- Once the memory region is found, shellcode is written into it (using poketext or process_vm_writev)
- The inserted shell code is made to execute (using ptrace_getregs, ptrace_setregs and ptrace_con).

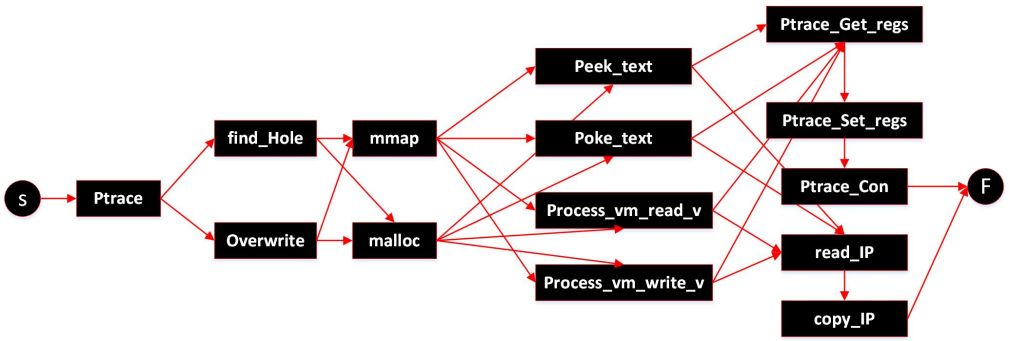



Figure 3.2: Shell code injection in Linux based virtualization environment

Figure 3.2 shows the granularity of system call level. From the captured events, we draw the introspection graph and correlate that with the existing static call graph(s). This helps us to know whether the incident occurred or not and also, the path followed to accomplish the corresponding incident. Detailed analysis and its results are discussed in Section 3.5.

## 3.4 Root Cause Analysis through Complex Event Processing

We discussed the approaches involved in detecting the suspicious events/incidents. Detection alone is not sufficient for most of the incident handling challenges [127]. We add the feature of basic root cause identification to the trigger based introspection. To accomplish this, we use complex event processing (CEP) approach. CEP systems analyze large flows of primitive events received from a monitoring environment to timely detect composite events (CE) [128]. Every event trace can be a positive or a negative. A trace is said to be positive when the defined composite event has been satisfied else it is a negative trace.

### 3.4.1 Existing work on CEP

To the best of our knowledge, the research on CEP has focused only on the processing efficiency of complex events. The issue which has taken less attention is, how to devise mechanisms for generating effective rules. In general, knowing sequence of events and the relative events with the information they carry may not be known. This difficulty is addressed by our approach.

### 3.4.2 Proposed architecture for root cause analysis targeting effective introspection of VMs

The challenges involved with CEP in knowing the sequence of events can be addressed by the process of introspection. Until now, CEP techniques have focused only on the positive traces for automated rule generation [128]. We devise our approach by considering the negative traces as well so that the root cause of the occurred incident can be identified. The proposed process is depicted through the architecture in Figure 3.3.

Break points are generic, composite events and provide them to act as input to the CEP engine. In the introspection process, we consider each set of events with atleast one primary key as an individual trace. Some examples of break points:

- Gaining root privileges by accessing the packages like *xmodulepath* at bash shell
- Observing changes in current shell path
- Frequent accessing of password hashes from shadow files

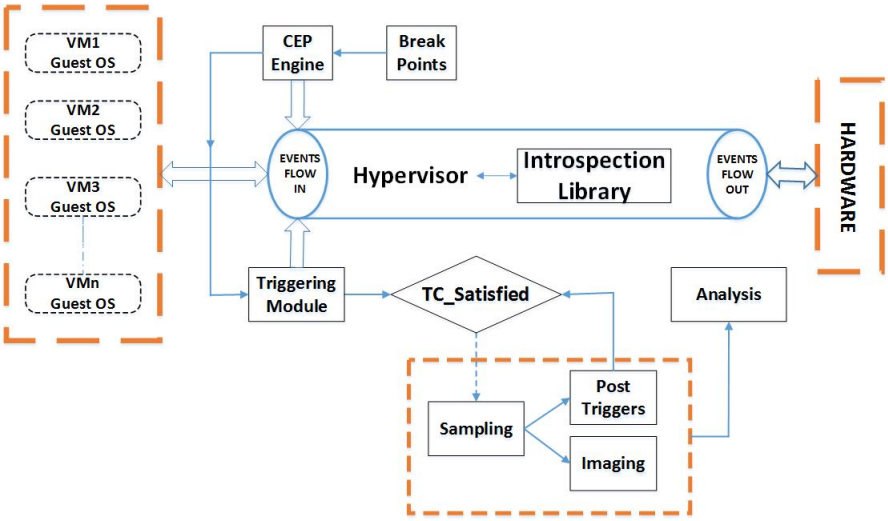



Figure 3.3: Proposed architecture for root cause identification using CEP

- Handle creation by a process targeting another process

These are a few break points pertaining to Linux operating system and according to the incident handler requirements, there is scope of developing new break points.

Once the event trace under consideration satisfies the break point(s) then the CEP engine follows the process in Algorithm 1 to identify the root cause of the incident. This knowledge is given as an input to the trigger module for new and effective rule generation which finally helps in detecting the previously unknown incidents.

**Algorithm 1** Root cause identification using complex event processing

1: $S_i[]$ = { $B_\alpha$, $B_\mu$, $B_\gamma$...until t} ▷ $1 <= t <=$ number of break points
2: **procedure** FLOW_EVENTS($E_1$,$E_2$....$E_t$) ▷ Captured through Introspection process
3: **for all** t break points in $S_i$ **do**
4: **if** $S_i$ == *true* **then**
5: $R_i[\,] \leftarrow$ *Store*($S_{i-1}$, $S_i$)
6: $E_{pk_i}[\,] \leftarrow$ *findPrimary_key*($T_{S_i}$)
7: $K_\alpha \leftarrow$ Identify_NT in $R_i$ based on $E_{pk_i}$
8: $P_\beta \leftarrow$Search_NT in other buckets based on $E_{pk_i}$
9: **end if**
10: **end for**
11: event_aggregation($R_i$,$K_\alpha$, $P_\beta$ )
12: **end procedure**

*Explanation:* Based on the break points, the CEP engine can identify, store and process the events in two levels.

- Level-1(#L1): Identify the event trace and store at composite event level.
- Level-2 (#L2): Accumulate the stored traces for better correlation.

#L1: Define the break points before the actual introspection process starts ($S_i$[]). The CEP engine checks for the break point satisfaction in each trace. Once it is satisfied, we store all the event traces starting from the previously satisfied break points ($S_{i-1}$) to the current satisfied break point ($S_i$), irrespective of trace type-positive or negative.

#L2: Initially, identify the key $E_{pk_i}$ for the satisfied composite event trace $T_{S_i}$. Based on $E_{pk_i}$, search for negative traces (NT) in the same bucket where the composite event is satisfied. The same negative trace search is applicable for other buckets also. This search results in identifying the positive and negative traces based on the key of satisfied break point. All these filtered traces ($R_i$,$K_\alpha$, $P_\beta$) are aggregated and correlated to get the root cause of the incident.

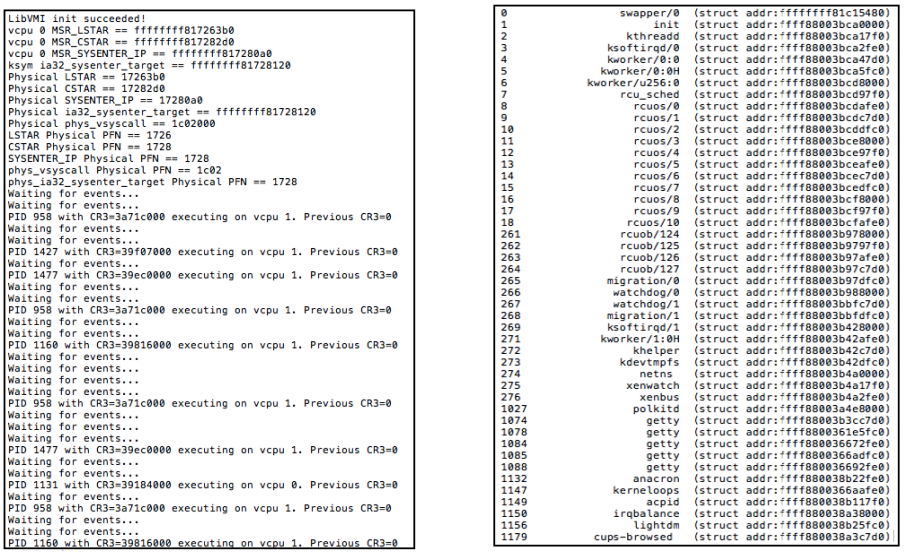
Once the new trigger conditions (TC) are satisfied then incident handler can still continue and enhance the analysis by invoking the sampler module. This module will provide the facility of taking the image of virtual memory from hypervisor level. Moreover, the process of introspection can still be continued after the incident and can check for consequent events happening in relation with the incident (post triggers). Once these post conditions are satisfied then re-invoking sampler module is required for taking further necessary action.

## 3.5 Evaluation of our Work

We have done extensive experiments to increase the effectiveness of introspection by reducing the semantic gap that was part of the existing approaches. We installed LibVMI which is an introspection library in host OS with Linux 14.04 server and deployed all virtual machines with Ubuntu 14.04 servers. Initially, we ran the existing examples of memory events in the LibVMI library. The corresponding observations are:

- Figure 3.4(a) gives pid and CR3 details (when paging is enabled, the processor uses CR3 register to find the page table directory for the current process and it is finally used for converting linear addresses into physical addresses of the corresponding process)
- Figure 3.4(b) shows pid, pname and structure address details of each process
- The common deductions from both:
  - Semantic interpretation of introspection data is difficult and may not help the incident handler in all the cases.
  - We observed that the rate of generated introspection events is very high and this can make the job of incident handler difficult especially in finding the events of interest.

We address these issues with the designed scenarios and the same are discussed below:




(a) (b)

Figure 3.4: (a) CR3 events (b) User and Kernel Process Structure details

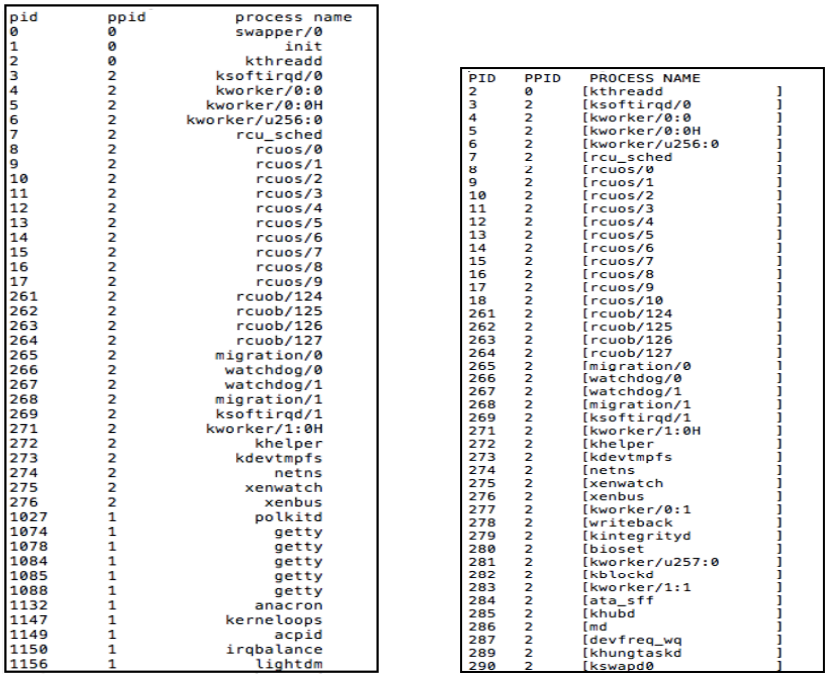
*Scenario-1:* We populated high level information of each process using VMI. Figure 3.5(a) depicts parent id, child id, process name and these details enhance the semantic interpretation of introspection events. For example, if we know the white list of the processes, then identifying suspicious processes becomes easier. Moreover, if a parent process is suspicious then finding all of its children can be done automatically. This can reduce the time invested in the analysis phase of forensic based incident handling.

*Scenario-2:* When there is a kernel-based malware in the target virtual machine's memory then the incident handler wants a list of kernel processes in less time. This can be achieved easily from VM level but getting the similar information using VMI is difficult. We achieved this by accessing the required kernel data structures of the target VM. The same is shown in the Figure 3.5(b).

From the above, it is evident that detecting and analyzing the events that satisfied the rules configured in Trigger module helps the incident handler to generate accurate logical findings. When there is a variation of known incident, this rule based approach may not detect the incident. This issue was discussed theoretically using a graph based approach in Section 3.3.2 and it is validated in the following subsection.

### 3.5.1 Detecting the variation of known incidents

In reality, the intruder may create a variation of the known incident by which the chances of being caught would be reduced. To detect these sort of incidents, we introspected the event traces at the system call level. In this context, there are three possibilities: (1) Introspected system call trace of the target process can follow the exact sequence with one of the paths in static call graphs. (2) The captured system calls of the target process can be incomplete as all the events may not be properly introspected. (3) The introspected system call sequence



| pid | ppid | process name |
|---|---|---|
| 0 | 0 | swapper/0 |
| 1 | 0 | init |
| 2 | 0 | kthreadd |
| 3 | 2 | ksoftirqd/0 |
| 4 | 2 | kworker/0:0 |
| 5 | 2 | kworker/0:0H |
| 6 | 2 | kworker/u256:0 |
| 7 | 2 | rcu_sched |
| 8 | 2 | rcuos/0 |
| 9 | 2 | rcuos/1 |
| 10 | 2 | rcuos/2 |
| 11 | 2 | rcuos/3 |
| 12 | 2 | rcuos/4 |
| 13 | 2 | rcuos/5 |
| 14 | 2 | rcuos/6 |
| 15 | 2 | rcuos/7 |
| 16 | 2 | rcuos/8 |
| 17 | 2 | rcuos/9 |
| 261 | 2 | rcuob/124 |
| 262 | 2 | rcuob/125 |
| 263 | 2 | rcuob/126 |
| 264 | 2 | rcuob/127 |
| 265 | 2 | migration/0 |
| 266 | 2 | watchdog/0 |
| 267 | 2 | watchdog/1 |
| 268 | 2 | migration/1 |
| 269 | 2 | ksoftirqd/1 |
| 271 | 2 | kworker/1:0H |
| 272 | 2 | khelper |
| 273 | 2 | kdevtmpfs |
| 274 | 2 | netns |
| 275 | 2 | xenwatch |
| 276 | 2 | xenbus |
| 1027 | 1 | polkitd |
| 1074 | 1 | getty |
| 1078 | 1 | getty |
| 1084 | 1 | getty |
| 1085 | 1 | getty |
| 1088 | 1 | getty |
| 1132 | 1 | anacron |
| 1147 | 1 | kerneloops |
| 1149 | 1 | acpid |
| 1150 | 1 | irqbalance |
| 1156 | 1 | lightdm |

(a)

| PID | PPID | PROCESS NAME |
|---|---|---|
| 2 | 0 | [kthreadd ] |
| 3 | 2 | [ksoftirqd/0 ] |
| 4 | 2 | [kworker/0:0 ] |
| 5 | 2 | [kworker/0:0H ] |
| 6 | 2 | [kworker/u256:0 ] |
| 7 | 2 | [rcu_sched ] |
| 8 | 2 | [rcuos/0 ] |
| 9 | 2 | [rcuos/1 ] |
| 10 | 2 | [rcuos/2 ] |
| 11 | 2 | [rcuos/3 ] |
| 12 | 2 | [rcuos/4 ] |
| 13 | 2 | [rcuos/5 ] |
| 14 | 2 | [rcuos/6 ] |
| 15 | 2 | [rcuos/7 ] |
| 16 | 2 | [rcuos/8 ] |
| 17 | 2 | [rcuos/9 ] |
| 18 | 2 | [rcuos/10 ] |
| 261 | 2 | [rcuob/124 ] |
| 262 | 2 | [rcuob/125 ] |
| 263 | 2 | [rcuob/126 ] |
| 264 | 2 | [rcuob/127 ] |
| 265 | 2 | [migration/0 ] |
| 266 | 2 | [watchdog/0 ] |
| 267 | 2 | [watchdog/1 ] |
| 268 | 2 | [migration/1 ] |
| 269 | 2 | [ksoftirqd/1 ] |
| 271 | 2 | [kworker/1:0H ] |
| 272 | 2 | [khelper ] |
| 273 | 2 | [kdevtmpfs ] |
| 274 | 2 | [netns ] |
| 275 | 2 | [xenwatch ] |
| 276 | 2 | [xenbus ] |
| 277 | 2 | [kworker/0:1 ] |
| 278 | 2 | [writeback ] |
| 279 | 2 | [kintegrityd ] |
| 280 | 2 | [bioset ] |
| 281 | 2 | [kworker/u257:0 ] |
| 282 | 2 | [kblockd ] |
| 283 | 2 | [kworker/1:1 ] |
| 284 | 2 | [ata_sff ] |
| 285 | 2 | [khubd ] |
| 286 | 2 | [md ] |
| 287 | 2 | [devfreq_wq ] |
| 289 | 2 | [khungtaskd ] |
| 290 | 2 | [kswapd0 ] |

(b)

Figure 3.5: (a) Identifying suspicious processes (b) Identifying kernel based malware processes

may not exactly match with any one of the paths in the corresponding static call graph.

In all the above possibilities, we suggest that incident detection is possible through pattern matching algorithms. For illustration, we considered shell code injected vRAM of the target virtual machine. Each introspected event trace of system calls at the process level (Figure 3.6) is correlated against the pattern in the paths of the corresponding static call graph (Figure 3.2). Finally we identified that, in the process id 1417, the shell code was injected using the below sequence:

$$pid(1417) : ptrace\_set\_regs \rightarrow ptrace\_get\_regs \rightarrow process\_vm\_write\_v \rightarrow sys\_mmap$$

The logical finding from this approach is, the incident handler can know the sequence of steps followed by the intruder to accomplish an incident. In reality, root cause analysis would further help the incident handler to answer many forensically relevant questions and can increase the event interpretation of the target system. Taking its forensic relevance, we discuss that in the following subsection.

### 3.5.2 A scenario depicting root cause analysis

Scenario: *Identifying the fake binaries*

We detect the fake binaries using complex event processing.

- Fake binaries are generally placed in the temporary file system so that detection is not possible once the virtual machine is turned off. Figure 3.7 shows the identified fake

| Pid | System call |
|---|---|
| 1153 | sys_poll |
| 1417 | sys_mmap |
| 1456 | sys_ioctl |
| 1673 | sys_access |
| 1217 | sys_mmap |
| 1673 | sys_shmat |
| 1346 | sys_bind |
| 1346 | sys_listen |
| 1417 | process_vm_write_v |
| 1457 | sys_fork |
| 1278 | sys_access |
| 1345 | sys_ioctl |
| 1457 | sys_pipe |
| 1417 | process_get_regs |
| 1321 | sys_getittime |
| 1278 | sys_poll |
| 1417 | ptrace_set_regs |
| 1468 | sys_shmat |
| 1135 | sys_access |
| 1642 | sys_semop |

Figure 3.6: Introspecting the system calls for OpenStack VM

| | | | |
|---|---|---|---|
| 1132 | 1 | rmmod | /sbin/rmmod |
| 1147 | 2 | vim | /usr/bin/vim |
| 1149 | 2 | ls | /bin/ls |
| 1150 | 1 | cat | /bin/cat |
| 1156 | 1 | df | /bin/df |
| 1179 | 1 | insmod | /sbin/rmmod |
| 1215 | 1156 | rm | /tmp/rm |

Figure 3.7: Identified fake binary path

binary with the aid of full path and the incident handler can identify that, */tmp/rm* is a fake binary created by the pid=1215 as the original path should be */bin/rm*.

We added this break point to the CEP engine. Our CEP engine uses Esper which is one of the popular complex event processing softwares [133]. When the defined break point is satisfied, it generates a warning along with time. We identify the primary key of the fake binary event trace (i.e. pid). Once the key is known, then we search other buckets for event traces with the same primary key value (here pid=1215). A screenshot depicting the same is shown in Figure 3.8.

```
1 WARN: Event of interest-fake binaries
2 Time: Tue Mar 7 12:07:16  CEST 2015
3 Pid 1215 is the primary key for the current trace
4 Performing primary key based search...
5 Performing primary key based search...
6 Performing primary key based search...
7 Storing all the identified event traces: Time Tue Mar 7 12:07:59 CEST 2015
```

Figure 3.8: Alert generation using CEP

| PID | PPID | PROCESS NAME |
|---|---|---|
| 1 | 0 | init |
| 531 | 1 | upstart-udev-br |
| 537 | 1 | systemd-udevd |
| 666 | 1 | upstart-socket |
| 835 | 1 | upstart-file-br |
| 849 | 1 | rsyslogd |
| 854 | 1 | dbus-daemon |
| 920 | 1 | ModemManager |
| 939 | 1 | systemd-logind |
| 941 | 1 | bluetoothd |
| 958 | 1 | avahi-daemon |
| 963 | 958 | avahi-daemon |
| 970 | 1 | NetworkManager |
| 1027 | 1 | polkitd |
| 1074 | 1 | getty |
| 1078 | 1 | getty |
| 1084 | 1 | getty |
| 1085 | 1 | getty |
| 1088 | 1 | getty |
| 1132 | 1 | anacron |
| 1147 | 1 | kerneloops |
| 1149 | 1 | acpid |
| 1150 | 1 | irqbalance |
| 1156 | 1 | lightdm |
| 1179 | 1 | cups-browsed |
| 1215 | 1156 | xorg |
| 1218 | 1 | accounts-daemon |
| 1230 | 1 | cron |
| 1247 | 1 | getty |
| 1263 | 1 | whoopsie |
| 1277 | 1156 | lightdm |
| 1292 | 1277 | lightdm-greeter |
| 1300 | 1 | dbus-daemon |
| 1301 | 1292 | unity-greeter |
| 1305 | 1 | at-spi-bus-laun |
| 1309 | 1305 | dbus-daemon |
| 1313 | 1 | at-spi2-registr |
| 1317 | 1 | gvfsd |
| 1321 | 1 | gvfsd-fuse |
| 1332 | 1 | dconf-service |
| 1342 | 1156 | lightdm |
| 1346 | 1 | init |
| 1348 | 1 | nm-applet |
| 1351 | 1346 | indicator-messa |
| 1352 | 1346 | indicator-bluet |

(a)

| PID | PPID | PROCESS NAME | DTB VALUE |
|---|---|---|---|
| 0 | 0 | swapper/0 | ---- |
| 1 | 0 | init | 0x36309880 |
| 2 | 0 | kthreadd | ---- |
| 3 | 2 | ksoftirqd/0 | ---- |
| 4 | 2 | kworker/0:0 | ---- |
| 5 | 2 | kworker/0:0H | ---- |
| 6 | 2 | kworker/u256:0 | ---- |
| 7 | 2 | rcu_sched | ---- |
| 8 | 2 | rcuos/0 | ---- |
| 9 | 2 | rcuos/1 | ---- |
| 10 | 2 | rcuos/2 | ---- |
| 11 | 2 | rcuos/3 | ---- |
| 265 | 2 | migration/0 | ---- |
| 266 | 2 | watchdog/0 | ---- |
| 267 | 2 | watchdog/1 | ---- |
| 268 | 2 | migration/1 | ---- |
| 269 | 2 | ksoftirqd/1 | ---- |
| 270 | 2 | kworker/1:0 | ---- |
| 271 | 2 | kworker/1:0H | ---- |
| 272 | 2 | khelper | ---- |
| 531 | 1 | upstart-udev-br | 0x36308a80 |
| 537 | 1 | systemd-udevd | 0x36308700 |
| 576 | 2 | kpsmoused | ---- |
| 578 | 2 | ttm_swap | ---- |
| 666 | 1 | upstart-scoket | 0x3b396c80 |
| 939 | 1 | systemd-logind | 0x3b395b00 |
| 941 | 1 | bluetoothd | 0x3b390e00 |
| 955 | 2 | krfcommd | ---- |
| 958 | 1 | avahi-daemon | 0x3674aa00 |
| 963 | 958 | avahi-daemon | 0x3630e900 |
| 1132 | 1 | anacron | 0x38a61500 |
| 1147 | 1 | kerneloops | 0x38a62a00 |
| 1149 | 1 | acpid | 0x3b395400 |
| 1150 | 1 | irqbalance | 0x3b391c00 |
| 1156 | 1 | lightdm | 0x3b394d00 |
| 1179 | 1 | cups-browsed | 0x38a63100 |
| 1215 | 1156 | xorg | ---- |
| 1218 | 1 | accounts-dameon | 0x3630e580 |
| 1230 | 1 | cron | 0x3630de80 |
| 1247 | 1 | getty | 0x38a61c00 |
| 1263 | 1 | whoopsie | 0x38a62d80 |
| 1277 | 1156 | lightdm | 0x38a61880 |
| 1285 | 2 | kauditd | ---- |
| 1292 | 1277 | lightdm-greeter | 0x3630aa00 |
| 1300 | 1 | dbus-daemon | 0x38a62300 |

(b)

Figure 3.9: (a) Enumerating the target process details (b) Populating the DTB value

To make the illustration simple, we considered only two CEP buckets in this case study. The first CEP bucket gives the process name if the input is pid. The other one gives the DTB (Directory Table Base) value of the corresponding process. *System.map* file is used in identity paging to convert some of the static addresses. The drawback is, it cannot translate addresses pertaining to all the regions of the memory. In most of the cases, list walking and taking process memory details needs the potential to convert virtual addresses. To accomplish this, we can use Directory Table Base (DTB) and the observations from the victim virtual machine are:

- We identified the process name of the pid=1215 as xorg and the same is shown in Figure 3.9(a).
- To know more about the process xorg, we populated its DTB (Directory Table Base) value and we identified that, it does not have a DTB value (Figure 3.9(b)). The processes which are not having DTB value can be treated as kernel processes.
- In the process of identifying the root cause, the next challenge is to understand as to how the intruder created a fake binary with xorg kernel process ? To answer this, we have taken the memory image of the victim virtual machine through VMI. The acquired image bash history is correlated with the VMI results and we identified that, the intruder aliased the bash kernel process to xorg and created fake binaries.

Finally, we parsed and stored each bucket (introspection file) in MySQL database along with time. The incident handler can submit customized queries to identify the root cause of the incident.
Example Query: Select pid, ppid, pname, DTB_Time, path, trigger_Time by joining the

results from DTB bucket, pname bucket and path bucket where pid=1215. The result of it is shown in Figure 3.10 and the observations are:

- Fake binary is inserted by a process with pid=1215 and pname=xorg
- DTB_time is the time when the corresponding event trace is parsed and stored in the database
- trigger_Time is the time when the defined break point was satisfied
- We identified the root cause of the incident from the above analysis i.e. a kernel process is aliased to create fake binary

This scenario shows the capabilities of the proposed trigger module and CEP engine when they are integrated into the introspection process. The same concept can be applied for any type of incident by considering the positive traces and the negative traces for the root cause identification.

Some more implementation details: We treat each of the populated introspected data as a separate bucket. Accordingly, we stored it as a separate entity in the database by adding timeline i.e. the time of parsing each trace and the time of trigger generation given by the CEP engine. This resulted in improving the semantic interpretation for root cause identification.

| pid | ppid | pname | DTB_time | path | trigger_Time |
|---|---|---|---|---|---|
| 1215 | 1156 | xorg | Mar 17 2015 12:06:59 | /tmp/rm | Mar 17 2015 12:07:16 |

Figure 3.10: Root cause identification for fake binaries

### 3.5.3 Merits of the approach

Our main objective is to increase the semantic interpretation of introspection events in the cloud environment. From our theoretical formulations and the experimental results, our approaches have the following advantages:

- Independent of the type of the introspection model, the trigger module and CEP engine can identify the irrelevant events pertaining to the occurred incident.
- With the aid of CEP engine, we created the flexibility of automatic updates to the rule sets present in the trigger module .
- Analysis time for the incident handler can be reduced as only the suspicious and more relevant events are passed to the correlation engine.
- Devised root cause identification and it drives the incident handler to effectively identify the primary source of failure.

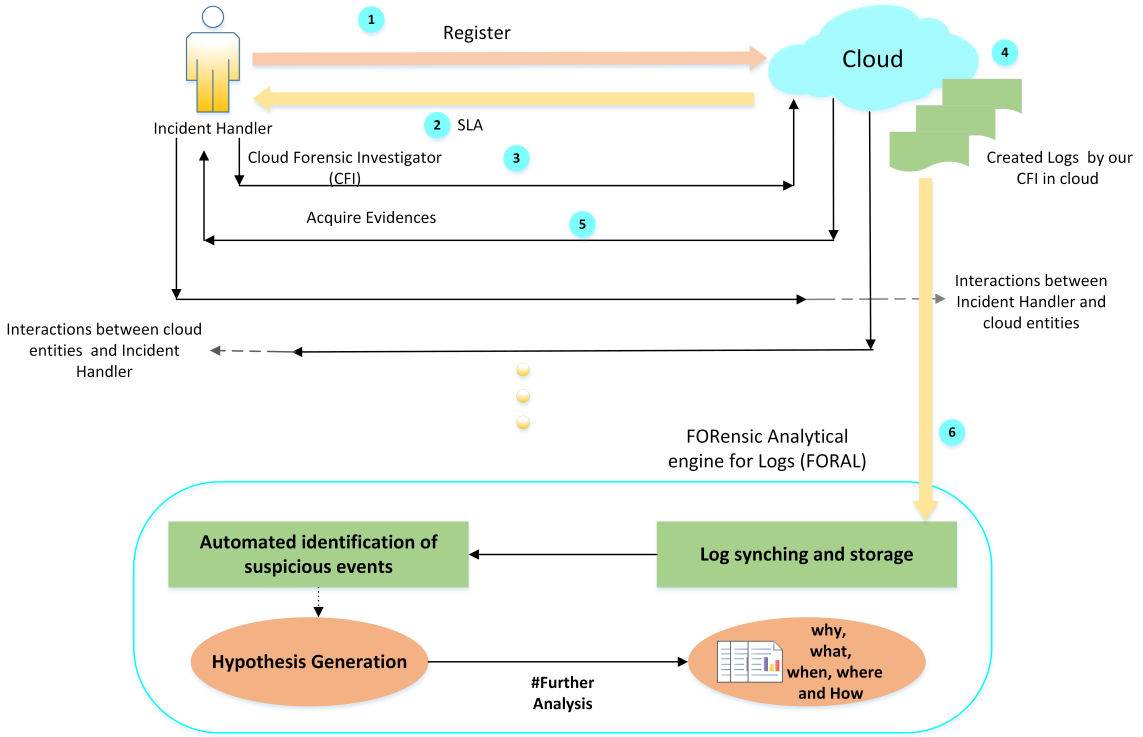



Figure 3.11: Proposed model to improve the transparency between the incident handler and the CSP

The rest of the chapter discusses the part 2 and its contributions.

*Organization of Part 2*: We discuss the proposed model ALTRA in Section 3.6 which can increase the transparency between the incident handler and the CSP. We discuss the two proposed approaches (SeMS and CoPS) as part of our model and these can automatically detect suspicious events performed by the incident handler (Section 3.7). In Section 3.8, we validate both the approaches using a typical investigative scenario.

## 3.6 ALTRA- Proposed Model to Address Lack of Transparency

The Cloud Incident Handler (CIH) should go through the process of registration for each incident which should be standard and legally admissible. The registration will be reviewed by the cloud entities and then accordingly SLAs are prepared. From then, the incident handler can start the process of incident handling using any cloud forensic toolkit. We test the proposed ALTRA model using CFI tool (Cloud Forensic Investigator). CFI is a tool developed by us to perform forensic investigation in IaaS cloud [129]. More details about the CFI are mentioned in Appendix II.

When *CFI or any cloud forensic tool* is used by the Incident Handler then there could be two possibilities: (1) The CIH can be trustworthy and uses the forensic tool (here, CFI) to perform all the healthy activities. (2) The CIH can be untrusted and performs suspicious

activities using the cloud forensic tool (here, CFI). Here, we classify an activity as healthy or suspicious based on the access control policies given to the incident handler. If the incident handler violates those policies then it comes in the category of suspicious else it is treated as a healthy activity. For example, if he/she accessed the data of a tenant for which there is no permission then it falls under the category of suspicious. Currently, no cloud provider is facilitating advanced incident handling services to the end user because of many architectural and legal challenges. *One of the main architectural challenges for the cloud provider is to handle when the CIH is not trusted*.

Since the incident handler is given the access for the cloud infrastructure during the incident handling process, he/she can exploit the opportunity to perform any suspicious activity. The CSP should be aware of the activities being performed by the incident handler when using any CFI. We propose to achieve this by creating a CFI log at the cloud side. This log is basically an application log and contains information like, the time at which the incident handler started the process of acquisition, locations accessed by the incident handler, the objects read along with the corresponding time, the list of artifacts acquired, the time elapsed to acquire each artifact, the IP from which the incident handler accessed the cloud, the objects modified by the incident handler along with its access and modification time, the time at which the incident handler completed, etc.

the process of evidence acquisition.

By analyzing the CFI log in cloud, the CSP can know whether the activities performed by the incident handler are suspicious or not. Manually, it consumes a lot of time to analyze the events in the CFI logs. We reduce this time with our Forensic Analytical engine for Logs (FORAL). It contains two modules as shown in Figure 3.11 and we brief each of them as below:

### 3.6.1 Remote log creation and syncing

The log created by the CFI will be stored in the cloud. In the worst case, the incident handler can even access and modify the events in the log as he/she can access the cloud infrastructure during forensic investigation. Our model handles this using the concept of *remote syslog*. In our context, the cloud itself acts as a rsyslog client (node 1) whereas rsyslog server is the dedicated host assigned for the purpose of storing the CFI logs (node 2). Once both the nodes are configured with rsyslog then all the events recorded in node 1 will be continuously synced and stored in the specified node 2 as well. The CFI log in node 1 is accessible to both the incident handler and the CSP but the CFI log events in node 2 can be accessed only by the CSP. This policy of replicating the log events helps the CSP to always have the valid logs with high availability.

### 3.6.2 Automatic detection of suspicious events from the CFI logs

The common drawback across multiple log analyzer tools like Cloudlytics [130], Google Analytics [131] is, they are mostly used for statistical knowledge extraction and cannot be directly applied to answer forensic questions. Our approach of log analysis is well suited for incident handling.

Most of the relevant work which detect suspicious events are at the level of system logs but not at the level of application logs. For example, in [132], the authors found suspicious events from the system logs using known black list and whitelist. This approach of detecting suspicious activities cannot be applied in our case, as individually, the events in the application may seem healthy but when we interpret them as a sequence, they may be categorized as suspicious. We handle this problem using Causality Models and its details are described in the subsequent subsection.

## 3.7 Approaches for Finding Suspicious Events Performed by the Incident Handler

Various challenges are involved in automatically finding suspicious events from any application log (here, CFI application log). In this context, the following points are worthy of mention.

### 3.7.1 Identified challenges

- There are numerous events in the CFI logs and it is very difficult for the CSP to find the events of interest.
- Analyzing all the events to generate the hypothesis consumes a lot of time and sometimes may even lead to wrong hypothesis generation.

We address the above issues by using the concept of *causality models* [150].

### 3.7.2 Building a causality model from cloud forensic application logs to identify forensically relevant events

Causality models are used to show the cause-effect relationship between the events/processes. We construct this causality using Directed Acyclic Graphs (DAG). The major advantage with DAGs is, it can represent different associations starting from simple to complex ones like confounding, endogenous association, and d-separation [134]. This makes one to use DAG for modeling the relations for any type of application irrespective of its complex logic behind event generation. Our idea is to construct a DAG from the logs of the corresponding application. For this, we apply the proposed approaches- SeMS/CoPS.

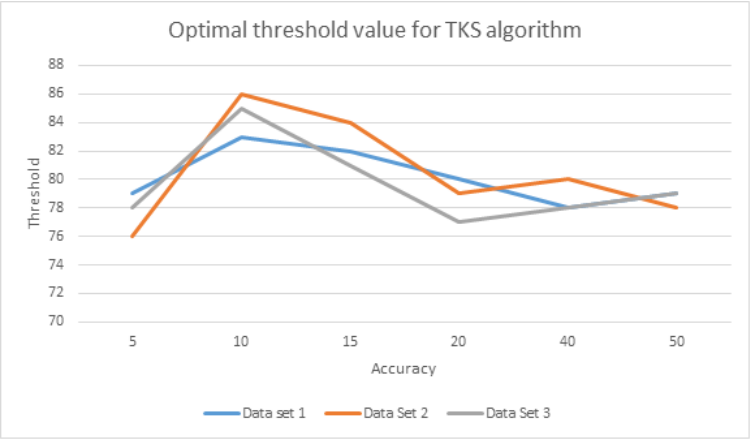



Figure 3.12: Experiments conducted to decide the k value for different data sets

**Applying Sequence Mining to identify and build causal relations between the events (SeMS)**

Applying sequence mining to build causalities will work due to the following reasons:

- In every application, a sequence of events will occur starting from its launch time to termination/closing time. If two events $e_x$ and $e_y$ occurred in a sequence of an application session then we can say that $e_y$ is the outcome of the cause $e_x$.
- The events in the sequence which are not co-occurring frequently can be identified easily through sequence mining. Those sort of events can be treated as outliers and will be forensically relevant.

There are many sequence mining algorithms but we used TKS (Top-k Sequences) as it is more suitable for dense data sets. We used TKS to initially get the top-k frequent item sequences. Here, k is the count of the most relevant sequences out of all the sequences. We arrived at a K value of 10 percentage of the data set after a series of trials and observed accuracy improvement (Figure 3.12). Now, Algorithm 2 is applied to get the suspicious sequences. Say, the CSP is interested to know the suspicious sequences in CFI log, then each new sequence in the log during Time Window T is compared with the frequent item sequences (freq_seq). If a mismatch occurs, the percentage of fraction left ($per\_fractionLeft$) will increase and if it is more than the user threshold ($th_{seq}$) value then it is considered as suspicious sequence.

It is always tricky to decide the exact value of threshold. For the current problem, we identified various threshold value decision parameters like, history about the suspicious sequences generated from the target application, investigating entity experience, and the environment in which the log is stored. The only problem with SeMS is, it cannot quantify the prediction of suspicious sequences. We proposed another approach called CoPS.

**Algorithm 2** Finds suspicious sequences from cloud forensic application logs using SeMS

Input: A set of cloud forensic application sequences during TimeWindow T, $th_{seq}$
Output: Suspicious sequences $S_p, S_q, ..., S_y$ where each sequence contains set of events.
$freq\_seq[\,] = apply\_seqMining()$
**for each** $sequence\ S_i\ in\ TimeWindow\ T$ **do**
  **for each** $sequence\ S_j\ from\ freq\_seq$ **do**
    **for each** $item\ I\ in\ S_j$ **do**
      **if** $S_i\ contains\ I$ **then**
        remove $I from\ S_i$
      **end if**
    **end for**
  **end for**
  $residue = original\_length(S_i) - new\_length(S_i)$
  $per\_fractionLeft = (residue/original\_len) * 100$
  **if** $per\_fractionLeft > th_{seq}$ **then**
    consider $S_i$ as suspicious
  **end if**
**end for**

**Building the causalities between the application events using conditional probability (CoPS)**

We also identify the suspicious sequences of a cloud forensic application using conditional probability. A conditional probability can be simply defined as:

"*The extent of belief of the occurrence of an event $E_x$ when $E_y$ was given* [135]"

This will work because there is a cause and effect association between the log events of any application (here CFI). The extent of a cause leading to effect can be decided with the associated probabilities. Once the probabilities are calculated then deciding the suspicious sequence is a trivial job.

Here also, we take all the new sequences in the time window T and initially construct a trie tree. The reasons for using trie tree are: (a) It is an ordered tree where keys are generally strings. (b) Descendants of any node will have some common prefix which can be useful in sequence matching.

For each node in trie tree, we calculate the conditional probability (P) using equation 3.7.1.

$$P = (C(n_i)/C(n_j)) \tag{3.7.1}$$

where $n_i$ is the current node, $n_j$ is the parent node of $n_i$ and C($n_i$), C($n_j$) indicate the respective node count. If the probability of an item in the sequence is less than the predefined threshold ($th_{seq}$) then the sequence is treated as suspicious (Algorithm 3).

**Algorithm 3** Finds suspicious sequences from cloud forensic application logs using CoPS

Input: A set of cloud forensic application sequences during TimeWindow T, Threshold $th_{seq}$,
Output: Suspicious sequences $S_p, S_q, ..., S_y$ where each sequence contains set of events.
**for each** $sequence\ S_i\ in\ TimeWindow\ T$ **do**
  **for each** $item\ I_a\ in\ sequence\ S_i$ **do**
    calculate the probability P of $I_a$ node considering the occurrence of previous item $I_{a-1}$ node for all $S_j$
  **end for**
  **if** $P(I_a) < th_{seq}$ **then**
    consider $S_i$ as suspicious
  **end if**
**end for**

### 3.7.3 Comparison of SeMS annd CoPS

Both the approaches have been applied to find the suspicious events in CFI logs. Based on the CSP's requirement, an appropriate method can be chosen. To take up this decision, a comparison between the two approaches is made and the same is briefed in Table 3.3.

## 3.8 Automatic Identification of Suspicious Events: A Typical Scenario

To validate the proposed approaches (SeMS and CoPS), we have taken our own application-Cloud Forensic Investigator (CFI) which is developed for performing incident handling in the IaaS cloud environment [129]. The detailed scenario is given below.

### 3.8.1 Scenario description

A cloud user virtual machine $VM_x$ is compromised by the intruder's virtual machine $VM_i$. Then $VM_x$'s user raises a complaint to the CSP. The CSP employs an internal incident handler to start examining the incident. In this case, the incident handler can use our proposed model which involves major activity of running the cloud forensic application. The capabilities of our CFI tool are briefed in Appendix II. More fine grained sequence of steps are briefed below:

- New Case (Step 1): The incident handler creates a new case based on the registered complaint. In this stage, the CIH enters various details about the case.
- Configuration Settings (Step 2): Here, the incident handler can enter credentials for various cloud nodes. For example, when the cloud forensic application runs in the openstack cloud, then the incident handler should enter the cloud compute node credentials. The compute node acts like a hypervisor in the openstack and this node

Table 3.3: Comparison between SeMS and CoPS for the application logs

| S. No | SeMS (using Sequence Mining for finding Suspicious sequences) | CoPS (using Conditional Probability for finding Suspicious sequences) |
|---|---|---|
| 1 | Finds the top-k frequent item sequences and calculates the residue by comparing with the sequences in Time window T | Finds out the conditional probabilities of all events in the input sequences of time window T |
| 2 | Input sequences above the threshold are suspicious | Input sequences of conditional probability below the threshold are suspicious |
| 3 | Memory efficient when the data set is small | Memory efficient when the data set is large |
| 4 | Time complexity of finding top-k frequent patterns: O(M*N) where N is user sequences and M is input sequences | Time complexity of building trie : O(N*L), where L is length of largest sequence and N is number of user sequences |
| 5 | Time complexity of finding outlier: O(N*k*L) where N is number of user sequences, L is the maximum length of frequent sequence | Time complexity of finding outliers: O(N) where N is number of user sequences |
| 6 | Accuracy is governed by threshold and value of k. If value of k is not set correctly, then certain suspicious sequences may be reported as healthy | Accuracy is governed by threshold alone. |

contains the vDisk and service logs of the target and non-target instances. The configuration should also accept the Openstack cloud user's dashboard credentials by which the target virtual machine's vRAM can be acquired. There is a chance of trust violation in this step which we describe in the next section.

- Selective acquisition (Step 3): The incident handler can select the required evidences and transfer them to an isolated environment. To prove to the court of law regarding the admissibility of the acquired evidences, the checksum is calculated at the cloud side and recalculated at the incident handlers node. If both the checksums are equal then it is legally accepted otherwise not.

- Analysis (Step 4): All the acquired evidences can be examined using the CFI analysis capabilities and then the results are exported for legal proceedings.

### 3.8.2 Challenges to handle in the above scenario

There are certain issues that can be raised during Step 2 and Step 3. For example in step 2, to acquire the introspection based vRAM events of the $VM_x$ user, the incident handler needs to enter the compute node (hypervisor) details. Since the cloud is a multi-tenant environment, the incident handler can introspect the vRAM events of non-target users and this leads to the privacy violation of those users. We developed CFI with the basic assumption that the incident handler is trustworthy. In reality, always this assumption may not hold good i.e. if the incident handler is not a trusted entity then he/she can perform any suspicious activity like acquiring other users' data.

### 3.8.3 Applying the proposed approaches to find suspicious events in CFI logs

We setup Openstack cloud with high end configuration following its legacy networking architecture. We emulated the role of a bad incident handler and then performed suspicious activities using CFI tool in the Openstack cloud. The activities performed using CFI tool are logged at the cloud side. SeMS and CoPS are then used to detect the suspicious activities in CFI log.

#### SeMS

We pre-processed the log by assigning a unique number for each event and the same is shown in Figure 3.13. For example, presence of event "1" in the pre-proceesed log indicates that new case object has been invoked, number "2" indicates that the details of the case are entered by the incident handler and like these, each number in Figure 3.13 represents an event generated by the incident handler (-1 is used as separator between every two events and We have taken a minimum number range of 10 and maximum of 2 million). After giving the pre-processed CFI log as input to the sequence mining algorithm, we got top-k frequent sequences at one instance of time (Figure 3.14). The DAG is constructed by running the TKS for multiple instances (Figure 3.15). Each user sequence in Time T is given as input to Algorithm 2 and then it checks whether the input sequence is suspicious or not. The same is shown in Figure 3.16 (here, user threshold value is taken as 25 %). We decide this as threshold value based on the number of event types in the CFI application log.

#### CoPS

We represented the sequences in a given Time window T using *trie* data structure (ordered tree with a common prefix to the descendants of each node). We iterated through *trie* for each non-T sequence of the application and updated the count of each node for the matching prefix. Finally, the probability of event occurrence P in the sequence is calculated. If the

```
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 7 -1 4 -1 -2
12 -1 11 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 9 -1 4 -1 -2
12 -1 11 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 4 -1 -2
12 -1 11 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 9 -1 10 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 9 -1 10 -1 11 -1 4 -1 -2
12 -1 11 -1 10 -1 4 -1 -2
12 -1 11 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 9 -1 10 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 4 -1 -2
12 -1 11 -1 10 -1 4 -1 -2
12 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 4 -1 -2
12 -1 11 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 4 -1 -2
12 -1 11 -1 -2
12 -1 11 -1 10 -1 4 -1 -2
1 -1 2 -1 3 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 7 -1 4 -1 -2
1 -1 2 -1 3 -1 5 -1 6 -1 7 -1 8 -1 4 -1 -2
```

Figure 3.13: Events in the CFI log after pre-processing

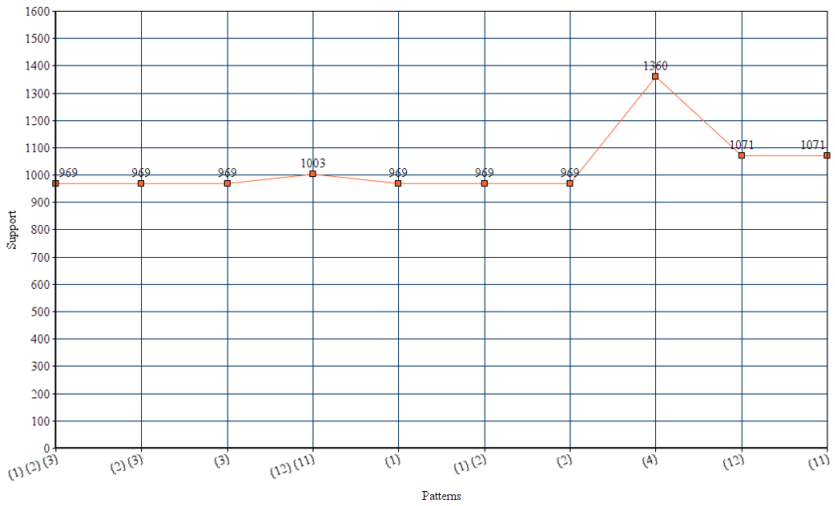



Figure 3.14: Frequent top-k sequences identified using SeMS

probability of any event is less than the given threshold then we considered that as suspicious. For the same input file in Figure 3.13, CoPS identified the suspicious events and the same is shown in Figure 3.17.

The comparison between SeMS and CoPS is shown in Table 3.4. Summarizing it, execution time for SeMS is high than CoPS and the memory consumption for CoPS is high when compared with SeMS.

Since the same CFI log events are given as input to both the SeMS and CoPS, the same sequence is detected as suspicious by both the approaches (Figure 3.16 and Figure 3.17). Describing the suspicious sequence found i.e. (Seq: {1} -1 {2} -1 {3} -1 {5} -1 {6} -1 {7} -1 {8} -1 {9} -1 {13} -1 {4}), the incident handler created a new case object (item 1), then details of the case were entered (item 2), did basic enumeration about the target VM (item 3), acquired the vdisk of the target VM (item 5), modified the nova.api log at /var/log/ of compute node (item 6), modified the keystone-all log at /var/log/ of compute node (item 7), acquired non-target VM vdisk (item 8), did advanced enumeration about the

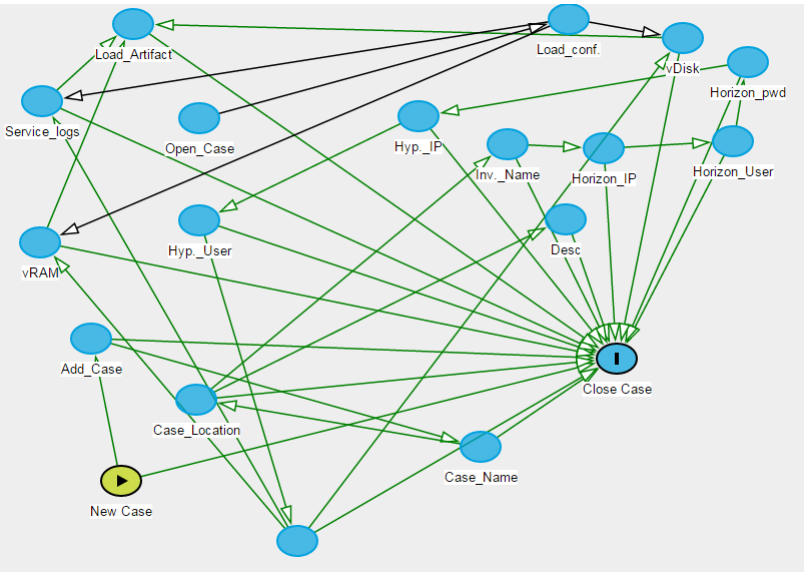



Figure 3.15: DAG constructed from the healthy causalities of CFI log

Table 3.4: Comparison between SeMS and CoPS

| No. of sequences | Total Time (ms) for SeMS | Peak Memory (mb) for SeMS | Total Time(ms) for CoPS | Peak Memory (mb) for CoPS) |
|---|---|---|---|---|
| 10 | 22 | 1.81805603 | 11 | 1.890281577 |
| 1450 | 130 | 6.30869293 | 91 | 7.572235107 |
| 4350 | 184 | 15.12903595 | 162 | 19.5663223 |
| 142100 | 709 | 79.12886047 | 519 | 81.42918396 |
| 2121600 | 7356 | 709.9880905 | 3797 | 711.2549823 |

non target VM (item 9), modified the host OS logs (item 13), terminated the application without closing the case (item 4). It is important to note that, each item details mentioned above are the message descriptions about the event in CFI log and can vary based on the application. When the suspicious sequences are identified by SeMS/CoPS, then it indicates that the incident handler who had accessed the cloud for evidence acquisition is not trusted and CSP can further proceed with legal actions.

*Correctness of SeMS and CoPS*: We gave all the events generated from the suspicious activities to both SeMS and CoPS. We observed that, both the approaches achieved an average accuracy of 82.3 %.

So SeMS and CoPS alert the CSP whenever the incident handler performs any suspicious activity in cloud. Analyzing the detected suspicious activities can make the cloud provider to generate the hypothesis quickly by which he/she can get the answers for several questions like, *when*, *what* and *where* the suspicious activities are performed by the incident handler.

```
<terminated> LogMiner2 [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (30-May-2016, 12:55:26 pm)
Enter input file path:
/home/batman/tksin2.txt
Enter output file path:
/home/batman/tksout2.txt
Enter k:
10
=============  Algorithm TKS v0.97 - STATISTICS =============
Minsup after preprocessing : 374
Max candidates: 9 Candidates explored  : 19
Pattern found count : 10
Time preprocessing: 57 ms
Total time: 73 ms
Max memory (mb) : 6.308692932128906
Final minsup value: 692
Intersection count 0
=================================================

Number of patterns: 10
Enter file containing sequences from date range:
/home/batman/tksseq.txt
Mismatches:

============================1============================
User seq: 1 2 3 5 6 7 8 9 13 4
Fraction Left: 60.000004%
Sequence left: 5 6 7 8 9 13
Verdict: Suspicious

============================2============================
User seq: 12 11 10 4
Fraction Left: 20.0%
Sequence left: 10
Verdict: Healthy
```

Figure 3.16: Suspicious sequences identified by SeMS

```
<terminated> cfiCondi [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (31-Mar-2016, 10:45:30 am)
Enter user sequence file:
/home/batman/tksseq.txt
Enter input file:
/home/batman/tksin2.txt
===============1===============
User seq: 1     2       3       5       6       7       8       9       13      4
Cond prb: 47    100     100     92      77      60      100     68      5       100
Verdict: Suspicious
===============2===============
User seq: 12    11      10      4
Cond prb: 52    93      32      94
Verdict: Healthy
```

Figure 3.17: Suspicious sequences identified by CoPS

### 3.8.4 Advantages of the proposed model

There are several aspects considered by the proposed model. Some of them are briefed below:

- For any user: Even though, the approaches are validated to identify untrusted incident handler, they can be used to identify any user with bad intentions. In that case, SeMS or CoPS takes the events from the cloud system logs/VM logs.
- Completeness: Our model gives the complete process involved in cloud incident handling starting from the incident handler registration to report generation.
- High availability and reliability: Our CFI application uses the concept of remote logging at multiple nodes. This ensures that CSP can rely on the log entries in the node which is not accessible to the incident handler. Due to the redundant sync logging feature, the log events are always available.
- Accurate hypothesis generation: Without applying SeMS or CoPS, the number of events on the event reconstruction timeline will be very high and this will create difficulties in generating hypothesis. After applying our approaches, the number of events

on the timeline reduced drastically and they are forensically relevant. This finally would lead to arrive at accurate hypothesis about the occurred incident.

- Automatic identification of suspicious events: The suspicious events from the CFI log are identified automatically without much human effort.

## 3.9 Summary

The contributions of this chapter have been organized into two parts. The first part deals with increasing the availability of the volatile evidence vRAM and then analyzing it to have accurate logical findings. The second part addresses the architectural issue pertaining to the first part which can make the CSP to asses the trust aspect of the incident handler activities.

The volatile evidences play a key role during the forensic based incident handling. Its availability cannot be always ensured. In this chapter, we focused on capturing the memory events from VMM level through Virtual Memory Introspection (VMI). The approach of introspection reduces the changes in the target instance virtual memory and this may increase the legal admissibility of the acquired memory events. We observed that the number of introspected events of a VM is usually high and this introduces new difficulties for the incident handler in finding events of forensic relevance. We reduced the challenges in this issue by designing a *trigger based introspection* model. This can detect the known incidents and variants of known incidents. It is important to note that, detection alone is not sufficient and requires the incident handler to find the root cause of the incident. We achieved this using Complex Event Processing (CEP). During the process of handling cloud incidents, incident handler can perform suspicious activities and once it happens, FORAL engine of our model detects and alerts the CSP using the proposed approaches- SeMS or CoPS. Thus, the model improves the transparency between the incident handler and the CSP.

The work presented in this chapter is accepted and published work in [PUB4], [PUB6], [PUB9, [PUB10] and [PUB11] (refer page no. 136-137). In the next chapter, we discuss the proposed approaches for analyzing cloud service logs for incident handling.

# Chapter 4

# A Model for Effective Event Reconstruction using Cloud Service Logs

*"Cloud services help organizations move faster, lower IT costs, and scale"*

- Amazon

In Chapter 3, we discussed the devised solutions for handling volatile evidences. In the current chapter, we discuss the proposed approaches for analyzing the cloud specific artifact named *service logs* which can serve as evidences for incident handling. Firstly, we discuss their role during forensic based cloud incident handling. We then describe the proposed segregation approaches and finally apply the proposed *aggregation* approaches on the events of service logs which can lead to effective incident handling.

## 4.1 Background and Motivation

In this chapter, we consider cloud service logs as evidence for incident handling as we identified many events which are relevant to the occurred incident at VM level.

### 4.1.1 What are service logs?

The IaaS cloud solutions are provided by many inter-operable services. For each service, there is an associated log created at the cloud node(s). Each log contains events of multiple tenants, multiple users and multiple virtual machines. An example service log event reflecting the same from Openstack cloud is shown in Figure 4.1.

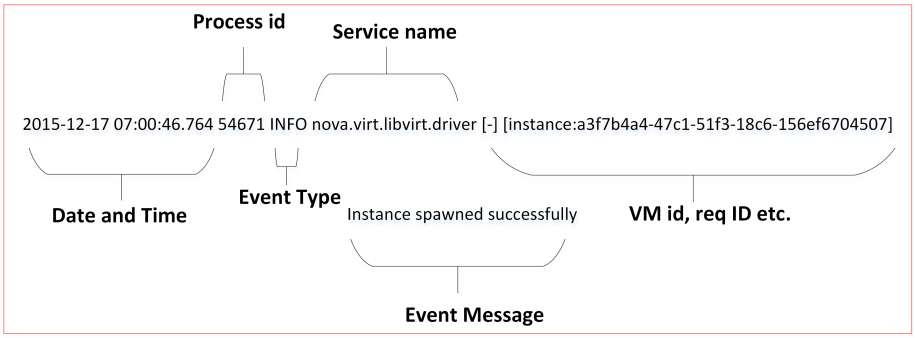



Figure 4.1: A sample service log event in the Openstack cloud

**Identified role of Service Logs for incident handling using forensic practices**

Service logs are generally used for statistical and debugging purposes [136]. After going through the extensive set of events in the service logs, we found that events in these logs can also be used for cloud incident handling (Figure 4.2). Also, cloud service logs can answer many forensically relevant questions during incident handling i.e.

- When a virtual machine has been terminated ?
- What are the critical events generated by an instance in the specified time period T ?
- When the event burst had happened and what was the service responsible for it ?
- Is there any anomaly usage at virtual machine level in the specified time range ?
- What are the high level VM activities performed by the target user ?
- Which user has more failed login attempts ? etc.,

2015-07-25 11:09:34.126 AUDIT nova.compute.manager [req-723411f3-4127-3516-6f4f-04d3417fd21eFixedIPsTestXml-754566134FixedIPsTestXml-561489020] [instance:43f3f7c3-34c1-452-36f7-174fa4895781] **Terminated instance abruptly**

2015-07-25 11:37:12.151 INFO nova.virt.libvirt.driver [req-f14571818-3d41-5d24-d464f563af5612 FixedIPsNegativeTestXml-6736541941-354326374FixedIPsTestXml-561489020][instance: fd027464-6e15-4f5d-8b1f-c389bdb8772a] **Creating image with negative IP pool**

2014-01-26 15:36:11.525 INFO nova.virt.libvirt.driver [req-ded47509-1e5d-4fb2-a0a92932eba9271 FixedIPsNegativeTestXml-1126589427 FixedIPsNegativeTestXml-38506689] [instance: fd0274d4-6e15-4fed-8a1f-c389bdb87a27] **Unauthorized access to config drive**

Figure 4.2: Identified service log events with forensic relevance to incident handling

Once the target evidence is acquired (here, cloud service logs), the incident handler will apply various analysis techniques [137]. One of the analysis approaches is Event Reconstruction and it helps the incident handler in many ways like, event sequencing and hypothesis generation/testing [138]. We observed that cloud service logs contain huge number of

events and it would be highly difficult to interpret/generate hypothesis from all of them. We propose that the incident handler can emphasize on segregation and aggregation based event reconstruction of the cloud service logs using which the hypothesis about the occurred incident can be generated quickly (our scope).

### 4.1.2 Effective event reconstruction of cloud service logs

Event Reconstruction (ER) is the process of finding why an evidence possess certain characteristics [138]. Event Reconstruction in cloud is relatively new and it poses additional challenges when a unique evidence like Service Logs is considered. We have done extensive literature survey to identify various aspects that the incident handler should consider while performing event reconstruction and the same are discussed below.

The work on event reconstruction is classified in to three categories: (i) Resource based ER (ii) Role based ER and (iii) Timeline based ER.

1. Resource based event reconstruction: Considers various system resources for evidence collection and then passes them as input for event reconstruction phase. We discuss the relevant research contributions below:

    - In [139], the authors have suggested that evidence availability plays a key role in event reconstruction and the availability of evidences can be improved with the forensic readiness models. They considered system call traces as events and collected all of them from various system resources on a continuous basis. *Advantage*: The events collected are comprehensive and reliable. *Disadvantage*: Storing and analyzing all those events is time consuming and involves a lot of human effort.
    - In [140], the authors captured process and file management based events and then performed event reconstruction. *Advantage*: For quick analysis, the number of events is reduced by considering offset intervals. *Disadvantage*: Since the number of resources from which events are collected is very less, the analysis of the events would be incomplete and hence legally inadmissible.
    - In [141], flight data recorder captures the system calls, converts them to events and finally stores in the logs. *Advantage*: Reduces the number of events to be analyzed by removing the duplicate entries. *Disadvantage*: It is advisable to identify the activity bursts in the logs so that time spent on the analysis can be reduced. It is difficult to find the activity bursts at regular intervals of time in the cloud environment due to its multi-tenant nature.

2. Role based event reconstruction [141]: In this, thousands of objects in the target evidence are classified based on their characteristics. Based on the incident features, relevant objects are identified and allowed to perform event reconstruction on each.

The sequence of phases followed to perform role based event reconstruction is: Evidence examination → Role classification → Event construction and testing → Event sequencing → Hypothesis testing.

- Advantages: (1) Initial impressions about the incident can be known (2) Considers the causal associations between the objects for hypothesis generation and testing.
- Disadvantages: (1) Most of the phases are not automated and require more human involvement (2) To arrive at correct hypothesis, the incident handler has to spend a lot of time.

3. Timeline and miscellaneous based event reconstruction approaches: In this, the hypothesis is generated based on time based event sequencing. This is a simple and generic approach that can be extended to other environments. Some of the works in timeline and miscellaneous based event reconstruction are briefed below:

   - There are certain tools for performing time based event reconstruction like Cyber Forensic Time Lab [143], Log2timeline [142], and Zeitline [144]. We identified that none of these address the challenge of handling the evidence when it contains more number of events as analyzing all of them to perform event reconstruction would be highly time consuming.
   - In [145], the authors talk about reducing the number of events by converting all the low level events to high level events. *Advantage*: The number of events on the timeline is reduced to aid in effective event reconstruction. *Disadvantages*: The current rules mentioned cannot be applied to any other scenario. Moreover, generating rules for each scenario is not an appreciable approach. The rules framed by the authors are specific to the file system and cannot be applied to detect the same scenario in another file system.
   - In [146], the authors tested the generated hypothesis using Finite State Machines. *Advantage*: Few widely accepted forensic theories are used. *Disadvantages*: Cannot represent complex scenarios as it may lead to combinatorial explosions.
   - In [147], the data extracted from the logs is given as input to neural networks for performing logical reasoning behind the current state of an object. *Advantage*: It can detect the activities of the application, based on the traces left by the user. *Disadvantage*: Performance overhead is very high.
   - In [148], the authors correlate various objects in an artifact to perform event reconstruction. *Advantage*: Multiple artifacts of the system are considered for analysis. *Disadvantage*: The number of events to analyze has to be reduced for effective reconstruction which is not discussed.

**Deductions:** From the above literature, we identified the aspects to be considered for effective event reconstruction of cloud service logs. They are:

- **Evidence Acquisition**: Evidences in cloud like service logs cannot be physically seized and acquired. If acquired, transferring them securely to the incident handler's machine should be considered.
- **Multi-tenancy and privacy violation**: Cloud service logs contain events pertaining to multiple users and they cannot be directly given to the incident handler who is generally interested in single user events. To preserve the privacy of other users, proper segregation approaches/policies should be devised; otherwise it leads to privacy violation.
- **Large number of events in the evidence**: Due to the high rapid elasticity property of cloud, the rate of event logging will generally be high. It would then be difficult for the incident handler to perform event reconstruction by considering all of those events. Events in the cloud service logs should be reduced without compromising on the data loss.
- **Data heterogeneity**: The data collected from a crime scene (here, service logs) can be in different formats. The forensic incident handler should then handle the format (the events in the same artifact will be logged in different formats), the temporal differences (the time related information of the same artifact may differ in cloud as it can provide the services from multiple time zones) and the semantic differences (an event can be represented in different ways) that can exist depending on the cloud service models and deployment models.

Event Reconstruction (ER) is a complex task in the entire forensic investigation process. It is important to note that, ER helps the incident handler in many ways like, event sequencing, and hypothesis generation/testing. The scope of our work for event reconstruction limits to effective hypothesis generation for incident handling.

## 4.2 Hypothesis Generation from Cloud Service Logs

Our objective is to generate quick interpretation from the target user/VM events in the considered cloud evidence (here, service logs) such that it should lead to effective hypothesis generation. We identified three challenges in addressing the above problem. They are:

- Service Logs Challenge 1- Event Segregation: Since the cloud is a multi-tenant environment, it contains evidences of multiple users/virtual machines. Acquiring the target user events without violating the privacy of other users is challenging.

Table 4.1: Statistics on Service log events of Openstack cloud

| Number of events in each service | | | |
|---|---|---|---|
| S.no. | Service | Description | No. of events |
| 1 | Nova | Manages VM life cycle | 546247 |
| 2 | Keystone | Authentication service | 65461 |
| 3 | Glance | Stores, retrieves disk images | 986545 |
| 4 | Ceilometer | Metering Service | 6531108 |
| 5 | Cinder | Facilitates volumes | 1751854 |
| 6 | Swift | Stores, retrieves unstructured objects | 212495 |

- Service Logs Challenge 2- Events of Forensic Relevance: All events in the target evidence (here, cloud service logs) cannot be used during incident handling. Automatically identifying whether an event is forensically relevant or not is challenging.
- Service Logs Challenge 3- Quick interpretation: The number of events in the cloud evidence (here, service logs) is very high and it would be highly time consuming for the incident handler to interpret all those events (Table 4.1). Devising new approaches is required so that the time spent by the incident handler for event interpretation should be less.

We address the above challenges by proposing a model named SEASER and the same is discussed in Section 4.3. The Section 4.4 discusses the proposed approaches for event segregation in service logs. We discuss the proposed aggregation algorithms for service logs in Section 4.5. Finally, we conclude the chapter in Section 4.6.

## 4.3 SEASER: Proposed Model for Effective Event Reconstruction of Cloud Service Logs

In this section, we discuss the proposed comprehensive model-SEASER (Applying Segregation, Aggregation on Cloud Service logs) which can address the three challenges identified for effective Event Reconstruction (Section 4.2). It contains the following phases:

1. Identification_L1: In this phase, the incident handler initially perform analysis on the cloud environment to know whether the incident had actually occurred or not. If so, its details are extracted at higher level for examination. For example, the tenant (victim/intruder) details, the zone(s) from which a tenant is receiving the cloud services, number of virtual machines used by the tenant and their configurations etc., can be known.

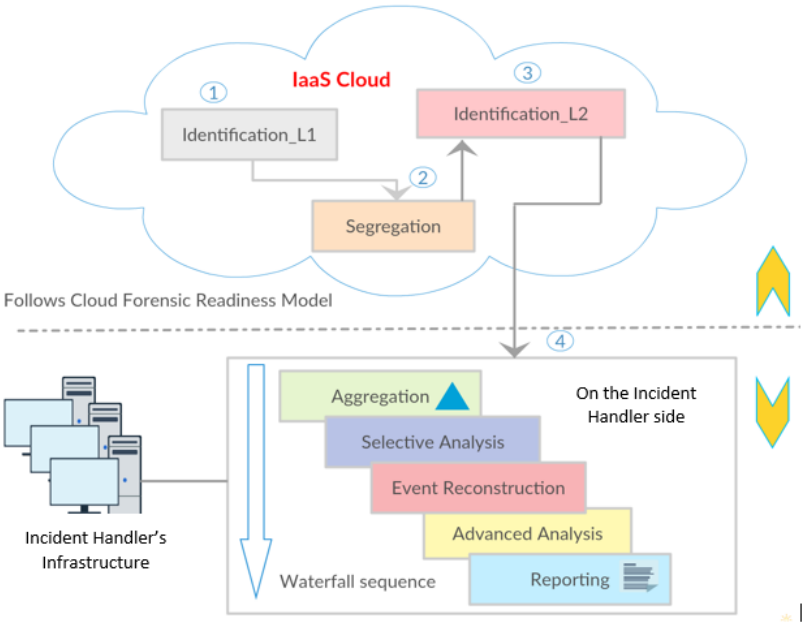



Figure 4.3: Proposed model for cloud event reconstruction

2. Segregation: Once the target user is identified then his/her evidences can be acquired. As cloud is a multi-tenant environment, we must ensure that the privacy of other tenants is not violated during acquisition.

3. Identification_L2: All the identified target user/VM events will not be useful for the incident handler. Automatically identifying forensically relevant events would help the incident handler generate accurate logical findings about the occurred incident. The objective of this phase is to divide the events in the service logs into two classes i.e. *incident handler* relevant events and useful events for *CSP*. It is important to note that, using this phase we address the mentioned service logs challenge 2-Events of Forensic Relevance (Section 4.2).

4. Aggregation: We observed that the number of events in the service logs is more. This phase reduces the total set of events without much data loss using which the incident handler can generate the hypothesis quickly.

5. Selective Analysis: Analysis on the aggregated events will make the incident handler to identify objects of forensic interest pertaining to the occurred incident. Each of those objects' state is recorded and then passed to the event reconstruction phase to know how an object reached its current state.

6. Event Reconstruction (ER): The set of filtered events from the above phases are given as input to any event reconstruction timeline tool. During ER phase, the extent of human effort involved is generally very high and cannot be completely automated [149]. As the process is manual, the strategy to perform ER differs from one incident handler to the other. Discussing all those possibilities is out of the scope of our work.

We discuss the proposed segregation and aggregation approaches. The advantage with our approaches is that they can be completely automated.

7. Advanced analysis: This is generally used for knowing more details about the occurred incident using indexing, carving, hash filtering, file system analysis, low-level file analysis etc.

8. Reporting: Using standard forensic templates, the results from the above phases are structured such that it is a legally admissible report with proper reasoning.

Among all the above phases, we focus on *Segregation*, *Identification_L2* (Section 4.4) and *Aggregation* phases (Section 4.5).

## 4.4 Proposed Evidence/Event Segregation Approaches for Cloud Service Logs

In this section, we address the following challenges when service logs are considered for cloud incident handling.

- Identifying events of the target instance/tenant (*Segregation* phase): For identifying the target user/VM events, we propose parameter-based and session-based approaches.

- Finding events of forensic relevance (*Identification_L2* phase): We address this challenge using machine learning algorithms from which the events in the service logs are segregated as forensically relevant and irrelevant classes.

**(A) Proposed parameter and session based segregation approaches**

Logs, especially the service logs play a crucial role in the domain of forensic based cloud incident handling [78]. If an incident occurs at the tenant virtual machine level then analyzing cloud service logs would take the investigation forward and thereby helping in the incident handling process. As said, the major problem over here is, service logs contain the events of all tenants and the incident handler should be allowed to access only the target tenant events otherwise it would violate the PRIVACY of other users.

**Parameter-based Segregation**: We devised a solution that segregates the events in service logs pertaining to each virtual machine by using a parameter like *instance-id* (Figure 4.4). This parsing logic takes the target instance details and service log as inputs from which it outputs the segregated events pertaining to a target user VM. We also used other parameters like uid and tenant-id to increase the accuracy in the segregation process. There are some cloud service logs without these segregation parameters. The above approach cannot be applied for those logs.

**Session-based Segregation**: To overcome this, we segregate the events based on the user sessions. The session of the target user is identified using the tenant name by which the list of VMs running can be known. For example, the Openstack instances running in the target tenant can be known using OS4j API [151]. Finally, by taking all those VMs as base, the events pertaining to the target tenant are identified (Figure 4.5).

```
2015-03-11 00:06:13.731 51530 INFO
nova.scheduler.filter_scheduler [req-789dade4-73c2-43bf-
b652-967782e300b3 5965ca04ed4742fb933449d5419542c6
1fd0ad38cf464ddf939accf861db21e1] Attempting to build 1
instance(s) uuids: [u'22c32366-05e4-4c66-b8f9-b0e355eb5640']
2015-03-11 00:06:13.734 51530 INFO
nova.scheduler.filter_scheduler [req-789dade4-73c2-43bf-
b652-967782e300b3 5965ca04ed4742fb933449d5419542c6
1fd0ad38cf464ddf939accf861db21e1] Choosing host WeighedHost
[host: compute, weight: 1.0] for instance 22c32366-05e4-4c66-
b8f9-b0e355eb5640
2015-03-11 00:06:19.946 51530 INFO
nova.scheduler.filter_scheduler [req-789dade4-73c2-43bf-
b652-967782e300b3 5965ca04ed4742fb933449d5419542c6
1fd0ad38cf464ddf939accf861db21e1] Attempting to build 1
instance(s) uuids: [u'22c32366-05e4-4c66-b8f9-b0e355eb5640']
```

Figure 4.4: Segregating the service log events for each instance

```
[Tue Oct 13 08:39:46.868885 2015] [:error] [pid 39619:tid 139734801639168]
Login successful for user "demo"
[Tue Oct 13 08:40:29.396521 2015] [:error] [pid 39617:tid 139734868780800]
Suspended Instance: "MyUbuntuVM"
[Tue Oct 13 09:12:27.015016 2015] [:error] [pid 39619:tid 139734877173504]
Resumed Instance: "MyUbuntuVM"
[Tue Oct 13 09:13:02.293125 2015] [:error] [pid 39618:tid 139734877173504]
Could not delete token 573851e8b35224da5f9e8066c16b1ab0
[Tue Oct 13 09:13:20.029372 2015] [:error] [pid 39617:tid 139734801639168]
Forbidden (CSRF token missing or incorrect.): /horizon/auth/login/
[Tue Oct 13 09:16:40.925280 2015] [:error] [pid 39617:tid 139734868780800]
Deleted token 573851e8b35224da5f9e8066c16b1ab0
[Tue Oct 13 09:17:33.912900 2015] [:error] [pid 39617:tid 139734810031872]
Recoverable error: Quota exceeded for cores: Requested 7, but already used
15 of 20 cores (HTTP 413) (Request-ID: req-f5438ce3-d92b-4829-83cc-
9d50301aae0b)
[Tue Oct 13 09:21:17.640800 2015] [:error] [pid 39617:tid 139734868780800]
Scheduled termination of Instance: "MyUbuntuVM"
[Tue Oct 13 10:08:42.109793 2015] [:error] [pid 39618:tid 139734810031872]
Logging out user "demo"
```

Figure 4.5: Identified **demo** user events using session based segregation

The above approaches would not help the incident handler in identifying the forensically relevant events. We propose a new approach called as, Class-based Segregation and the same is discussed below.

### (B) Applying supervised machine learning algorithms for event segregation on the cloud service logs

*Basic Idea*: Upon manual inspection, we identified that all the events in service logs are not useful for incident handling. We want to divide the events in the service logs into two classes i.e. **Incident Handler (E)** class and **CSP (A)** class. The events in the class *A* should be useful in addressing statistical and maintenance issues of the CSP. The events in the class *E* should be useful for the incident handler i.e. the events in the E class should contain

forensically relevant events. To achieve this segregation (A or E class), we applied various supervised machine learning algorithms.

These algorithms are generally used to infer the supervisory signal for the new instances based on the pre-labeled training data. We applied four major and popular supervised algorithms (**Naive Bayes, Decision trees, Support vector machines and Random forests**) on the cloud service logs. Accordingly, the proposed *class-based segregation* follows the below steps:

1. Service logs are initially preprocessed to suit the requirements of the supervised machine learning algorithms

2. The selected classification algorithm is applied on each service log of the target cloud environment and then corresponding accuracy is measured.

**Results and Discussion:** If the classification accuracy of an algorithm is x % then in our case, it indicates that x % of that new log events were correctly classified as class E. Our objective is to correctly classify and label each new event in the service log as either E or A class.

The Openstack dataset that we considered for applying class based segregation is shown in Table 4.2. We initially applied Naive Bayes on all the major service logs of Openstack. For example, Naive Bayes classified Ceilometer log events into A and E classes and a sample output file for the same is shown in the Figure 4.6. We got the Naive Bayes classification accuracy of 84.90 % for the nova service. It is important to note that, the first column in the Figure 4.6 is the predicted class label (A or E) and the second column is the actual class label (A or E).

Similar process is followed to apply other supervised algorithms for every service log. The end results interms of classification accuracy is shown in Figure 4.7(a), 4.7(b), 4.8(a) and 4.8(b).

Table 4.2: Number of events in the training and testing data

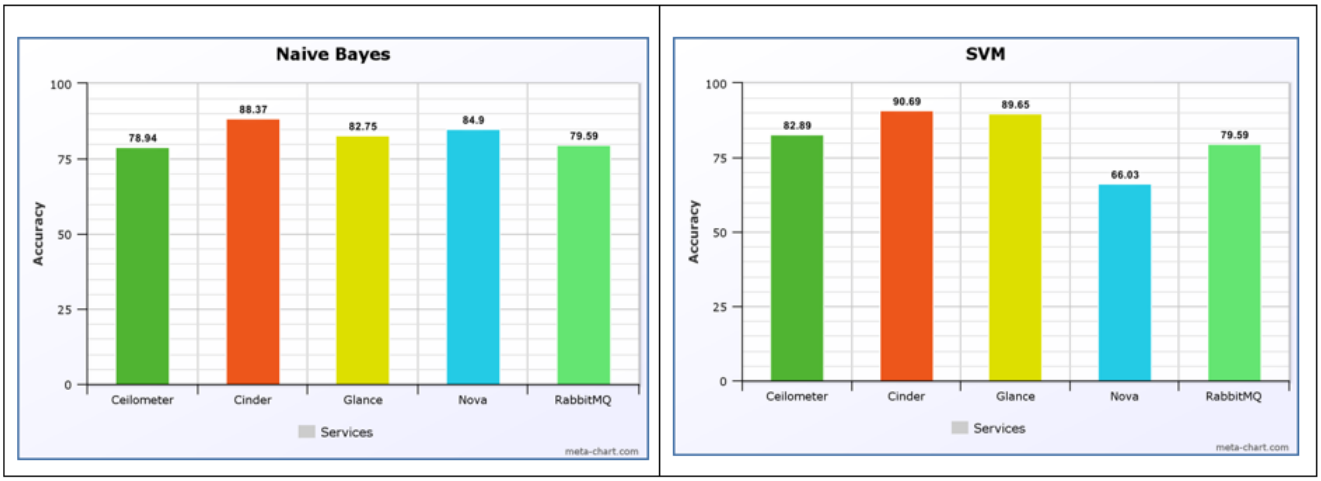
| **Openstack Cloud services** | | | |
|---|---|---|---|
| S.no. | Service | Training events | Testing events |
| 1 | Nova | 546247 | 31040 |
| 2 | Keystone | 65461 | 14871 |
| 3 | Glance | 986545 | 41760 |
| 4 | Ceilometer | 6531108 | 487610 |
| 5 | Cinder | 1751854 | 310971 |
| 6 | Swift | 212495 | 79132 |

From above, we can conclude that for most of the service logs, Random Forest segregated the events with high classification accuracy i.e. when a set of new events are given

```
E E WARNING keystonemiddleware.auth_token [ - ] Authorization
failed for token

E E WARNING keystonemiddleware.auth_token [ - ] Authorization
failed for token

E E INFO nova.osapi_compute.wsgi.server [ - ] 10.0.0.31 `` GET
/v2/bf1aa29cda374722a044b42694e73634/servers/detail ?
all_tenants=True & host=compute HTTP/1.1 '' status : 401 len :
261 time : 0.0135410

E E INFO nova.osapi_compute.wsgi.server [ req-8abe1fa4-b434-4a05-
a725-b38d53f286aa None ] 10.0.0.11 `` GET
/v2/bf1aa29cda374722a044b42694e73634/servers/detail ?
all_tenants=True HTTP/1.1 '' status : 200 len : 8481 time :
0.1817398

E E INFO nova.osapi_compute.wsgi.server [ req-9ec5e857-a462-4edb-
a333-3e42bef7b122 None ] 10.0.0.11 `` GET
/v2/bf1aa29cda374722a044b42694e73634/os-floating-ips HTTP/1.1 ''
status : 200 len : 192 time : 0.0201981
```

Figure 4.6: New events in the Nova are predicted with class labels (A or E)

to the Random Forest, it can correctly classify and identify the events of forensic relevance. This approach benefits the incident handler in arriving at quick hypothesis about the occurred incident.

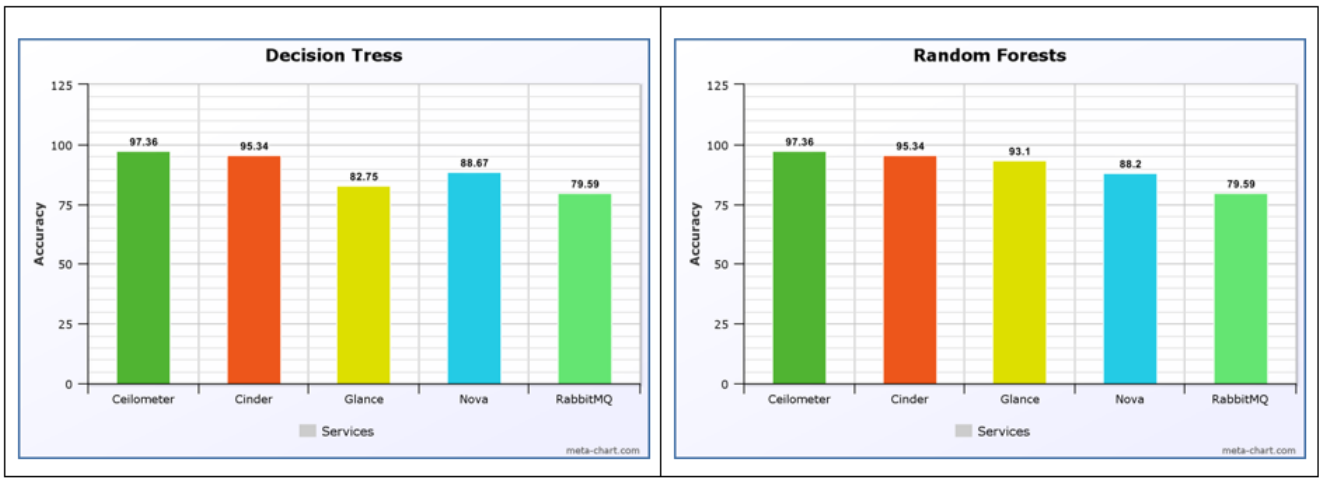



(a) (b)

Figure 4.7: Naive Bayes and SVM: Classification accuracy of various openstack services

## 4.5 Proposed Evidence/Event Aggregation Approaches for Effective Event Reconstruction

It is important to note that, we do not directly focus on the Event Reconstruction phase after segregation and instead we emphasize on a newly added phase called as Aggregation. The main technical difficulty in performing Event Reconstruction is the huge number of events in the target evidence (here, service logs). We reduce the events in the service logs by applying the proposed aggregation algorithms. Once the total number of events to investigate is reduced then it leads to effective hypothesis generation which is one of the main motivations




(a) (b)

Figure 4.8: Decision Tress and Random Forest: Classification accuracy of various openstack services

behind Event Reconstruction. We will explain the role of aggregation for achieving effective reconstruction with a scenario:

### 4.5.1 Scenario to show the role of aggregation for effective event reconstruction

There are usually many services running on each node of cloud systems (here, Openstack cloud) and each has its own set of logs. Moreover, every log has a unique role to play during incident handling. The major services of Openstack and the associated logs are shown in Table 4.3.

Table 4.3: Service logs created for each service of Openstack cloud

| Service | Service Logs |
|---|---|
| Ceilometer | *<-agent-control, -agent-notification, -alarm-evaluator >* |
| Cinder | *<cinder-api.log, cinder-scheduler.log >* |
| Glance | *<glance-api.log, registry.log >* |
| Nova | *<nova-api.log, -cert.log, -conductor.log, novncproxy, nova-scheduler >* |
| Keystone | *<keystone-all.log, keystone-manage.log >* |
| Swift | *<swift-all.log >* |

We observed that each of the above service logs had lots of events. For example, one of the service logs namely ceilometer-collector.log had more than 2 million entries in a short span of time (Table 4.4 shows the event count in the other major service logs) and the count keeps on increasing based on the usage and time. This creates difficulties for the incident handler to analyze all of those events manually.

Table 4.4: Number of alerts in major service logs of Openstack

| Service Log | Alerts |
|---|---|
| ceilometer-agent.log | 106105 |
| ceilometer-alarm-evaluator.log | 336025 |
| ceilometer-collector.log | 2010107 |
| cinder-api.log | 594 |
| cinder-scheduler.log | 6075 |
| nova-compute.log | 35705 |
| libvirtd.log | 128 |

At this point, the incident handler should be provided with the reduced set of events which can be done in two ways (In our work, we consider approach 2 mentioned below because the first one would vary from one incident to the other):

- Approach-1: Identify the occurred incident characteristics based on which the events from the evidences are extracted for selective analysis. This approach cannot be applied when the incident is unknown.
- Approach-2: Irrespective of the incident, the number of events can be reduced using aggregation algorithms. The aggregation will work well for cloud service logs as we have noticed the following common points across all the service logs:
  - Most of the events in service logs are similar (see Figure 4.9). Finding all of them manually is a time consuming task.
  - A single event may generate multiple similar alerts and finding them takes more effort and time (See Figure 4.10)

```
2014-10-26 13:24:09.969 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd14d23a10>,)
2014-10-26 13:24:09.971 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd14d23b50>,)
2014-10-26 13:24:09.973 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd14d23a10>,)
2014-10-26 13:24:10.023 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd14d23ad0>,)
2014-10-26 13:24:10.026 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd14d23b90>,)
2014-10-26 13:24:10.027 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd14d23a10>,)
2014-10-26 13:24:10.028 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd14d23ad0>,)
2014-10-26 13:24:10.029 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd14d23b90>,)
2014-10-26 13:24:10.066 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8e910>,)
2014-10-26 13:24:10.068 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8e9d0>,)
2014-10-26 13:24:10.071 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8e950>,)
2014-10-26 13:24:10.073 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8ebd0>,)
2014-10-26 13:24:10.074 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8eb10>,)
2014-10-26 13:24:10.091 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8e990>,)
2014-10-26 13:24:10.093 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8eb10>,)
2014-10-26 13:24:10.094 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8e910>,)
2014-10-26 13:24:10.096 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8eb50>,)
2014-10-26 13:24:10.098 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd11a8ec10>,)
2014-10-26 13:24:10.100 43852 WARNING ceilometer.transformer.conversions [-] dropping sample with no predecessor: (<ceilometer.sample.Sample object at 0x7fbd14d23b90>,)
```

Figure 4.9: Ceilometer-agent-compute.log of Openstack cloud: Generating multiple similar events

A common aggregation algorithm that is usually applied for logs is Leader-Follower (LF) algorithm [153]. We observed that it cannot be applied on cloud service logs due to the following reasons:

```
2015-10-19 11:11:05.117 1574 ERROR oslo.messaging.rpc.dispatcher [-] Exception during message handling: Zero fixed ips available.
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher Traceback (most recent call last):
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher   File "/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/dispatcher.py",
line 133, in _dispatch_and_reply
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher     incoming.message))
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher   File "/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/dispatcher.py",
line 176, in _dispatch
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher     return self._do_dispatch(endpoint, method, ctxt, args)
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher   File "/usr/lib/python2.7/dist-packages/oslo/messaging/rpc/dispatcher.py",
line 122, in _do_dispatch
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher     result = getattr(endpoint, method)(ctxt, **new_args)
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher   File "/usr/lib/python2.7/dist-packages/nova/network/floating_ips.py", line
119, in allocate_for_instance
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher     **kwargs)
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher   File "/usr/lib/python2.7/dist-packages/nova/network/manager.py", line 515,
in allocate_for_instance
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher     requested_networks=requested_networks)
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher   File "/usr/lib/python2.7/dist-packages/nova/network/manager.py", line 216,
in _allocate_fixed_ips
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher     vpn=vpn, address=address)
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher   File "/usr/lib/python2.7/dist-packages/nova/network/manager.py", line 899,
in allocate_fixed_ip
2015-10-19 11:11:05.117 1574 TRACE oslo.messaging.rpc.dispatcher     quotas.rollback(context)
```

Figure 4.10: Nova-network.log of Openstack cloud: A single event generating multiple similar alerts

- The existing LF is designed to be applied mostly for network logs.
- The attributes used to group various raw alerts (i.e. each individual event) in network logs are not present in cloud service logs. For example, LF checks for events with the same source IP and destination IP and places them in the same group. The cloud service logs will not have these attributes.
- The aggregation is done based on source IP and destination IP addresses which may not be useful in all situations. For example, if the given cluster of computers is compromised in an attack then all those victims are classified under different clusters just because they happen to have different IP addresses.

To address the above limitations of the existing LF algorithm, we propose two new versions of the LF which can aggregate the events in the cloud service logs.

### 4.5.2 Proposed algorithms for aggregating the events in cloud service logs

The following sections describe the proposed algorithms extended from the basic LF i.e. $LF_{V1}$ and $LF_{V2}$.

#### (A) Proposed aggregation algorithm-version1 ($LF_{V1}$)

It is important to note that we consider each event as raw alert and a group of similar raw alerts as one hyper alert. The proposed $LF_{V1}$ algorithm linearly scans through all the raw alerts and groups them into hyper alerts provided they abide by the following rules:

- The Raw Alert should fall within the time window T defined by the incident handler. This decision is made by subtracting the start time of the Raw Alert from the start time of the Hyper Alert.
- The alert type of the Raw Alert should match with that of the Hyper Alert.

- The Raw Alert should not be already assigned to any other Hyper Alert.
- The similarity measure of the Raw Alert and the first element of the Hyper Alert should be within the threshold value given by the incident handler. We observed that similar events contain similar message description. We used Levenshtein Edit Distance for calculating the similarity between the events' message descriptions.

More details about the Algorithm 4 (Leader-Follower (LF) version 1) is given below:

The first Raw Alert is directly made into a Hyper Alert and it is included in the *Added* list. Every other Raw Alert that is not included in the Added list is compared with this Hyper Alert and if all the conditions mentioned previously are satisfied, the Raw Alert is added to the Hyper Alert and the parameters of the Hyper Alert are updated. This Raw Alert is then included in the *Added list*. The same process is repeated for all the Raw Alerts. At the end, if a Raw Alert is not a part of any Hyper Alert, that Raw Alert is made into a Hyper Alert. The format of the Hyper Alert is as follows:

<**Log Name, Hyper ID, Start Time, End Time, Alert Type, Alert Location, Alert Message, Count** >

- Log name: All the events in each service log are aggregated based upon the format of the hyper alert mentioned above. As the attribute *log name* is included in each hyper alert, it becomes easier for the incident handler to identify the associated service log from a hyper alert . The other use of log name attribute is that it can even find the similar hyper alerts generated from different service logs.
- Hyper Id: Used to uniquely identify the hyper alert.
- Start time and End time: The starting and ending time of the raw alerts in the formed hyper alert.
- Alert type: There are various kinds of events generated by service logs. For example, Openstack cloud generates seven types of events (error, warning, critical, info etc.,). The incident handler can easily identify the different types of hyper alerts generated at each location.
- Alert location: If many events are triggered from a single location, it can be easily identified by using the location attribute of the hyper alert.
- Alert Message: This helps the incident handler to know more about the generated hyper alert in the target service log.
- Count: This indicates the number of similar raw alerts in the service log aggregated to form a hyper alert.

Our first version of the Leader Follower Algorithm has the following drawback:

**Algorithm 4** Proposed extended version of Leader Follower algorithm-version1

Input:A set of Raw Alerts $r_1, r_2, r_3, , r_n$, TimeWindow T, $threshold_{max}$
Output: A set of Hyper Alerts $h_1, h_2, h_3, , h_m$ where $m <= n$
$m := 1$
**for** $i \leftarrow 1\ to\ n$ **do**
    **if** $added[i] = false$ **then**
        start_index := i
        start_time := $r_i$.start_time
        count := 1
        added[i] := True
        **for** $j \leftarrow i + 1\ to\ n$ **do**
            **if** $r_j.start\_time - r_i.start\_time <= T and\ added[j] = False$ **then**
                **if** $r_i.type = r_j.type\ and\ levenshtein.distance(r_i.message, r_j.message)\ <= threshold_{max}$ **then**
                    count := count + 1
                    end_index := j
                    end_time := $r_j.start\_time$
                    added[j] := True
                **end if**
            **end if**
        **end for**
        $h_m$ := $(log\_name, m, start\_time, end\_time, r_i.type, r_i.location, r_i.message, count)$
        $m := m + 1$
    **end if**
**end for**

- Because it scans linearly and assigns the Raw Alerts to Hyper Alerts on the go, there is a possibility that an alert is assigned to an incorrect Hyper Alert. For example, a Raw Alert may be more suitable for Hyper Alert 3, because the Raw Alert satisfies all the conditions previously mentioned, it could be assigned to Hyper Alert 1.

To counter this particular problem, we propose another version of LF algorithm i.e. $LF_{V2}$.

**(B) Proposed Leader Follower algorithm-version 2 ($LF_{V2}$)**

The second Leader Follower Algorithm - $LF_{V2}$ that we propose retains the benefits of the $LF_{V1}$ and also addresses the said drawback of $LF_{V1}$. The main difference between $LF_{V1}$ and $LF_{V2}$ is, instead of using the threshold input by the user in one go, $LF_{V2}$ progressively increases the value of the threshold from 1 (Algorithm 5). This gives the algorithm the ability to first group the most similar Raw Alerts and then move on to less similar Raw Alerts. This tackles the drawback of Raw Alerts being assigned to wrong Hyper Alerts to a large extent.

The hyper alert attributes are same in both the versions of LF but the way raw alerts are grouped to form hyper alerts has been changed in $LF_{V2}$ to improve the classification

accuracy.

**Algorithm 5** Proposed extended version of Leader Follower algorithm-version2

Input:A set of Raw Alerts $r_1, r_2, r_3, , r_n$, TimeWindow T, $threshold_{max}$
Output: A set of Hyper Alerts $h_1, h_2, h_3, , h_m$ where $m <= n$
Intermediate Stage: A set of intermediate alerts $z_1, z_2, z_3, ...z_k \; where \; m <= k <= n$
$z1 := r1, z2 := r2, ....zn := rn$
**for** $threshold \leftarrow 1 \; to \; threshold_{max}$ **do**
    k:= Number of elements in z
    **for** $i \leftarrow 1 \; to \; k$ **do**
        **if** $added[i] = false$ **then**
            start_index := i
            start_time := $r_i$.start_time
            count := $z_i.count$
            **for** $j \leftarrow i + 1 \; to \; k$ **do**
                **if** $z_i.start\_time - z_j.start\_time <= T and \; added[j] = False$ **then**
                **if** $z_i.type = z_j.type \; and \; levenshtein.distance(z_i.message, z_j.message) <= threshold$ **then**
                count := count + $z_j.count$
                end_index := max($z_i.endindex, j$)
                end_time := $z_{endindex}.end\_time$
                added[j] := True
                **end if**
                **end if**
            **end for**
        **end if**
    **end for**
**end for**
$m := 1$
**for** $i \leftarrow 1 \; to \; k$ **do**
    $h_m$ := $(log\_name, m, start\_time, end\_time, z_i.type, z_i.location, z_i.message, count)$
    $m := m + 1$
**end for**

**(C) Application of our proposed algorithms for outlier detection**

Outliers are data points that are very far apart from most of the clusters and can be visualized as anomalies. Since we are aggregating the raw alerts to fall into hyper alerts in both the algorithms, the hyper alert less than the minimum support can be considered as an outlier. This outlier detection is highly beneficial to the incident handler in the forensic perspective. A more detailed description of the same is given in the next section.

### 4.5.3 Results and Discussion

We applied the proposed aggregation algorithms on the cloud service logs. We then compared both the approaches and discussed their importance from forensic perspective. At the

end, we have taken the output of our aggregation algorithms to detect outliers.

### Applying modified versions of leader follower algorithms on cloud service logs

As said, the existing Leader-Follower algorithm cannot be applied to the cloud artifacts like service logs. We applied our proposed aggregation algorithm $LF_{V1}$ to the cloud service logs and the observations are:

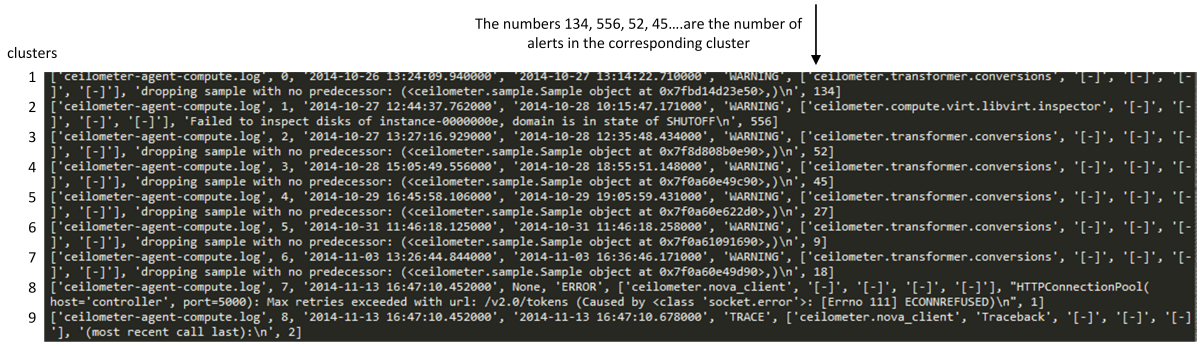



Figure 4.11: Output clusters (hyper alerts) after applying Algorithm 4- $LF_{V1}$

- Figure 4.11 shows the aggregated entries of ceilometer-agent.log. Before applying the Algorithm 4, the number of entries in this log are 106105 and after applying the aggregation, the number is reduced to 559 (Table 4.5).
- The same extent of aggregation was observed even when $LF_{V1}$ was applied on the other service logs (Table 4.5)
- Almost no data loss is observed after the application of $LF_{V1}$

The main advantage of the proposed Algorithm 4 is that the time taken to aggregate the events is less. The same is shown in Table 4.5. The disadvantage with this algorithm is that it may result in a very few improper Hyper Alerts (not correctly classified). We increase the accuracy of the aggregation using our Algorithm 5. We applied it on the same log (ceilometer-agent.log) and the aggregation result obtained is shown in Figure 4.12. The observations after applying $LF_{V1}$ are briefed below:

- The number of raw alerts in ceilometer-agent.log were reduced drastically to form few sets of hyper alerts. The exact statistics are shown in Table 4.6.
- Similar results were observed when $LF_{V2}$ was applied on the other service logs (Table 4.6).

Table 4.5: Number of raw alerts and hyper alerts ($LF_{V1}$) for major service logs of Openstack cloud

| Service Log | Number of raw alerts before aggregation | Number of hyper alerts after applying $LF_{V1}$ | Time taken by $LF_{V1}$ |
|---|---|---|---|
| ceilometer-agent-compute.log | 106105 | 559 | 1.363 sec |
| ceilometer-alarm-evaluator.log | 336025 | 444 | 35.291 sec |
| ceilometer-collector.log | 2010107 | 198 | 153.298 sec |

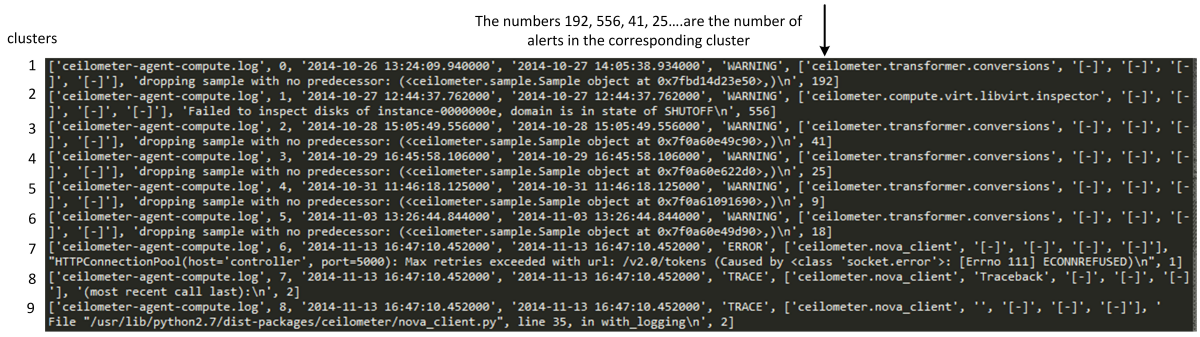



Figure 4.12: Output clusters (hyper alerts) after applying Algorithm 5- $LF_{V2}$

**Comparative discussion on the algorithms-$LF_{V1}$ and $LF_{V2}$**

We compare both the algorithms using two parameters: (1) Classification accuracy (2) Time taken to perform aggregation.

- Classification accuracy of the algorithms

  From both Table 4.5 and Table 4.6, we cannot directly say that, the accuracy of $LF_{V2}$ is more than $LF_{V1}$. For example, the number of events in ceilometer-collector.log before applying the aggregation is more than 2 million. But when applied $LF_{V1}$, it reduced to 198 Hyper Alerts and where as $LF_{V2}$ aggregated them to 205 Hyper Alerts. From this, we can have following hypothesis:

  - H1: $LF_{V1}$ can aggregate events better than $LF_{V2}$
  - H2: $LF_{V2}$ can aggregate events better than $LF_{V1}$

  To find the correct hypothesis, the following arguments are helpful:

  - The difference between the aggregated events of $LF_{V1}$ and $LF_{V2}$ is very small. For example, in the case of ceilometer-collector.log, there is only a difference

Table 4.6: Number of raw alerts and hyper alerts ($LF_{V2}$) for major service logs of Openstack cloud

| Service Log | Number of raw alerts before aggregation | Number of hyper alerts after applying $LF_{V2}$ | Time taken by $LF_{V2}$ |
|---|---|---|---|
| ceilometer-agent-compute.log | 106105 | 572 | 3.291 sec |
| ceilometer-alarm-evaluator.log | 336025 | 445 | 37.275 sec |
| ceilometer-collector.log | 2010107 | 205 | 156.784 sec |

of 7 hyper alerts. If the number of dissimilar events is more, then the algorithm with more number of Hyper Alerts is better. On the other side, if the number of dissimilar events is less then the algorithm which generates lesser number of Hyper Alerts is better.

– On manual inspection, we observed that almost all Openstack cloud service logs had few events which are completely dissimilar. As said, the algorithm with more Hyper Alerts should be better. For ceilometer-collector and other service logs, we noticed that the number of Hyper Alerts for $LF_{V2}$ was greater than that of $LF_{V1}$. From the above statements, we can safely conclude that $LF_{V2}$ is more accurate than $LF_{V1}$.

Our Hypothesis H2 is correct in terms of aggregation accuracy.

- Time Consumption by $LF_{V1}$ and $LF_{V2}$

  Both the versions of $LF_{V1}$ and $LF_{V2}$ are compared in terms of time taken (in sec) to aggregate the events in the service logs and the same is shown in the Figure 4.13. The time consumption by $LF_{V2}$ is slightly more since its aggregation is based on the incremental threshold.

**Identified application for our $LF_{V1}$ and $LF_{V2}$: Outlier detection**

We observed that the results of $LF_{V1}$ and $LF_{V2}$ aggregation can be used for outlier detection. There are few existing log clustering tools which can detect outliers. Most of those tools, process the log files at word level i.e. they first split the entire document in to set of words and then cluster all the lines that conform to the cluster description. Any line whose support is less than the given support value is considered an outlier. But most of them when applied to cloud service logs will give inappropriate outliers as it might repeat the same

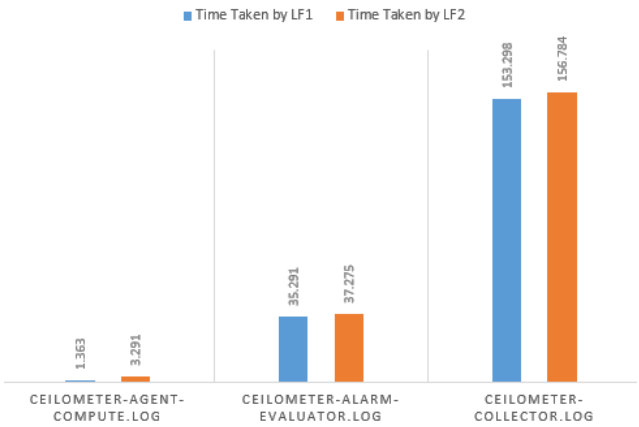



Figure 4.13: Time consumed by $LF_{V1}$ and $LF_{V2}$ for aggregation

outlier multiple times. It may do so because the message descriptions are not the same. But actually they are similar to each other and should be grouped into one record. For example, we used SLCT (a log clustering tool) to detect outliers from the cloud service logs and the same is shown in Figure 4.14. From this, it is very obvious that many duplicate outliers exist and it is a time consuming task for the incident handler to find all those manually.

```
2014-11-13 16:47:10.452 1471 ERROR ceilometer.nova_client [-]
HTTPConnectionPool(host='controller', port=5000): Max retries exceeded
with url: /v2.0/tokens (Caused by <class 'socket.error'>: [Errno 111]
ECONNREFUSED)
2014-11-13 16:47:10.678 1471 ERROR ceilometer.agent [-] Unable to
discover resources: None: Max retries exceeded with url: /v2.0/tokens
(Caused by redirect)
2014-11-24 09:56:38.438 1512 ERROR ceilometer.nova_client [-]
HTTPConnectionPool(host='controller', port=5000): Max retries exceeded
with url: /v2.0/tokens (Caused by <class 'socket.error'>: [Errno 113]
EHOSTUNREACH)
2014-11-24 09:56:38.461 1512 ERROR ceilometer.agent [-] Unable to
discover resources: None: Max retries exceeded with url: /v2.0/tokens
(Caused by redirect)
2014-11-24 10:06:38.530 1512 ERROR ceilometer.nova_client [-]
HTTPConnectionPool(host='controller', port=5000): Max retries exceeded
with url: /v2.0/tokens (Caused by <class 'socket.error'>: [Errno 113]
EHOSTUNREACH)
2014-11-24 10:06:38.532 1512 ERROR ceilometer.agent [-] Unable to
discover resources: None: Max retries exceeded with url: /v2.0/tokens
(Caused by redirect)
2015-04-21 13:03:11.649 1711 ERROR ceilometer.nova_client [-]
HTTPConnectionPool(host='controller', port=5000): Max retries exceeded
with url: /v2.0/tokens (Caused by <class 'socket.error'>: [Errno 111]
ECONNREFUSED)
```

Figure 4.14: Oultliers identified by log clustering tool (SLCT)

We now propose an alternate method of finding the outliers directly from the Hyper Alerts generated by our $LF_{V1}$ and $LF_{V2}$. In this case, the incident handler can have his own definition as to how frequent an alert should be so that it is not classified as an outlier. This works on the output of our aggregation algorithms and so no additional computation time is needed.

The output of the algorithm(s) is passed as input to a filter which removes all the Hyper Alerts whose count value is more than minimum support to identify outliers. It also has the ability to restrict the output to only one type. These features were missing from the existing log clustering tools like SLCT. The outliers identified by our $LF_{V2}$ is shown in Figure 4.15 (here, we defined the hyper alerts with minsup <= 10 as outliers. It is important to note

that, deciding the count to define as outlier can vary depending on the incident handler's requirement). We conducted several experiments to decide the minsup value for detecting the outliers effectively (Figure 4.16).

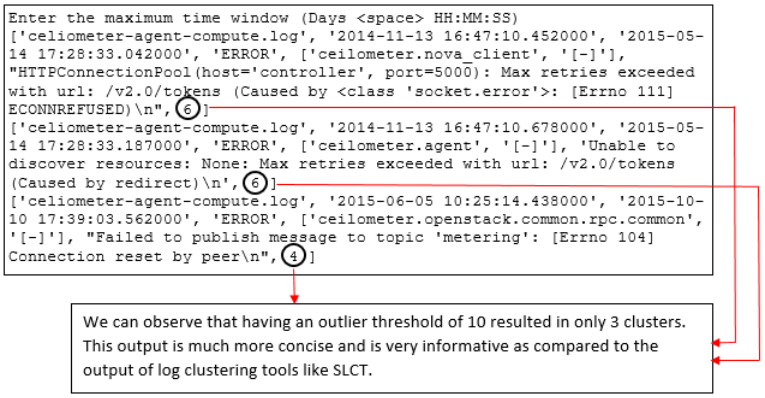



Figure 4.15: Outlier detection by the proposed aggregation algorithms

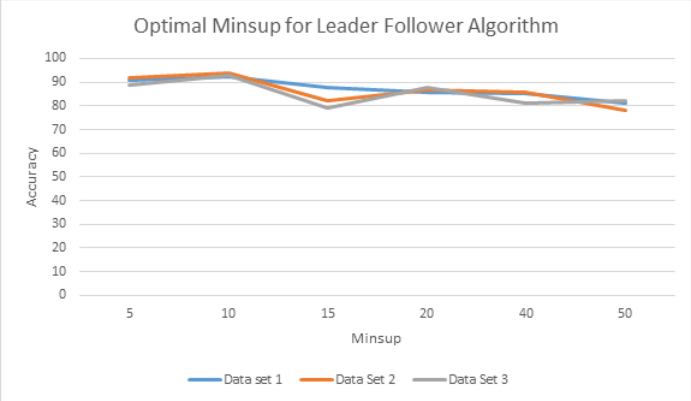



Figure 4.16: Deciding the minsup value for detecting the outliers effectively

Finally, when these reduced set of events are projected to any timeline tool then it would lead to effective hypothesis generation which is the main motivation behind event reconstruction.

### Advantages of our Aggregation algorithms-$LF_{V1}$ and $LF_{V2}$

The proposed aggregation algorithms for cloud service logs have the following advantages:

- Less time and Effort: If the number of events on timeline is high then the time to perform event reconstruction will also be high because, multiple phases like evidence examination, role classification, hypothesis testing and event sequencing has to be followed for Event Reconstruction [16]. The effort involved in going through all these phases can be reduced when the initial total number of events in the corresponding evidence are reduced which we have done through our aggregation algorithms.

- Almost no data loss: In incident handling, performing analysis on the incomplete evidence would give wrong findings. In the name of reducing the number of events, the data loss should not happen and this is taken care by our aggregation algorithms.
- Legally admissible: Immediately after the incident, the intruder will try to tamper the existing evidences. In such a scenario, analysis on the modified evidence is not legally acceptable. In this chapter, we considered cloud service logs which record events at the cloud host systems and the chances of accessing and modifying them by the intruder is very less. The integrity of the evidence (here service logs) is by default maintained, the logical findings made out of it would also be admissible in the court of law.
- Generic in nature: Our aggregation algorithms do not work on the basis of specific attributes like IP addresses. The attribute we considered for aggregation is *message description* as it will be the common attribute in most of the logs. Our approach can even be applied to traditional logs.
- Effective event reconstruction: The process followed for ER should be able to generate correct hypothesis in less number of iterations. This can be achieved when the number of events is reduced without much data loss. The same is achieved by the proposed aggregation algorithms with which event reconstruction can also be done faster.

## 4.6 Summary

Event Reconstruction in incident handling would help the incident handler in various aspects. In this work, we considered one of those capabilities namely *Hypothesis Generation*. But performing Hypothesis Generation in cloud is challenging due to its multi-tenancy and the huge scale of events generated per unit time. We proposed a model- SEASER which can generate effective hypothesis. This is made possible by segregating and reducing the number of events without much data loss. The model proposes segregation and aggregation before initiating the process of hypothesis generation. We devised various approaches for segregation (parameter based, session based and ML based) and validated them using Openstack cloud service logs. We proposed two new aggregation algorithms-$LF_{V1}, LF_{V2}$ as existing Leader-Follower algorithm (LF) cannot be applied on the cloud artifacts. We also tested our aggregation algorithms on the Openstack cloud service logs independently and observed that there is a high extent of event reduction without much data loss. Once the total set of events in the cloud service logs is aggregated, those can be given as input to any timeline tool for hypothesis generation.

The work in this chapter is accepted and published work in [PUB5], [PUB7] and [PUB8] (refer page no. 136-137). In the next chapter, we discuss the proposed approaches for analyzing VM snapshots for incident handling.

# Chapter 5

# Incident Handling using Cloud VM Snapshot Objects

*"Faster IT Innovation is possible by providing open source cloud platforms"*

-Openstack

In Cloud Incident Handling, virtual machine snapshots act as one of the richest source of evidences and can aid the incident handler for getting accurate logical findings. Various cloud service providers like Azure and Openstack facilitate the end-users to take snapshots. It is important to note that, this feature was introduced to address the data backup challenges and not to meet the incident handling requirements of cloud [154][155]. This points to the fact that there is a need to devise *forensic based approaches for VM snapshot acquisition and analysis to handle cloud incidents*.

In reality, snapshots for a cloud VM may not always exist. To increase their availability, we suggest the use of Cloud Forensic Readiness (CFR) models in which the possible evidences (here, snapshots) are collected before the actual incident or while the incident is happening. The captured snapshots have to be transferred to the incident handler's environment via a network. Since the cloud VM snapshots will be generally of huge size, transferring them elsewhere for processing may lead to the problem of data gravity (i.e. the network and analysis overhead for incident handling is referred as data gravity). We handle this problem by designing a framework named SNAPS which we arrived based on the existing spatio-temporal models and then customized to suit for effective Cloud Incident Handling. The motivation behind proposing SNAPS is to generate *provenance* for each object/file in the target virtual machine using its multiple snapshots. Also, SNAPS can be used to address various forensic based incident handling challenges starting from simple to complex ones.

On the other side, SNAPS cannot retrieve the deleted objects from the snapshots which is generally of high relevance during incident handling. We used Natural Language Processing

(NLP) techniques to recover deleted objects from the snapshots.

## 5.1 Introduction

Basically, the feature of cloud VM snapshot is introduced for the backup purposes and not intended to solve the incident handling challenges. Only limited work exists on using the cloud VM snapshots for forensic based incident handling. In [156], the authors used existing VNsnap model and then modified it to take the periodic snapshots during the time of attack. The advantages of their approach are:

- They ensured the integrity of the acquired evidences which is important in the court of law.
- The downtime of the cloud VM while capturing snapshots is less.
- They have used fuzzy clustering techniques to increase the detection accuracy of cloud attacks.

The drawbacks of their work are:

- Their approach lacks practical implementation and the questions like, how to analyze a captured cloud VM snapshot is not discussed.
- The role of the CSP in capturing the snapshots is not mentioned without which forensic based incident handling in the cloud is highly difficult.
- They did not validate their approach using any evaluation parameters like, data loss of the evidence.
- The possibility of retrieving deleted objects from the cloud VM snapshots is not assessed.

In [158], the authors developed forensic prototypes which can acquire data from public cloud environments. The contributions and capabilities of their prototypes are:

- They concluded that the client side evidences will not help the incident handler to conduct cloud forensic analysis.
- They built three interesting forensic prototypes for cloud drive acquisition (kumodd), Google docs Acquisition (kumodocs) and cloud drive data preservation (kumofs).

The drawbacks with their approach are:

- The tool works only for the SaaS models and cannot be applied to IaaS cloud environment, the service model which poses maximum challenges for incident handling.

- The challenges of data gravity were not handled. This is especially important when the huge sized evidences have to be acquired and then transferred to the incident handler's environment.

In [157], the authors experimented on traditional snapshot acquisition techniques and proposed a distributed framework for snapshot acquisition ensuring its admissibility. They did not mention any practical findings of their approach.

**Deduction:** Based on the above literature, it is evident that there is no approach for increasing the availability of snapshots considering the analysis time and network overhead. Accordingly, the framed problem statements are briefed below:

- The availability of the cloud virtual machine snapshots should be improved.
- The snapshots at the cloud side should be transferred to the incident handler's environment by addressing the issue of data gravity.
- Analyzing multiple snapshots is a time consuming job. The incident handler's time and effort for snapshot analysis should be reduced.
- Approaches for recovering the deleted objects from the cloud VM snapshots should be devised.

The rest of the chapter deals with addressing the above issues.

## 5.2 Improving the Availability of the Cloud Virtual Machine Snapshots

Cloud Forensic Readiness (CFR) models present ways to increase the evidence availability for forensic based incident handling [159][160]. The core idea of CFR is to acquire the possible evidences (here, VM snapshots) before the incident or while the incident is happening. There are several issues with respect to these models:

- The monitoring capabilities of most of the existing models are deployed at the VM level. This can make the intruder to alter/stop the running CFR model.
- Some CFR models rely on the user to collect the evidences which may be an overhead to the user.
- Many practical details about the suspicious activity detection in a VM have not been discussed.

To overcome the above, we propose an improved Cloud Forensic Readiness Model (iCFR) [161]. It is deployed at the hypervisor level and uses the Virtual Machine Introspection (VMI) technique to detect suspicious activities of the target VM. It is important

to note that, iCFR also uses the cloud service logs stored in the hypervisor to analyze the VM level activities. Once the suspicious activity is detected, then immediately the target evidences (here, multiple VM snapshots) are captured till the incident impact is reduced and finally transferred to the incident handler's isolated environment via a network for analysis.

The definition of a suspicious activity may vary depending on the organizational policies and cloud service provider's terms and conditions. In general, the activities which can be treated as suspicious are, over resource consumption, violation of security policies, an attempt to obtain unauthorized access, modifications without the actual owner instruction or knowledge etc. For illustration, we applied iCFR to know whether the activity of over resource consumption by a VM is suspicious or not.

.

### 5.2.1 Detecting the suspicious resource consumption of virtual machines in the cloud environment

For this, the proposed iCFR uses the default metering service running in the cloud hypervisor and measures the resources consumed by the target VM. If the VM resources are over utilized then we may consider the activities running in it as suspicious. But in all the cases, over utilization should not be concluded as suspicious due to the rapid elasticity property of the cloud. To reduce the false positives, iCFR checks multiple conditions i.e.

- We consider the recent resource consumption history of a target VM and if there is a drastic deviation from its average resource consumption then we may consider that as suspicious (Base Condition).

  Along with the base condition, we check many additional conditions and some of them are presented below:

- We check the extent of VM errors generated during the same time in the cloud service logs. We consider this as one of the supporting conditions because, when the over resource consumption in cloud is non-suspicious, service logs may not generate many error events. If the number of errors is more than usual, then with more belief, one can confirm the over resource consumption as suspicious.

- If the same VM is used from different geographical locations at the same time then with more support one can confirm that the resource consumption is suspicious.

- We also check whether there is an over resource consumption or not, especially after observing multiple login failures from the same system. If they are present, it indicates that the intruder had compromised the original cloud user account and he (intruder) started over-consuming the resources in a short time.

Additional rules can be framed in accordance with the underlying cloud SLAs to reinforce the suspiciousness of the activity. iCFR selects and checks the rules depending on the suspicious activity.

### 5.2.2 Scenario testing: over-resource consumption of a VM in openstack cloud

*Experimental Setup*: We setup Openstack private IaaS cloud with four rack servers, each with a high end configuration i.e. server 1: Controller node, server 2: Compute node, server 3: Storage node, server 4: Object storage node (Appendix I).

Each VM will be billed and charged according to its usage. In Openstack, it is achieved by the ceilometer service. The testing is performed on a VM named *test2_010316_clr* which is present in the demo user account- john. We queried various service logs pertaining to the ceilometer service and then calculated the average resource consumption of *test2_010316_clr* VM. In Figure 5.1, the middle line shows the average CPU utilization of that VM, the first line shows the upper standard deviation and the bottom line shows the lower standard deviation. We calculated the read-write cycles during the same time period and observed a deviation (Figure 5.2). We observed that, these deviations are more than the threshold value and this indicated that, the resources which are over consumed may be used for performing suspicious activities. To confirm this, we checked multiple conditions i.e.

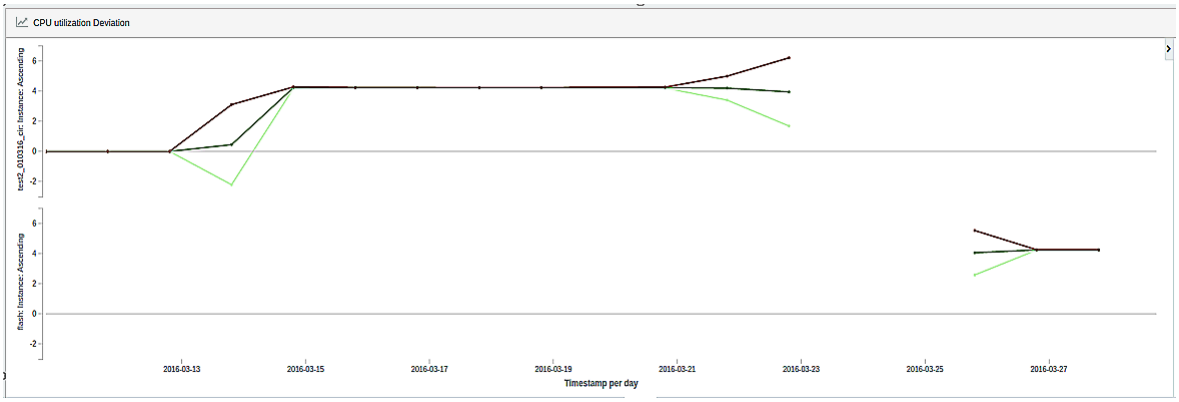



Figure 5.1: CPU utilization of the target VM

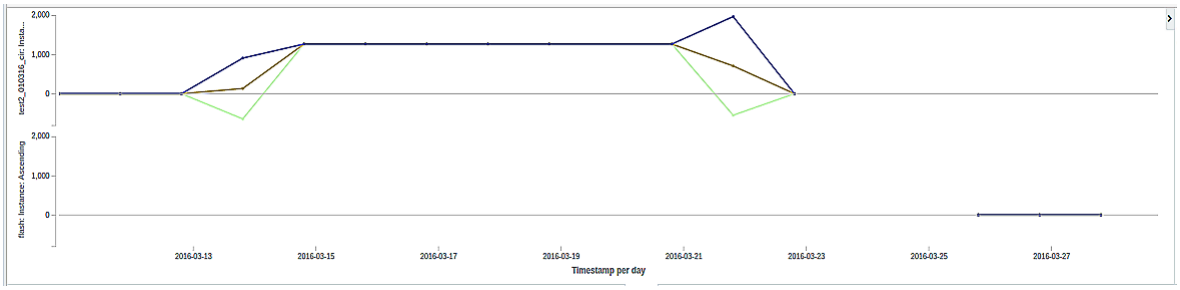



Figure 5.2: Read-write cycles calculated for the target VM

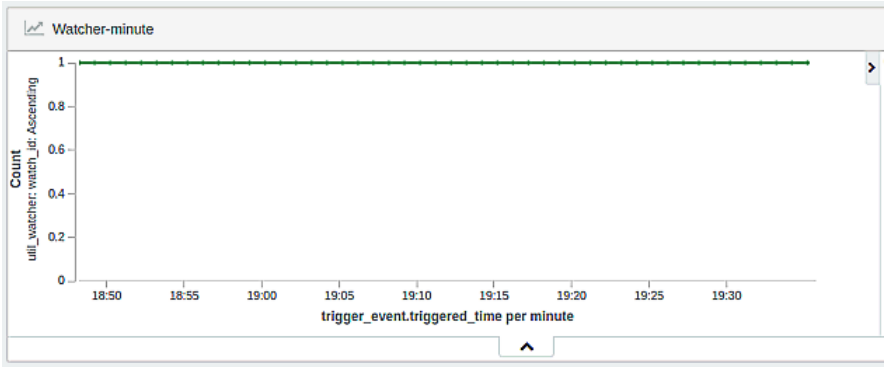



Figure 5.3: No. of non-target VM error events in the service logs

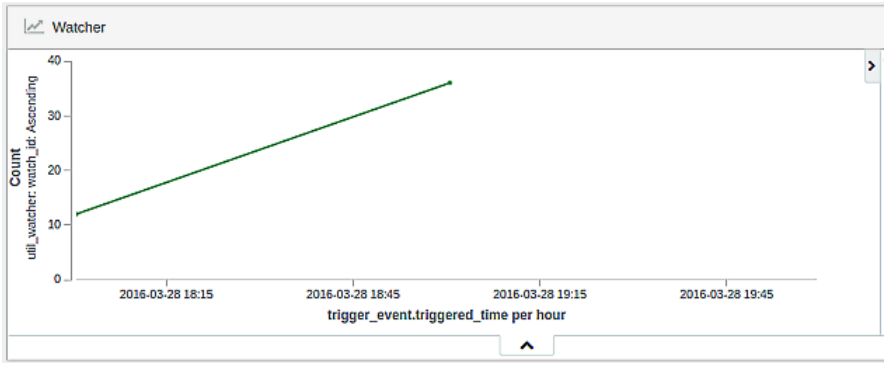



Figure 5.4: No. of target VM error events increased drastically in service logs

*Error events in the cloud service logs:* When there is an increase in the resource consumption, we observed that for a non-victim VM, the error events are stable (Figure 5.3). But for a suspicious VM (i.e. here, *test2_010316_clr*), the number of error events increased drastically (Figure 5.4). Since this deviation is unusual when compared to the history of *test2_010316_clr* VM, the argument of considering over resource utilization as suspicious can be strengthened.

*Multiple logins at the same time:* We initially identified the system IP from which the cloud VM (*test2_010316_clr*) is being accessed. From the IP-172.16.6.186 ($IP_1$), there are a few login failures. After receiving a complaint by the victim from $IP_1$ regarding failed login attempts, we observed that there are multiple successful logins from different systems with IPs-172.16.6.193 ($IP_2$), 172.16.6.112 ($IP_3$), 172.16.6.124 ($IP_4$). This confirms the Incident Handler that *test2_010316_clr* VM was compromised and misused by multiple hosts, each having different IPs. We arrive at this finding by normalizing the format of various cloud service logs. We integrated all of those to form a single log revealing the details as shown in Figure 5.5.

After verifying multiple conditions, we confirm whether an over consumption of resources may be considered as suspicious or not. Thus, the number of false positives is reduced. This is a scenario to detect a suspicious activity using the conditions mentioned above. Defining multiple customized conditions will help the incident handler detect any

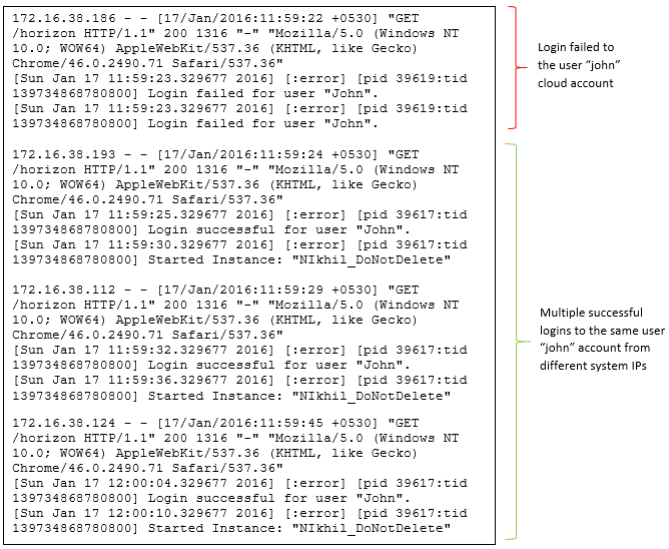



Figure 5.5: Log events showing multiple login failures

suspicious activity. Once a suspicious activity is detected, we start capturing multiple snapshots of the corresponding VM at frequent intervals till the effect of the incident is reduced.

If the incident handler has multiple snapshots of the target VM then it will definitely aid in performing faster in-depth analysis and generate better logical findings about the incident. In this context, we identified that the following issues need to be addressed:

- Data gravity: The analysis overhead involved for target VM snapshots should be minimized by considering the network overhead aspects (Section 5.3).
- Legally admissible: The evidences transferred to the incident handler's infrastructure should be complete and reliable (Section 5.3).

To achieve both the above, we propose a model namely SNAPS- Snapshot based Provenance System which we describe in the next section.

## 5.3 Proposed Model to Handle Data Gravity of Cloud VM Snapshots

SNAPS contains various modules like, Monitoring module (deployed at CSP), Snapshot accumulator (deployed at CSP), spatio-temporal modules-iRSM (deployed at CSP) and iSTSM (deployed at the incident handler side), Regeneration module (deployed at the incident handler side). The major modules of it are described as below.

### 5.3.1 Modules of the SNAPS

- Monitoring Module: This collects the data regarding the VM activities. We use virtual machine introspection (VMI) technique and cloud service logs to collect the target

VM events. Those are then analyzed to check for suspicious events. The advantage of this is, it is difficult for the intruder to alter/stop the module functionality as all the monitoring capabilities of it are from the hypervisor which is generally inaccessible to any user.

- Snapshot Accumulator: Once a suspicious activity is detected in a VM, the snapshots of it will be captured at frequent intervals of time and the frequency depends on the severity of the incident. Each of those snapshots will be stored in a sandbox environment (It is an isolated space to which only the CSP entities have access). Before sending the snapshots to the incident handler's environment they are passed to the spatio-temporal module (iRSM) deployed at the cloud end. The details about iRSM is discussed in the following subsection.

### 5.3.2 Spatio-Temporal model for efficient storage of cloud VM snapshots

In general, spatial temporal models are used when the data has both the time and space attributes. Research in this domain had started three decades ago and most of the research papers used these models to solve the environmental issues. For example, the occurrence of acid rains in New York (USA) has been predicted along with the possible location and time [162]. Various spatio-temporal models were proposed in the literature [163][164]. In this chapter, we use two of those models namely Refined Snapshot Model (RSM) and Simple Time Stamping Model (STSM) for efficient storage and analysis of the cloud VM snapshots.

A brief on the two models is given below:

- Refined Snapshot Model (RSM) [163]: Initial snapshot is stored completely. The sequence of changes to it are stored as Delta objects. To get the current state, each delta object has to be applied on its previous complete snapshot. The existing literature in spatio-temporal domain considered *snapshot* as current state showing the geographical location. In our case, *snapshot* refers to the current state of the cloud virtual machine.

- Simple Time Stamping Model (STSM) [164]: Each object in this model is tagged with its creation time and cessation time. It also maintains links to the preceding and succeeding versions of the object. The advantage with this model is that the state of any object can be retrieved with respective to time.

We identified that the existing spatio-temporal models (RSM and STSM) cannot effectively handle the issue of data gravity due to the following reasons.

- As the number of VM snapshots in the sandbox goes higher, the number of delta objects which needs to get transferred to the incident handler will also go high. In

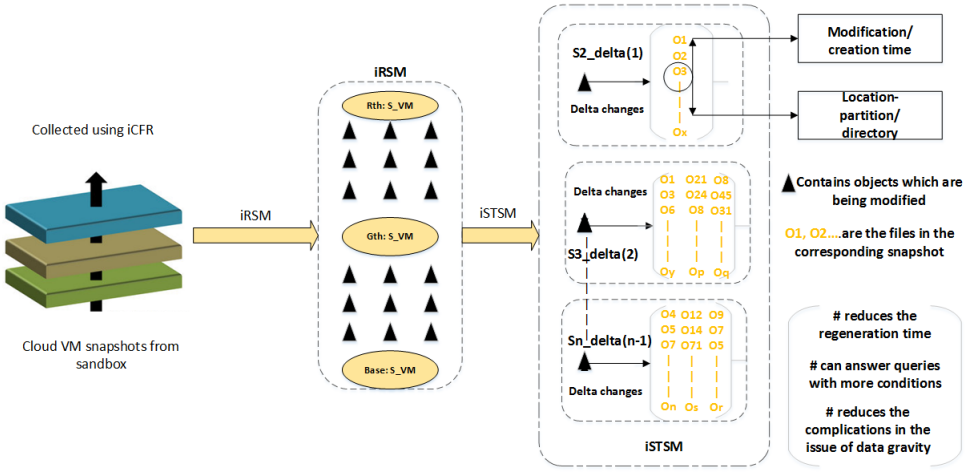



Figure 5.6: Organization of iRSM and iSTSM Modules

this connection, we observed that the time to reconstruct the recent snapshot from its delta file is time consuming as from the base snapshot to the current delta file, a sequence of multiple regenerations have to be performed. To reduce the time spent, we modified the basic RSM and proposed an improved version of RSM (i.e. iRSM).

- Similarly we observed that, when STSM is directly applied to the cloud VM snapshots it cannot answer the forensically relevant questions on incident handling. For example, STSM only stores the time associated with an object. Due to this, the incident handler can only get the objects which are modified/deleted at a certain point of time. In most of the cases, the incident handler is interested in both the spatial and temporal attributes. We propose a new version from the basic STSM and we call it as iSTSM where each object in it is associated with both the time and location attributes.

The organization of both the iRSM and iSTSM is shown in Figure 5.6. All the target VM snapshots in the cloud sandbox cannot be directly sent to the incident handler as it can complicate the issue of data gravity. All of them were passed through iRSM module which would keep multiple full snapshots instead of only one. The incident handler then receives *multiple full snapshots and several delta files* and finally complete snapshots from the corresponding delta file(s) are regenerated.

### 5.3.3 Results and Discussion

Using the Openstack cloud test bed (Appendix I), we created two instances. One is configured with tinyOS and the other virtual machine holds Ubuntu Server. To illustrate the proposed model-SNAPS, we present a simple investigative scenario.

#### Scenario description

Through the proposed iCFR model, we captured several snapshots of the TinyOS VM after observing a suspicious activity in it. All those snapshots are initially stored in the sandbox

of the cloud. After the suspicious activity was confirmed then the spatio-temporal model-iRSM is applied.

We compare our proposed approach (A3) with the existing approaches (A1 and A2) as given below.

- Approach-1 (A1): In this, the captured snapshots will be stored in its original form.
- Approach-2 (A2): Only one full snapshot is stored in the sandbox and remaining all snapshots of the same VM are stored as delta objects.
- Approach-3 (A3-Proposed approach (SNAPS-iRSM)): Instead of one full snapshot, we will have multiple complete snapshots which are captured after every threshold number of delta objects.

For comparison, we have taken eight snapshots of both the virtual machines separately. If the incident handler's interest is to regenerate and analyze the recent snapshot (worst case) then the various comparison parameters are:

*Regeneration Time*: A3 has lesser regeneration time than A2 due to its intermediate complete snapshots (Figure 5.7). It is important to note that, though A1 has the least regeneration time still is not an efficient approach due to its higher space occupancy.

*Space Consumption*: There is not much difference in the space occupancy of the snapshots between A2 and A3 (Figure 5.8). The reason is, among the eight snapshots there is only one additional full snapshot in A3 when compared with A2. The threshold for taking full snapshots is three. The threshold value can change depending on the number of snapshots for that virtual machine.

*Data Loss*: In the name of handling the data gravity challenges, the evidence (here snapshot) should not be modified as it diverts the incident handler to give inaccurate logical findings about the occurred incident. For all the three approaches, the snapshots and delta objects were transferred from the cloud sandbox to the incident handler's infrastructure. We observed that A1 had more data loss when compared with A2 and A3 (Figure 5.9).

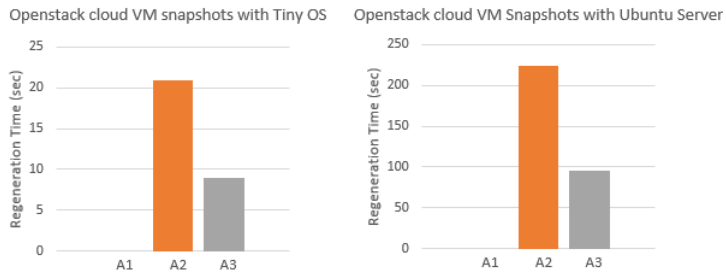



Figure 5.7: Snapshot regeneration time for all the three approaches

From the above discussion, it is clear that A3 is better than others in terms of handling data gravity issue with less space consumption, less regeneration time and less data loss.

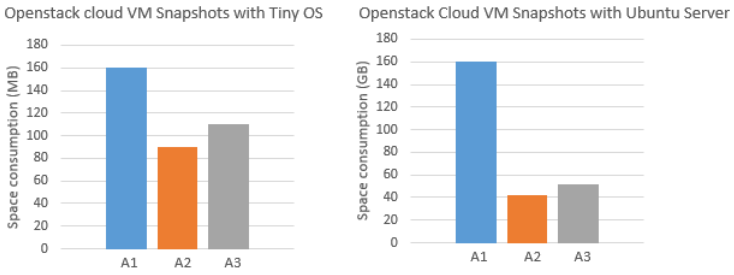



Figure 5.8: Space consumed by the VM snapshots for all the three approaches

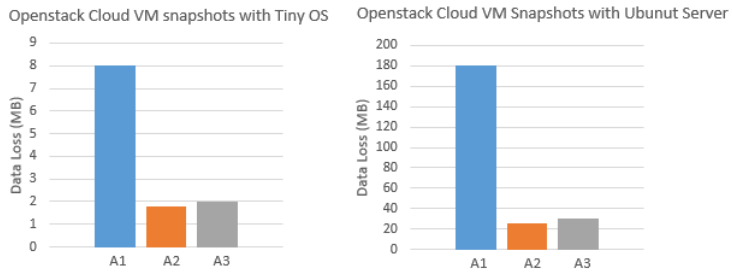



Figure 5.9: Data loss incurred during snapshot transfer for all the three approaches

### Analyzing a single regenerated snapshot

Once snapshots are transferred from cloud infrastructure to the incident handler's environment using iRSM, then the two common tasks in the incident handling are: (1) Automatically regenerate a snapshot from its delta file such that the entire process should be legally admissible (2) Perform analysis on the regenerated snapshot.

*Ensuring Evidence Admissibility:* The acquired evidences should be legally admissible; otherwise the logical findings made out of them will not withstand in the court of law. Two main aspects to confirm evidence admissibility is completeness and preserving integrity. These properties are followed for all the three approaches (A1, A2 and A3).

*Snapshot Analysis*: Using the forensic tools, only a single snapshot can be analyzed at a time. In reality, the incident handler may want to analyze multiple snapshots simultaneously and in that situation, the time for performing forensic based incident analysis would further increase. To address this, we proceed in the following two stages.

- Building provenance: It represents the history of an object. We propose that, building provenance from multiple snapshots would reduce the time for analyzing multiple snapshots (Section 5.4).

- Generating recommendations: Using the provenance system built, we generate the recommendations and each of these will help the incident handler to arrive at quick logical findings about the incident or the suspicious activity (Section 5.5).

The above two modules are deployed at the incident handler's end. We use the other spatio-temporal model- iSTSM for building the provenance.

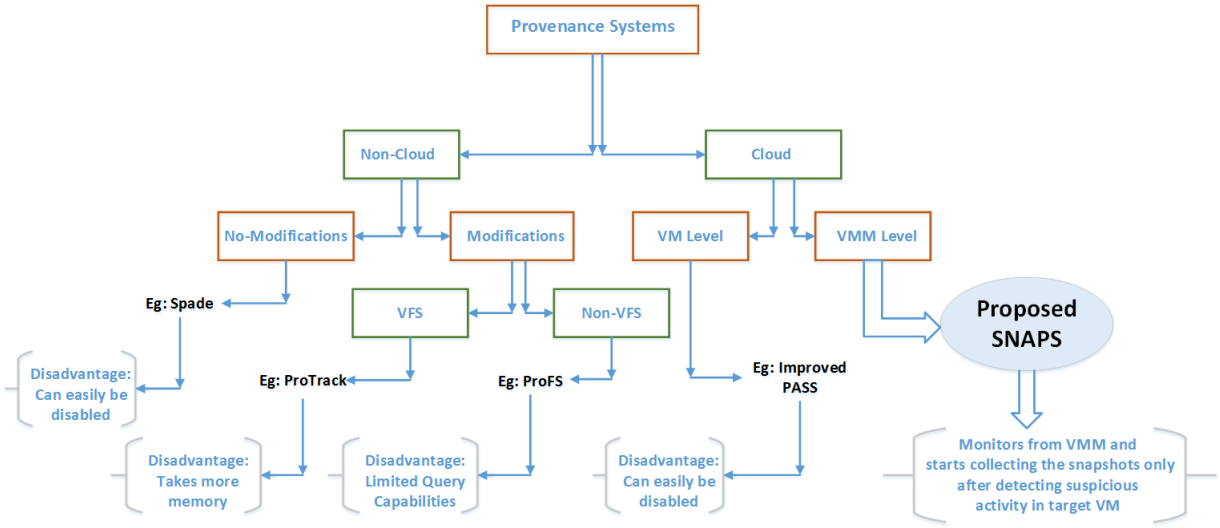



Figure 5.10: The proposed high level taxonomy for the existing provenance systems

## 5.4 Building a Provenance aware System for Analyzing the Cloud VM Snapshots

Provenance has a lot of forensic applications for handling the occurred incidents in the target environment [165]. We emphasize on two aspects of it i.e. *to know how an object reached to the current state* and *to improve the quality of search*. We built provenance using the objects in multiple snapshots.

### 5.4.1 Proposed taxonomy for existing provenance systems for cloud and non-cloud environments

Upon extensive literature, we proposed a taxonomy of the existing provenance systems and the same is shown in Figure 5.10. Provenance systems exist for both non-cloud and cloud environments. To collect provenance data, some provenance systems mandate the requirement of modifying the underlying file system (Eg: ProTrack, ProFS) and some other systems can collect the provenance data without any requirement of changing the target file system (Eg: SPADE) [165]. ProTrack is a provenance data collector and builds Versioning File System (VFS) [166]. Most of the VFS based provenance systems suffer from memory overhead. ProFS is a non-VFS provenance system which does not suffer with memory overhead issues but it cannot provide effective querying capabilities [167].

Similarly, provenance systems can also be built for the cloud environment. An initial effort in this direction is made by authors in [168]. They have customized the existing Provenance Aware Storage System (PASS) to suit the cloud requirements. Their provenance collector has to be configured and installed in the target VM. This may not be always possible as the user of a VM may not be interested in the provenance data. Moreover, if the intruder attacking a VM finds a provenance system then he/she can disable it or atleast

delete/modify the collected data. Observing these drawbacks, SNAPS proposes to capture and acquire the VM snapshots from the hypervisor level and then build the provenance for each object in multiple snapshots.

### 5.4.2 Novelty of proposed SNAPS

- SNAPS is a first of its kind solution which builds provenance system from snapshots by considering various forensic aspects to handle cloud incidents.
- It is difficult for the intruder to disable the provenance system as it is hosted at VMM level.
- It facilitates efficient querying capabilities and can answer several forensically interesting questions:
  - Get all the objects modified between time $t_x$ and $t_y$ in the location $Loc_t$.
  - Find the most commonly modified object across multiple snapshots in the location $Loc_p$.
  - Identify all the copied files in various locations within the given time range.
  - The location which had undergone more changes etc.,
- Analyzing multiple snapshots is a time consuming task. By incorporating a recommendation engine, the proposed SNAPS technique reduces the time spent on analyzing the snapshots.

### 5.4.3 SNAPS approach to build provenance

The following terms are worth mentioning:

- Provenance chains: Each object's history is represented as a provenance chain. All the nodes in a provenance chain represents the same object. A node to an $O_i$ provenance chain is added when there is a change in $O_i$ 's metadata. The provenance chain of $O_i$ will stop growing when it is deleted from the target VM.
- Provenance graph: Combining provenance chains of all objects creates a provenance graph. Generally provenance graph is a DAG but we call the provenance graph generated from iSTSM as *informal DAG* [168]. The reason of calling it as informal is, our provenance graph does not represent a cause and effect relation but can represent only the effect.

### Provenance Builder

Once the set of snapshots and delta files are transferred to the incident handler's machine, then we regenerate the complete snapshots. During the process of regeneration, we build the provenance chain for each object using the proposed model iSTSM (Figure 5.11). Once the provenance system is constructed, any existing object can be queried to get its provenance chain using the query engine. The output of this engine is given to the recommendation module where it will give a set of logical clues about the object which was queried. The exact steps for provenance generation is given in Algorithm 6.

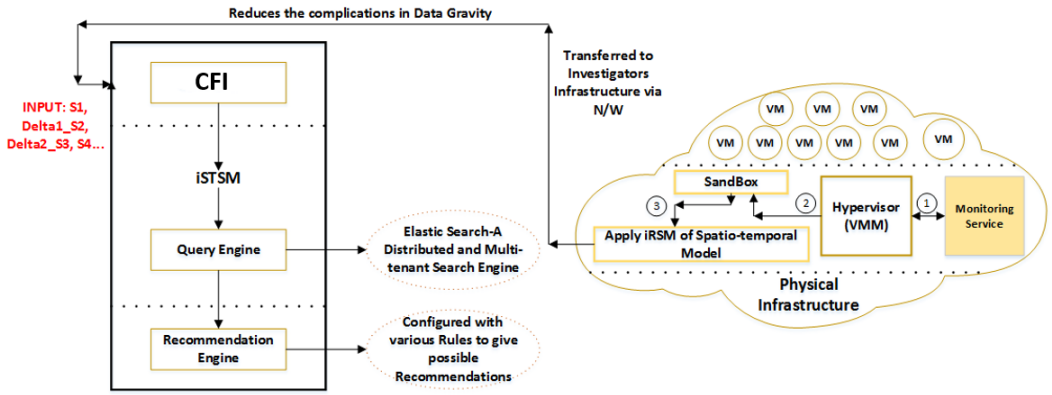



Figure 5.11: Building the provenance system from multiple snapshots acquired from the cloud

**Algorithm 6** Building the provenance chain for each queried object

**Input:** A set of snapshots and delta files i.e. S1, S2_Delta1, S3_Delta2, S4, S5_Delta3 etc. and say incident handler wants provenance data for object $O_i$
**Output:** Provenance Chain for $O_i$
**for each** Snapshot $S_i, S_{i+1}$ **do** ▷ Got $S_{i+1}$ from its delta file and $S_i$
    Search for $O_i$ metadata entry in $S_i$
    **if** $O_i$ metadata exists **then**
        $M_i \leftarrow Retrieve\ O_i\ Modified/Created\ Time$;
    **end if** ▷ $Else, Assign\ M_i \leftarrow NULL$
    Search for $O_i$ metadata entry in $S_{i+1}$
    **if** $O_i$ metadata exists **then**
        $M_j \leftarrow Retrieve\ O_i\ Modified/Created\ Time$;
    **end if** ▷ $Else, Assign\ M_j \leftarrow NULL$
    **if** $M_i == M_j$ **then**
        Do not add a node in $O_i$'s Provenance Chain
    **else**
        Add a node in the $O_i$'s Provenance Chain
    **end if**
**end for**

### 5.4.4 Results and Discussion

We validated the SNAPS model using Openstack cloud. After detecting the suspicious activity in two Openstack VMs (Tiny OS VM and Ubuntu Server VM), SNAPS captured several snapshots of each of them at frequent intervals of time till the incident impact is reduced. All of them are passed through our spatio-temporal model (iRSM) and then transferred to the incident handler's environment. The protocol followed by the incident handler is shown in Figure 5.12. All the data of the snapshots is stored in the database and all their metadata is extracted and indexed using *Elastic search* (It is a massively distributed system which supports multi-tenancy and can perform advanced indexing). Performance of the protocol is measured and the same is shown in Table 5.1 and Table 5.2.

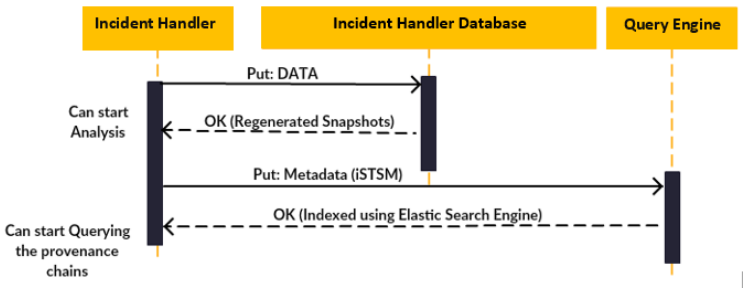



Figure 5.12: Protocol followed by the incident handler after acquiring the snapshots

Table 5.1: Time for building the provenance chains

| S. No. | VM OS | Snapshots | Total Size | Time |
|---|---|---|---|---|
| 1 | TinyOS | $S1$, $S2\_delta$, $S3\_delta$, $S4\_delta$, $S5$, $S6\_delta$, $S7\_delta$, $S8\_delta$ | 50 MB | 5 sec |
| 2 | Ubuntu Server | $S1$, $S2\_delta$, $S3\_delta$, $S4\_delta$, $S5$, $S6\_delta$, $S7\_delta$, $S8\_delta$ | 65 GB | 6640 sec |

From the above, it implies that building provenance chains takes time and it depends on the size of the snapshots. The overhead in time is obvious as the indexer in the configured search engine has to go through each regenerated snapshot. We do not consider this as setback as this is done only once. On the other hand, we observed that the response time for querying the indexed provenance chains is less.

If the incident handler wants to analyze specific sets of objects across multiple snapshots then doing it manually will consume a lot of time. To reduce this, our recommendation engine will give possible findings about those objects automatically. The recommendation engine will work for simple scenarios to complex ones. Some of them are described in the next section.

Table 5.2: Response time for sample queries

| S. No. | Question | Time |
|---|---|---|
| 1 | All objects modified between time 10:20 IST and 17:00 IST in the location *Downloads* | 10 msec |
| 2 | Most commonly modified object across 8 snapshots in the location *MyFiles* | 11.5 msec |
| 3 | Check whether a specified object had modified hard links | 9.3 msec |
| 4 | Common block numbers modified across 8 snapshots | 10.4 msec |

### 5.4.5 Advantages of our provenance system

SNAPS benefits the incident handler in multiple aspects i.e.

- Improved evidence availability: This model can work in a realistic environment as a series of suspicious activities may lead to an incident and before which the evidence collection can be completed. Since, SNAPS is deployed at hypervisor, it would be difficult for the intruder to alter/disable the monitoring capabilities of it.
- Reduces the analysis time: Analyzing a single snapshot involves human effort. If the incident handler wants to analyze multiple snapshots then the scenario becomes more complex. With the aid of the recommendation engine, SNAPS gives the high level clues about the queried object. This saves a lot of time and effort for the incident handler.
- Data independent persistence: When an object is deleted, its provenance data should not be deleted. In our case, each object's provenance chain is stored in the developed recommendation engine and will be persisted even after the same object is deleted from the target cloud instance.
- Scalable: The query results are faster even for huge sized snapshots as each evidence is configured to go through the process of advanced indexing at object level.

## 5.5 Applying the Proposed Provenance System for cloud incident handling

SNAPS can reduce the time spent by the incident handler during snapshot analysis and it is designed to handle many interesting incidents. Some of them are described below:

### 5.5.1 Scenario-1: Giving a recommendation about digital forgery

This recommendation is used to identify digital forgery. It is very important especially during forensic based incident handling to know where a particular object originated from i.e. whether it is created or copied from some other location. We used Algorithm 7 for generating the corresponding recommendations automatically and the same are shown in Figure 5.13. For quick interpretation of the same, our provenance system also provides a visualization capability as shown in Figure 5.14 (shows the origin of the object with inode 384).

Digital forgery is a common operation done by the intruders. There are various types of digital forgery like splicing, retouching, and copy-move attacks [169]. We also identified and created rule sets for detecting each of these. For example, in **copy-move attack**, the object is moved to some other location and it is then modified keeping the base properties intact. Algorithm 8 detects the objects with copy-move attack. The Sim($O_i$, $O_j$) function uses document similarity techniques like cosine, Term Frequency- Inverse Document Frequency (TF-IDF) etc.

**Algorithm 7** Recommendation about whether a queried object is copied or not

```
Input: Query(O_i)            ▷ O_i is forensically significant object for the incident handler
if O_i ∈ S_i then            ▷ Say, O_i is present in location Loc_x and S_i is the first snapshot captured
    for each Snapshot S_{i+1} do
        if O_i ∈ S_{i+1} then
            if S_i_inode(O_i) != S_{i+1}_inode(O_i) then
                if S_i_hash(O_i) == S_{i+1}_hash(O_i) then
                    reco(O_i) = "File may be copied from Loc_x"
                end if
            end if
        end if
    end for
end if
```

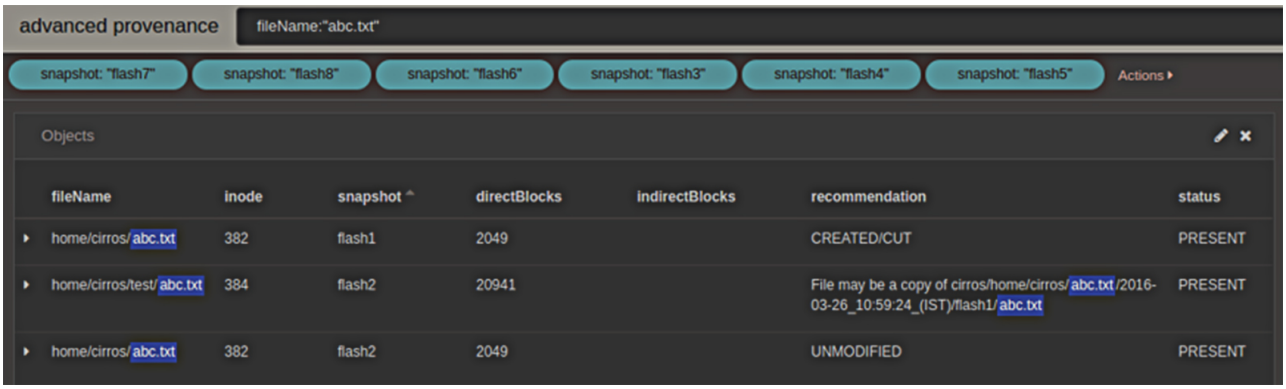



Figure 5.13: Recommendation about whether a queried object is copied or not

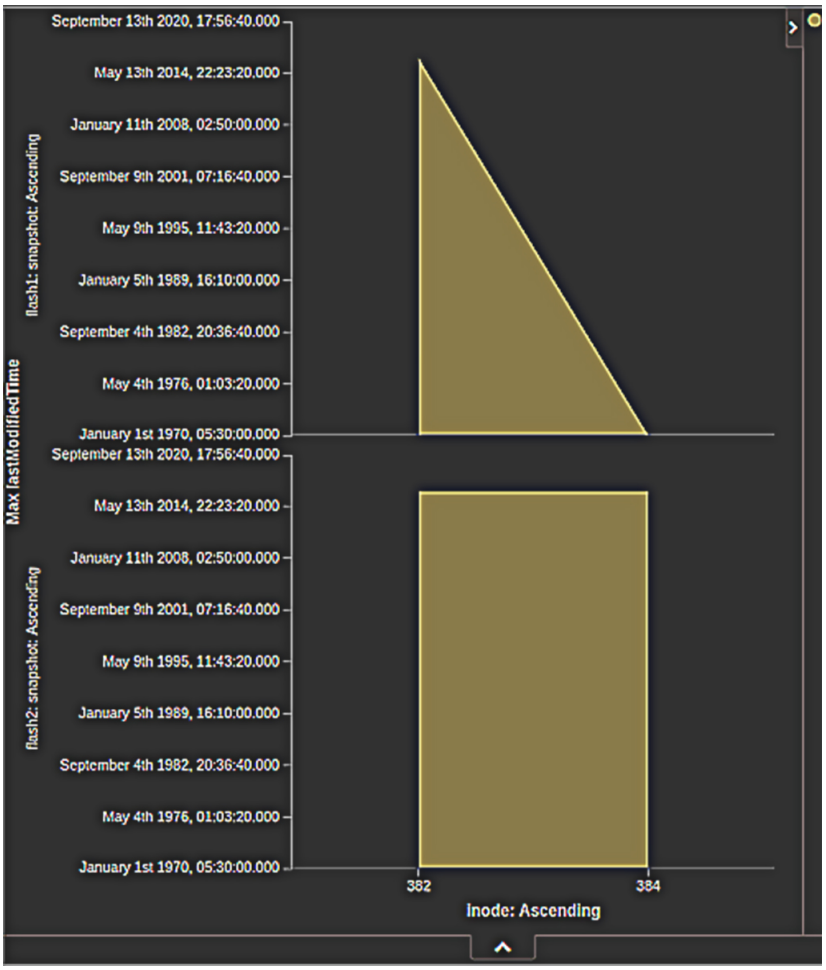



Figure 5.14: A visual representation of the queried object copy history

### 5.5.2 Scenario-2: Identifying the backdoors created

A backdoor can provide unauthorized access to a remote system. Any intruder who created a backdoor wants its life to stay for a long time. There are many ways in which an attacker can do this and the most common way is performing a **waterhole attack** [170]. It is a malicious program which can poison the system by injecting a backdoor at most frequently used locations. The backdoor will be reinitialized each time the victim had interactions with that common location/files.

A set of rules was devised to detect backdoors. Some of the important ones are shown in Algorithm 9. We also framed a waterhole condition i.e. whether queried object $O_i$ is commonly used by the victim or not. Also, we applied all the relevant rules to increase the correctness of the recommendation. For example, in Figure 5.15, we identified that abc.txt was a symbolic link attached to abc.jpeg.

We initially identified from the victim network logs that abc.txt was injected from external VM. After applying Algorithm 11 (subsection 5.5.4), we came to know that abc.txt was obfuscated and its actual extension is *.so* and contains malicious code which was attached as a symbolic link to abc.jpeg. Whenever VM restarts, the backdoor life may get expired. It

**Algorithm 8** Recommendation on whether the queried object is undergone with copy-move attack or not

```
Input: Query(O_i)          ▷ O_i is forensically significant object for the incident handler
if O_i ∈ S_i then                                          ▷ S_i is the first snapshot captured
    for each Snapshot S_{i+1} do
        for each Object O_j in S_{i+1} do
            if Sim(O_i,O_j) >= th_max then
                reco(O_i) = "Object O_j is a victim of Copy-Move attack"      ▷ O_j is not present in S_i
            end if
        end for
    end for
end if
```

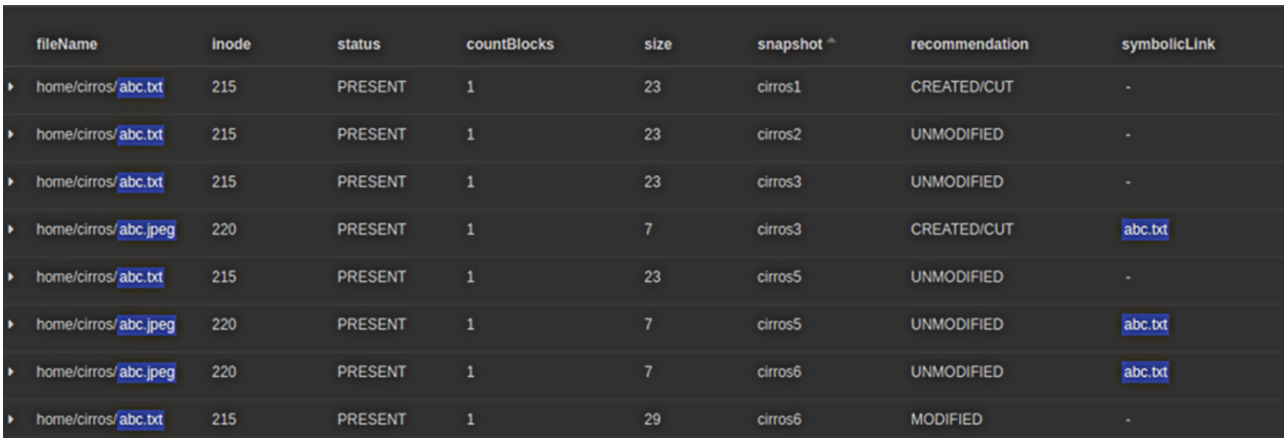
gets reactivated each time when the victim interacts with the abc.jpeg file (frequently used object in victim VM and it is known from its history of modification times). This makes the intruder to always have a remote access of the victim machine. We added tens of rules to our recommendation engine for detecting multiple variants of backdoor.

**Algorithm 9** Recommendation about whether a queried object is used as backdoor

```
Input: Query(O_i)          ▷ O_i is forensically significant object for the incident handler
if O_i ∈ S_i then                                          ▷ S_i is the first snapshot captured
    Store_O_i = sym_link (O_i)
end if
for each Snapshot S_{i+1} do
    if O_i ∈ S_{i+1} then
        if sym_link(O_i != NULL) then
            Store_O_i' = sym_link(O_i)
            if Store_O_i != Store_O_i' then
                reco(O_i) = " May be to O_i, backdoor was linked"
                reco'(O_i) = " The backdoor name is O_j"
            end if
        end if
    end if
end for
```

### 5.5.3 Scenario-3: Identifying suspicious activities

Once a backdoor is created, the intruder may alter/delete the data or metadata of an object in the victim VM which he/she is not supposed to do and can be treated as suspicious. For example, one of the common attacks is, **Host File Redirect** [171] i.e. in Windows, /Etc/Hosts file resolves the name-to-IP lookup as always contacting a DNS server for the same name resolution is a time consuming task. The intruder may add a malicious server IP to the commonly visited domain name of the user for which he initially has to change the permissions of the file.



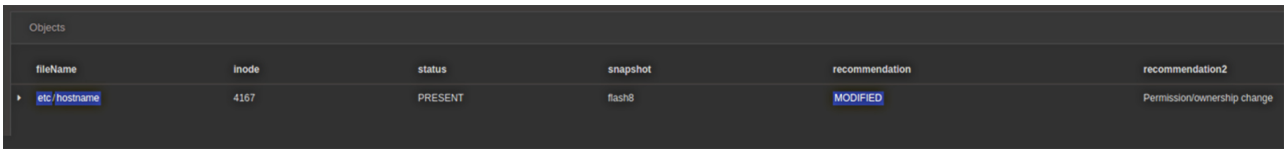
| fileName | inode | status | countBlocks | size | snapshot | recommendation | symbolicLink |
|---|---|---|---|---|---|---|---|
| home/cirros/abc.txt | 215 | PRESENT | 1 | 23 | cirros1 | CREATED/CUT | - |
| home/cirros/abc.txt | 215 | PRESENT | 1 | 23 | cirros2 | UNMODIFIED | - |
| home/cirros/abc.txt | 215 | PRESENT | 1 | 23 | cirros3 | UNMODIFIED | - |
| home/cirros/abc.jpeg | 220 | PRESENT | 1 | 7 | cirros3 | CREATED/CUT | abc.txt |
| home/cirros/abc.txt | 215 | PRESENT | 1 | 23 | cirros5 | UNMODIFIED | - |
| home/cirros/abc.jpeg | 220 | PRESENT | 1 | 7 | cirros5 | UNMODIFIED | abc.txt |
| home/cirros/abc.jpeg | 220 | PRESENT | 1 | 7 | cirros6 | UNMODIFIED | abc.txt |
| home/cirros/abc.txt | 215 | PRESENT | 1 | 29 | cirros6 | MODIFIED | - |

Figure 5.15: Backdoor is detected from the acquired VM snapshots of Openstack cloud

Algorithm 10 takes any suspicious permission change as input and gives all the objects associated with that activity. A similar detection was observed in the acquired snapshots and the same is shown in Figure 5.16. When the input changes, the set of conditions we check will also change.

**Algorithm 10** Detects the suspicious activities on system files

```
1:  Input: Query(reco)                                   ▷ suspicious permission change
2:  if O_i ∈ S_i then                                    ▷ S_i is the first snapshot captured
3:      for each Snapshot S_{i+1} do
4:          if O_i ∈ S_i then
5:              if modified_time(O_i(S_i)) != modified_time(O_i(S_{i+1})) then
6:                  if permissions(O_i(S_i)) != permissions (O_i(S_{i+1})) then
7:                      reco(O_i) = "Unauthorized operation, may be suspicious"
8:                  end if
9:              end if
10:         end if
11:     end for
12: end if
```



Objects

| fileName | inode | status | snapshot | recommendation | recommendation2 |
|---|---|---|---|---|---|
| etc/hostname | 4167 | PRESENT | flash8 | MODIFIED | Permission/ownership change |

Figure 5.16: Change in access control policy was detected on the system files of the target VM

### 5.5.4 Scenario-4: Detecting the obfuscated files

Obfuscation is a process used to confuse, disorient and divert the forensic incident handler during incident handling [172]. One of the common obfuscation techniques is to change

**Algorithm 11** Detecting the obfuscated objects from the acquired snapshots

Input: Query($O_i$) ▷ $O_i$ is forensically significant object
external_ext = $O_i$.ext;
**for each** Snapshot $S_i$ **do** ▷ i is max. no. of snapshots
    **if** $O_i \in S_i$ **then**
        internal_ext = carve($O_i$.ext);
        **if** external_ext != internal_ext **then**
            reco($O_i$) = "File $O_i$ may be obfuscated";
            current = i;
            break;
        **end if**
    **end if**
**end for**
**for each** object $O_x$ in $S_{current-1}$ **do**
    **if** $hash(O_i) == hash(O_x)$ **then**
        **if** $inode(O_i) == inode(O_x)$ **then**
            reco($O_i$) = "$O_i$ may be obfuscated";
            reco($O_i$) = "Obfuscated from $O_x$";
            break;
        **end if**
    **else if** $sim(O_i, O_x) > Thresh_{max}$ **then**
        reco($O_i$) = "$O_i$ may be obfuscated";
        reco($O_i$) = "Obfuscated from $O_x$";
        break;
    **else**
        print("Searching for next object");
    **end if**
    reco($O_i$) = " $O_i$ is newly created and obfuscated in $S_i$ and its source may not exist in previous snapshot"
**end for**

| fileName | inode | status | snapshot | recommendation | recommendation2 |
|---|---|---|---|---|---|
| home/cirros/file.png | 214 | PRESENT | cirros1 | CREATED/CUT | |
| home/cirros/file.png | 214 | PRESENT | cirros2 | UNMODIFIED | |
| home/cirros/file.txt | 214 | PRESENT | cirros3 | File may be an obfuscation of cirros/home/cirros/file.png/2016-03-22_10:57:30_(IST)/file.txt | Change in file extension |
| home/cirros/file.txt | 214 | PRESENT | cirros4 | UNMODIFIED | |
| home/cirros/file.txt | 214 | PRESENT | cirros5 | UNMODIFIED | |
| home/cirros/file.txt | 214 | PRESENT | cirros6 | UNMODIFIED | |

Figure 5.17: Recommendations about the obfuscated objects in multiple snapshots

the extensions of the objects. There are many ransomwares in the market today which can change the file extensions intelligently. They use unicode characters like (U+202e) which can override the file name from right to left and can force the file system to display the text in the reverse order. For example, a ransomware using this unicode character combination will make the file name *BirthdayGift by [U+202e]3pm.scr* to appear as *BirthdayGift by RCS.mp3*. This activity may create curiosity in the target user and makes him open the file which thereby may affect his/her system as it is actually an executable file (i.e. .scr).

Using carving techniques, we have read the first few bytes of the header to know a file internal extension. If both the external and internal extensions of an object are not equal then our recommendation engine says that, the object has been obfuscated. Using the provenance information generated from multiple snapshots, the capabilities of the recommendation engine were extended to identify the source of the incident. In this case, there are two possible approaches: (1) Obfuscation is done on the existing object (2) A duplicate copy of the object may be created and then obfuscation may be performed on it.

For both the cases, we identify the source object of the obfuscation and the same is shown in Algorithm 11. We configured these rules in to our recommendation engine to give possible findings about the target objects in the acquired snapshots (Figure 5.17). The source object of the obfuscation is also detected and given as recommendation.

Even though multiple snapshots are taken in a short interval of time, they may not capture every change in the VM. This drawback is neutralized by the several advantages that SNAPS provides i.e. it is the most reliable provenance system, handles the data gravity and helps the incident handler by giving possible recommendations. There will be cases where the existing provenance systems may not capture anything when they are disabled by the intruder but SNAPS can ensure the capture of multiple snapshots before the actual incident. We tested our provenance system recommendation engine for various objects in the acquired snapshots. The precision is 86% and recall is 81% (Table 5.3).

Table 5.3: Precision and Recall of the proposed provenance system

| **Predicted Negatives** | **Predicted Positives** |
|---|---|
| TN: 250<br>FN: 11 | FP: 8<br>TP: 50 |

## 5.6 Incident Handling using Deleted Cloud VM Snapshot Objects

It is important to note that, recovering deleted objects from the cloud is difficult than the traditional digital environment due to its multi-tenancy and rapid elasticity properties. In [86] and [174], the authors concluded that there are no proper assessment approaches to identify the extent of recovery possible for a deleted object in the cloud environment. To answer this, we conducted several experiments on a cloud VM's snapshot.

### 5.6.1 Can we recover a deleted object from cloud VM Snapshot ?

In forensic based incident handling, analysis on the deleted objects reveals interesting clues and facts about the occurred incident. To recover a deleted file, all its unallocated data units should be extracted. This can be done using either of the two approaches: Metadata-based recovery and Application-based recovery respectively [173]. Our emphasis is on metadata based object recovery as the application based recovery is not supported by all the file systems and operating systems.

The common step that any intruder might take after performing an incident is to delete the traces (suspicious files) from the system. Once a file is deleted, the extent of recovery depends on the mapping type. All the possible mapping types of **metadata category** are described below:

- One-to-one Mapping: In this case, recovery of a deleted file is possible as each data unit of this category is only associated with one metadata entry (Figure 5.18(a)).
- Many-to-one Mapping: More than one meta data entry may point to a single data unit. In this case, recovery is possible but it is difficult to identify the data unit's owner (Figure 5.18(b)).
- No Mapping: The data units in this category will not have any associated metadata entry. In such a scenario, the incident handler may not be able to recover the object (Figure 5.18(c)).

A deleted file's data units can internally be in any of the above mappings and it depends on several criteria like the file system, operating system, usage pattern and even the underlying environment.

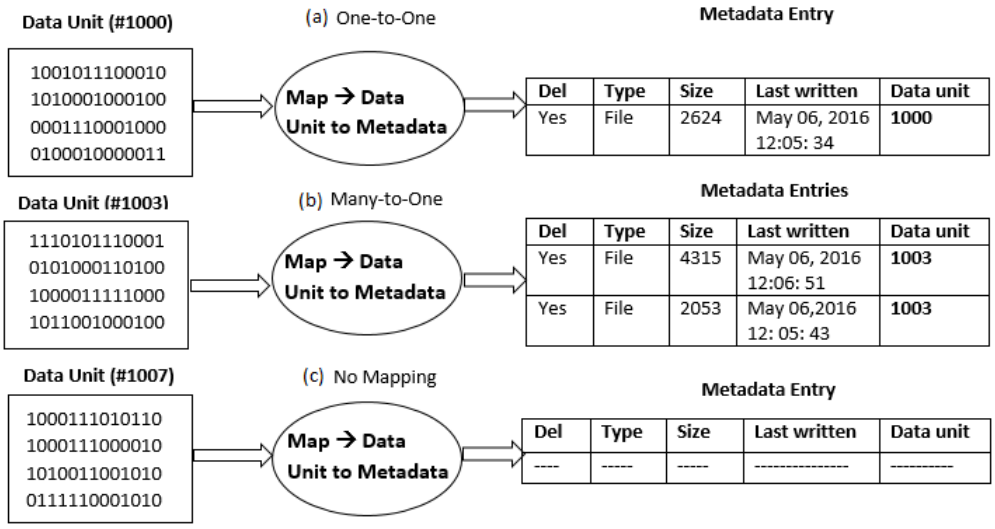



Figure 5.18: Status of a deleted object

### 5.6.2 Experimental observations on openstack VM snapshots

To test the feasibility of extracting the deleted files from a cloud VM snapshot, we acquired the ext3 file system snapshot from the Openstack cloud VM. We use some of the features provided by sleuthkit (an opensource forensic tool) to provide the basic analysis capabilities for the cloud evidences [99]. We followed the below steps to know the possibility of recovering any deleted object.

- Step-1: We extracted all the unallocated blocks of the snapshot using *blkls* command of the sleuthkit (Figure 5.19(a))
- Step-2: The output of the *blkls* is given as input to *blkcalc* to get the actual fragment number (Figure 5.19(b)).
- Step-3: The *ifind* command is then used to know the mapping followed by each block. It returns an inode associated with the given block number. In Figure 5.19(c), for the unallocated block number-1342, an inode does not exist which implies that the deleted file in the Openstack cloud VM snapshot followed *No Mapping* strategy.

To check whether it uses *No Mapping strategy in all the cases*, we followed the below steps:

- Step-1: We retrieved the list of all the deleted files along with their inode numbers using the fls command of sleuthkit (Figure 5.20).
- Step-2: We then tried to list the blocks associated with each deleted file. For instance, the inode 408 prints zero blocks (Figure 5.21). This indicates that the inode does not have any mapped blocks and this validates our earlier conclusion that it follows - *No Mapping* strategy.

It is important to note that, the list of deleted files in Figure 5.20 has some reallocated files. When we apply the same *istat* command with the reallocated inode-379 then it lists the newly associated block numbers (Figure 5.22). This indicates that the deleted inode-379 has been reallocated to some other new file and the old block numbers of 379 are unknown. This makes the recovery of those objects highly difficult.

**Deduction:** Since cloud VM objects follow No-Mapping strategy, it would be very difficult to retrieve deleted objects So, we propose an approach based on the NLP techniques to achieve this and the same is discussed in the next subsection.

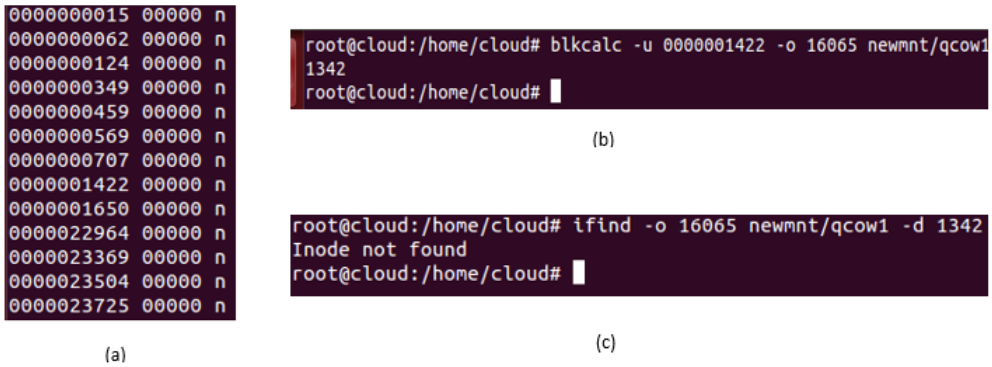



Figure 5.19: Identifying the mapping strategy followed by Openstack cloud VM snapshot

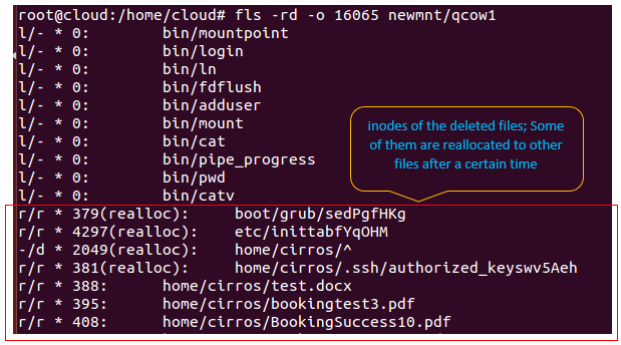



Figure 5.20: Showing the deleted and reallocated objects in cloud VM snapshot

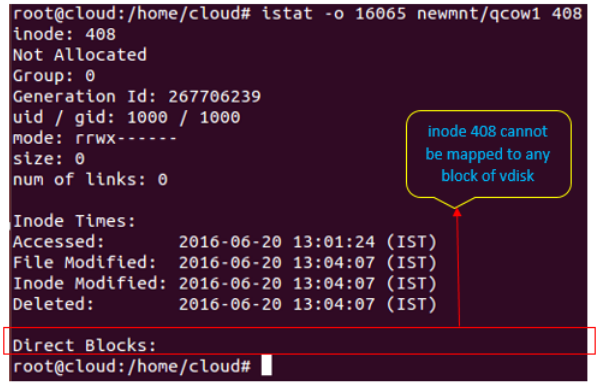



Figure 5.21: Direct blocks associated with the inode number

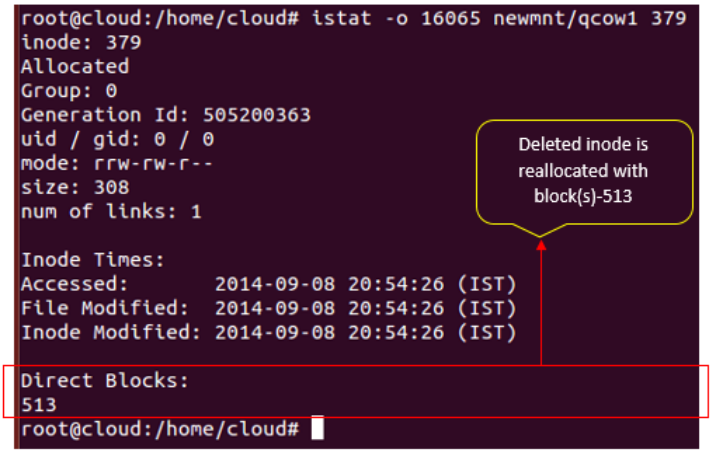



Figure 5.22: Reallocated object with newly allocated direct blocks

### 5.6.3 Proposed approach to recover deleted objects using NLP techniques

Analyzing every data unit will consume a lot of time. For example, if a 512 byte unallocated data unit requires 5 seconds for analysis, then to analyze 400 GB of deleted data, one requires 1940 days (nearly 5 years). Some researchers have considered every free data unit and identified its file type using Linear Discriminant Analysis [175], K-Nearest neighbor [176], Shannon Entropy [177] and Support Vector Machines [178]. These techniques are good at classification of free data unit's file type. These may not fit our requirement as knowing the file type would not help us in retrieving the deleted files.

Our approach would generate multiple groups from all the unallocated data units based on the content similarity. Each group pertains to a deleted object in the snapshot. We consider a realistic assumption that, all the data units of a single file may have content with some similarity. However, there could be some false positives which we reduce using various identified filters.

#### Approach proposed for recovering deleted objects from snapshots

Since mapping does not exist for deleted files' data units of a cloud VM snapshot, we extract all the unallocated data units and give them as input to the NLP engine where it can form various logical groups. Our NLP engine achieves this with the aid of its two modules i.e.

- *Similarity Measurement Module:* We use cosine similarity to calculate the similarity between the unallocated data units [179]. It is a part of the vector-space model. All the extracted unallocated data units are projected into an N-dimensional space where N is the number of words in the term-document matrix. The angle between the unallocated data units is calculated using the cosine rule (equation (5.6.1)).

$$cos^{-1}\frac{v1.v2}{\sqrt{v1^2 + v2^2}} \tag{5.6.1}$$

The lesser the angle between the vectors, the more closely the data units are related to each other.

- *Document Significance Module:* Term Frequency and Document Frequency (TF-IDF) is an amalgamation of two popular measures- (1) Term Frequency is the number of times a given word appears in a document (here, unallocated data unit). (2) Document Frequency is the number of documents in which the term appears. The higher the term frequency, the higher is the importance of the term. The lower the document frequency, the higher is the importance of the term (Equation 5.6.2).

$$TF - IDF = Termfreq * log(N/n_d) \tag{5.6.2}$$

where $n_d$ is number of documents or unallocated data units in which the term appears and N is the total number of documents.

By considering the above two measures, we applied Latent Semantic Analysis on the unallocated data units. We obtain logical groups where each group consists of a set of blocks with similar content. The blocks in a logical group may pertain either to a single deleted file or to multiple deleted files. This triggered us to define two cases i.e.

- Best Case: If there is no similar content across deleted files then the blocks in each logical group represent a deleted file. This helps the incident handler in knowing the exact content of every deleted file.
- Worst case: In reality, a set of deleted files may have similar content. This might lead to a situation where a few logical groups may also contain blocks of other files (false positives). The more similar the deleted files, the more the number of false positives in each logical group. This might lead to a reduction in the classification accuracy.

There are ways in which the classification accuracy can be improved even in the worst case scenario. We achieve this by designing and applying various filters. Each of the filters is described below.

- Pre-processing phase: Extract all the unallocated blocks and identify all the header blocks. Randomly select any header block ($Hb_{prim}$) and pass it to the NLP engine and that gives a set of blocks which are similar to $Hb_{prim}$.

It is important to note that, we do not pass each logical group through all the following filters but only the groups that had the same block number in more than one group.

- Header based Filter: Check whether there is any header block of other files, say, $Hb_{sec}$ in each logical group. If yes, then pass $Hb_{sec}$ header to NLP engine to get its own set of logical group containing similar blocks. Then, check for the below conditions:
  - Condition 1: If blocks in $Hb_{prim}$ logical group were present in the logical group formed from $Hb_{sec}$ then check for condition 2.
  - Condition 2: Extract a common block in both the logical groups ($Hb_{prim}$, $Hb_{sec}$). Check the similarity value of that block in both the groups. If the block has more similarity coefficient in $Hb_{sec}$ then remove the same block from the $Hb_{prim}$ else the block has to be removed from the $Hb_{sec}$. This condition has to be checked against all the common blocks of both the logical groups.

  A logical group may have more than two header blocks and even in that case the above conditions can be applied.

- Carving based Filters: We use various carving techniques to reduce the number of data units in each logical group. This reduction will increase the possibility of classifying the blocks that pertain to the same object into a single logical group.
  - Signature based: There can be multiple similar files with each file in a different format. All these file blocks should not be categorized under a single logical group. For example, a .docx and .pdf may contain very similar data but none of the blocks of .pdf should fall into .docx and vice versa. To achieve this, we carve the header block of a file to identify the signature of the file by which the file type can be known. Using this technique, similar blocks of other file types that were classified under the same logical group can be filtered out.
  - File Structure based: Some file types like .ppt, and .xlsx. have file system based data in their header block which include various details like modified time/deleted time, file length, and file name. Using this information, the accuracy of each logical group can be increased. For example, if an incident occurs during time period T then the other file blocks that were deleted much before the incident can safely be ignored. Even file names of each logical group can be known using the header of the logical group. The only limitation with this type of carving is that it cannot be applied on all file formats as some of them might not contain the required information in the header field.
  - Embedded size based: The incident handler can specify the file sizes that he/she is interested in. For example, after applying the above filters, if the size of a logical group is x and the incident handler only wants to investigate the logical groups whose size is above x, then only those groups can be considered. Furthermore, removing the blocks using the above filters can be done until the total size of logical group is equal to the size mentioned in the header block of that group.

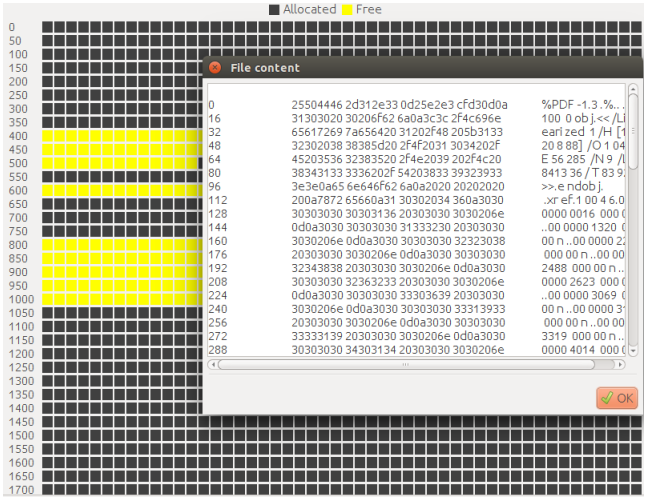



Figure 5.23: Memory window of cloud VM snapshot showing allocated and unallcoated data units

*Advantages of our approach:* (1) Reduces human effort (2) Reduces the time spent in analyzing unallocated data units (3) Helps the incident handler find out objects relevant to incident handling in a faster way (4) Our approach of recovering the deleted objects works for any file with readable content.

### 5.6.4 Validation of the proposed approach using Openstack VM snapshots

We initially acquired the VM snapshot of Openstack cloud VM. From the acquired snapshot, unallocated blocks are identified (Figure 5.23). The yellow blocks are unallocated (i.e. deleted) and the black blocks are allocated data units (non-deleted blocks). When incident handler clicks on any of the unallocated block, the content can be seen and exported to the desired location.

We select a header block data unit and gives that as an input to the NLP engine and it outputs a set of unallocated blocks. All of those blocks greater than or equal to the defined threshold are grouped into a single logical group. For example in Figure 5.24(a), the first unallocated block number given is 234 and the blocks 598, 599, 600, 671, 672 and 700 which had similar content as compared to block-234 are placed in the same group. Here, the threshold is fixed as 0.5 and can be changed depending on the block size, file size and the incident handler's requirement. For this, we used the data set $state\_union$ *from* $nltk\_data$. The approach is applied for various file types- pdf (Data set: movie reviews present in nltk data) and doc (Data set: movie reviews present in nltk data). The same are shown in Figure 5.24(b) and 5.25 respectively.

If each logical group contains the blocks of only a single object then the analysis becomes very easy. In reality, the classification accuracy may not always be cent percent and depends on the content of the files. Based on this, we identified three possible cases as

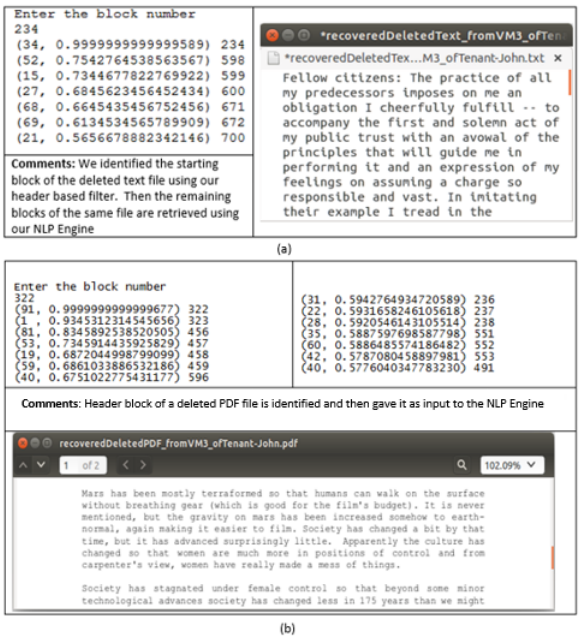



Figure 5.24: Unallocated data unit based logical grouping for text and PDF files

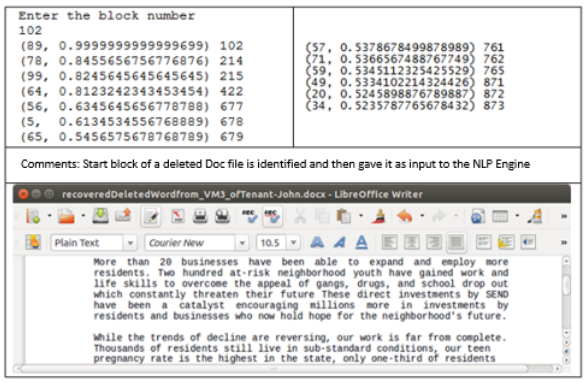



Figure 5.25: Unallocated data unit based logical grouping for doc files

shown in Table 5.4.

- Case 1: *Deleted objects are not similar to each other.* If content similarity is less among deleted files then the classification accuracy will be high. We observed the same (i.e. 96 % classification accuracy) when about 45 files were deleted (15 PDF, 15 text files and 15 Doc files) and they were reconstructed by forming 45 logical groups where each logical group represented a deleted object. We noticed very few false positives in all the logical groups. We then applied various filters to further enhance the classification accuracy and achieved an improvement of 2 %.

- Case 2: In this case, *some deleted objects have similar content and the remaining deleted files have dissimilar content.* The classification accuracy decreased as compared to the above case. After applying various filters, we observed a reasonable improvement in the classification accuracy (15 % better grouping).

Table 5.4: Before and after applying filters

| Role of designed filters in improving the classification accuracy | | | | |
|---|---|---|---|---|
| Possible Cases | Deleted Files | Unallocated Blocks | Classification accuracy (before) | Classification accuracy (after) |
| Case 1: Mostly distinct Files | 45 (15- txt, 15 - .pdf, 15- .doc files) | 180 Data units | 96 % | 98 % |
| Case 2: Both distinct and similar | 30 (10- .txt, 10- .pdf, 10- .doc files) | 115 Data units | 62 % | 77 % |
| Case 3: Mostly similar files | 30 (10- .txt, 10- .pdf, 10- .doc files) | 122 Data units | 41 % | 81 % |

- Case 3: *Most of the deleted files have similar content*. This is the most challenging case as each logical group may have blocks pertaining to other deleted files as well. We observed that this case had the least classification accuracy (41 %) but after applying various filters, it improved to more than 80 %.

From the above, it is evident that irrespective of the text content similarity across deleted files, the classification accuracy after applying the required filters has increased.

## 5.7 Summary

VM Snapshots are the most crucial evidences for forensic based cloud incident handling. The challenge is, always the snapshots may not be available. We addressed this problem by proposing a model called iCFR and validated it with a scenario. We also addressed the major issues of data gravity. The acquired snapshots from the cloud are transferred to the incident handler's environment where we build a provenance system using those snapshots. The design of our provenance system is inspired from spatio-temporal models and we then identified several applications of the proposed system which can reduce the time spent for incident handling. Finally, we also proposed a NLP based approach for recovering deleted objects indirectly from the target VM snapshot(s).

The publications from this work are [PUB1] and [PUB2] (refer page no. 136-137). In the next chapter, we discuss the proposed event correlation techniques considering multiple artifacts and these would help the incident handler to have quick interpretation about the occurred incident.

# Chapter 6

# Handling Cloud VM incidents using Event Correlation

*"Cloud is the biggest digital revolution for every industry"*

-Google

## 6.1 Background and Motivation

In this chapter, we deal with one of the technical challenges namely *Event Correlation* for cloud incident handling. The term *Event correlation* indicates the relations between two or more events [181]. Using this technique, the incident handler extracts higher level of knowledge from huge number of events, identifies the faults and filter the redundant events, finds irrelevant and spurious messages and makes the predictions about the future trends.

Event correlation benefits the incident handler in many ways like, to have better interpretation of the underlying incident, to identify the root cause of the incident and to know the incident scope and its intricacies [127]. Finally, all these will make the overall incident handling effective. This helps in arriving at more reliable and fool proof results, which can be submitted to the court of law.

Also, event correlation can enhance the natural process of incident handling to get the comprehensive picture about the occurred incident. Absence of smoking gun evidences can trigger one to use the event correlation techniques [183]. Event correlation has applications in various domains like system management, security management, network management and service management (Figure 6.1). In this chapter, we opt security management and in it the approach of *event correlation* was chosen. Based on the literature study, we identified that event correlation in cloud can be performed at two levels which we describe in the following subsections.

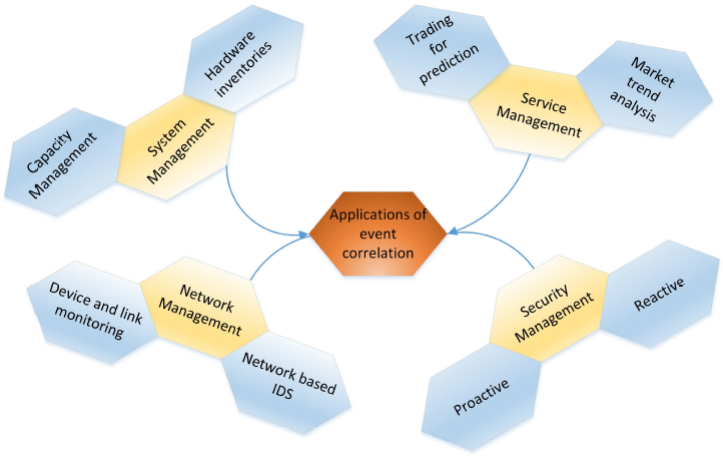



Figure 6.1: High level taxonomy for event correlation applications

### 6.1.1 Event correlation across evidences from single CSP

Regarding the event correlation from a single Cloud Service Provider (CSP), the initial major contribution is by [184] and was published as a Google US patent. It had designed an event repository where all the timestamped events will be stored. With the aid of a query engine, the events of interest were retrieved and analyzed. It just gave an overview of the phases to be followed for performing event correlation and did not discuss its relevance to incident handling.

In [95], an intrusion detection approach was designed using event correlation. Attack detection by correlating the casual events was described with the consideration of time and other constraints. The drawback is, it will not work for unknown attacks (i.e. a knowledge base consisting of the attack constraints is a prerequisite to make this correlation approach work which is not possible in the case of zero day attacks). Also, the suggested approaches can be used only in proactive scenarios and not after the incident.

The other works which have applied the event correlation technique in different contexts include:

- The power consumption of each Virtual Machine (VM) is calculated by using the PCA and they found that the pairs {CPU, Cache} and {DRAM and disk} had high correlation [185].
- Used the concept of correlation to check the dependence of multiple failure events from which the availability of proactive check pointing can be made [186].
- Proposed a framework to collect the data from multiple levels of cloud and applied the correlation technique to identify the intrusions that violate the service level agreements [38].

The major finding from our literature review is that *cloud event correlation has not been extensively applied for handling cloud incidents by considering forensic practices.*

Our objective, thus, is to handle them using the proposed forensic-based event correlation techniques. We discuss event correlation approaches for handling incidents in traditional environments below, which forms the motivation for our proposed approaches.

In [188], the idea of performing event correlation across disks was introduced. The authors discussed a solution to extract various features like social security number, and email id to perform both single and cross drive analysis. But, it has several disadvantages:

- The discussed approach is very basic and could not detect any incident automatically
- The set of features used for correlating various disks is very limited
- The correlation mechanism applies to only hard disk and not to any other digital artifacts (main memory, logs)

In [189], a framework was proposed to correlate the events among multiple forensic objects. It also discusses various parsers that can parse pcap files, configuration files and memory data. But, this approach has certain drawbacks:

- Only the artifacts from a single system were considered; also issues involved in cross artifact analysis from multiple computers were not discussed.
- The time spent on analysis can still be optimized
- No filtering module was present to reduce the extent of events that are to be analyzed

In [190], the authors proposed a correlation framework for botnet detection. They used a network analyzer to identify the suspicious clusters and then used a host analyzer to extract each suspicious system data for verification purpose. But, this work has the following drawbacks:

- Network and host activities were considered for effective event correlation but they are limited to live systems
- Their work for event correlation is limited to botnets and did not address other incidents.

In [191], the authors proposed an approach for finding the malware traces across computers. They initially extracted features from the target evidence and then applied clustering to detect botnets (key logger and spy bot). But this approach has the following shortcomings:

- No consideration of optimal feature set for the proposed event correlation approach
- Except disks of multiple systems, other artifacts were not considered

### 6.1.2 Event correlation across multiple cloud providers

The limited interoperability among the cloud providers increases the complexity of incident handling across CSPs. In reality, there are lot of dependencies among the cloud providers [33]. If the dependency chain gets broken then the difficulties involved in event correlation will become worse. Moreover, there is no standard policy or tool that facilitates the event correlation across the cloud providers [35]. Our focus is only on performing event correlation among the artifacts acquired from *single CSP*.

### 6.1.3 Novelty of our event correlation approach

In comparison with the above, our approach of event correlation differs in the following aspects:

- The first and obvious difference is, the proposed event correlation approach is specific to the cloud environment.
- To the best of our knowledge, our proposed approach is the first one to discuss in detail about the role of *cloud service logs* for handling cloud incidents.
- We correlate the events effectively from all the major evidences available in the target tenant VMs.

We identified all the possibilities of performing event correlation for handling cloud incidents and then we organized the entire chapter as follows.

- In Section 6.2, we propose a segregation model for cloud event correlation.
- Event Correlation can be homogeneous (same artifact is used) or heterogeneous (different artifacts are used) and each of these can occur with the *incident known* and *incident unknown* cases. Based on this, we derived four cases. In this work we address the following three cases: Homogeneous-incident unknown based correlation which we discuss in Section 6.3, Heterogeneous-incident known based correlation in Section 6.4 and Heterogeneous-incident unknown based correlation in Section 6.5. The fourth case namely *Homogeneous-incident known based correlation* is not within the scope of this work as it varies from one incident to the other.
- Finally, we conclude the chapter in Section 6.6.

## 6.2 Proposed Segregation Model for Cloud Event Correlation

Segregation of duties is a challenge while conducting forensic based incident handling in the cloud environment [182]. To address this, we proposed a segregation model for cloud event

correlation i.e. suggested phases of event correlation were clearly spread across various cloud actors (Figure 6.2). It is important to note that, this model has its base from our proposed approach named SEASER (Chapter 4).

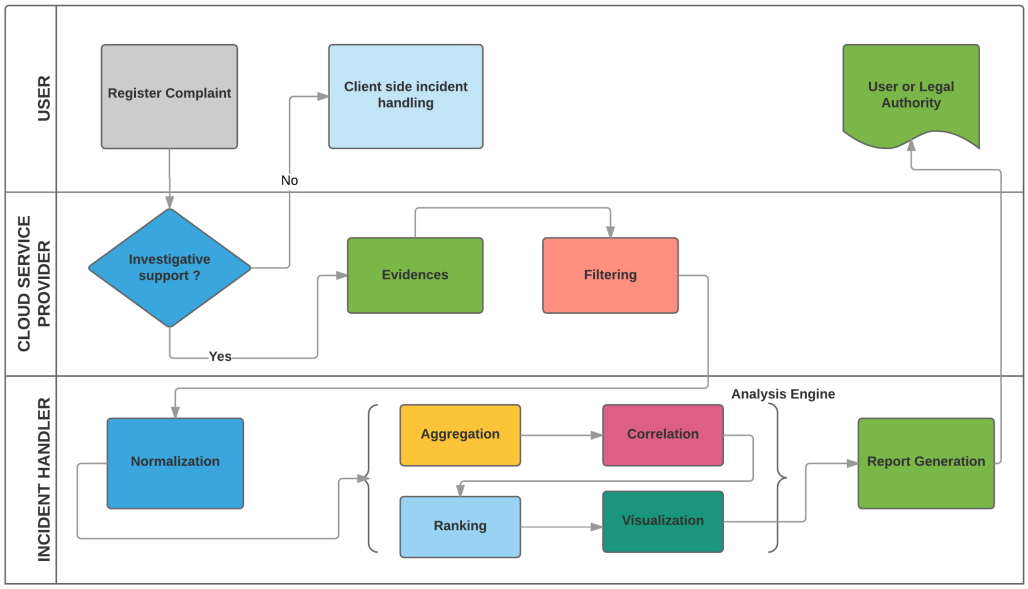



Figure 6.2: Proposed model for event correlation in cloud environment

Initially, the victim tenant will generate a complaint to the corresponding CSP. With our strategy of mapping incident handling phases with forensic investigation stages, the evidence availability is always ensured. All those evidences will be given as input to the filter module (or segregation phase). This module preserves the privacy of other tenants i.e. it takes the job to reveal only the data pertaining to the target tenant and all these events are given to the normalization module present at the incident handler side. It synchronizes the structure of various events to a common format so that it will help the incident handler in effective correlation. The correlation module considers both the incident symptoms and the event data to identify associations among the normalized events.

After correlation, groups are formed from the events based on their similarity. Prioritizing these groups by assigning appropriate ranks will speedup the incident handling process. Moreover, visualizing the formed groups will enhance the interpretation of correlations among the events. Finally, the results of the analysis module will be given to the user/legal authority in the form of a report.

Taking this model as a base, the following sections discuss the proposed approaches for performing various types of event correlation.

## 6.3 Performing Homogeneous Correlation in the Incident Unknown Case

There are four major cloud VM artifacts - vRAM, Service logs, vDisk and Snapshots. In homogeneous event correlation, any one of these artifacts can be selected and it depends on the context and as well as the incident handler's requirement(s). We will discuss our approach of performing homogeneous event correlation with a scenario- *finding the host IP from which the cloud VM was accessed*. We considered this specific scenario as it helps the incident handler to apply it irrespective of the incident (or when the incident was not known). To accomplish this objective, we considered service logs artifact and then applied event correlation.

### 6.3.1 Importance of this scenario from forensic based incident handling

Even in traditional incident handling, finding the IP of the intruder system will benefit the incident handler to identify and directly reach to that location. Similar advantages will be reflected especially in the case of organization's private cloud where the dedicated systems can access the cloud. We assume that one of the VMs has been accessed by an intruder from his/her dedicated system. Once we identify the host IP of the intruder then the incident handler can easily identify the host system and then this allows him to get more information about the incident.

We have used Service logs in the Openstack cloud test bed (Appendix I). These logs are present in the compute and controller nodes of the cloud.

### 6.3.2 Detailed scenario description and methodology employed

Let us say, an incident $a_i$ has been performed on $VM_a$(victim) by $VM_i$ who is an intruder. The tenant of the $VM_a$ may register a complaint to the CSP. The incident handling process can be started by prioritizing the artifacts in the order of volatility i.e. doing vRAM imaging of $VM_a$ and checking for open network connections will give the set of IPs -$\{IP_i\}$ of other VMs which are connected to the $VM_a$. Each address in the set $IP_i$ is the virtual machine's IP address and not the IP of the host system from which the VM has been accessed. Knowing the host IP of the intruder will surely speedup the process of incident handling for which we used service logs. The challenge involved over here is, they hold all the tenant's information and there will not be any explicit segregation. Our aim is to *find the host IP of the intruder system without violating the privacy of other tenants*.

*Filter Module:* A tenant may have many virtual machines and each VM will have an *instance id*. Based on this, events pertaining to the target VM can be identified. The problem is, all events in each service log may not have the attribute-instance id which makes


Figure 6.3: Using MongoDB service log to extract metadata associated with each instance

the investigating entity to identify only few events of the target VM. To address this, we identified other parameters (i.e. request-id, tenant-id, project id, (pid,tid) pair etc.,) which will enhance the process of segregation.

We also verified that, knowing at least one of these parameters, the other VM attributes can be extracted from the MongoDB service log (Figure 6.3). For example, when request-ids are known, the target VM events can be mapped and identified using MongoDB. Similarly, if the virtual machine IP is known then the other required metadata can be found in the same log itself.

The filtered events will be given to the incident handler for further analysis. All the service logs may not be useful during incident handling. For example, *rabbit@controller* log is a messaging broker between two nova components which may not be considered as evidence. Another log, *nova-compute* falls under the same category i.e. it is only used to store and track the resource consumption for the entire cloud and not specific to the VM. This log will be more beneficial to the CSP (resource consumption rate may be known and accordingly resources scaling can be planned) than to the incident handler.

*Normalization:* We observed that, there are variations in the format of service logs. Converting them to a unique format will further make the correlation process easier. For example, in Figure 6.4, we can see the service logs which are in different formats i.e. *nova-api-metadata* of compute node, *nova-scheduler* of controller, *error.log* of controller node etc. All these logs were considered and normalized with the schema as shown in Figure 6.4. Accordingly, all the entries in each log were parsed and stored in the database.

*Correlation:* To achieve our objective, we have taken the *access.log* from the controller

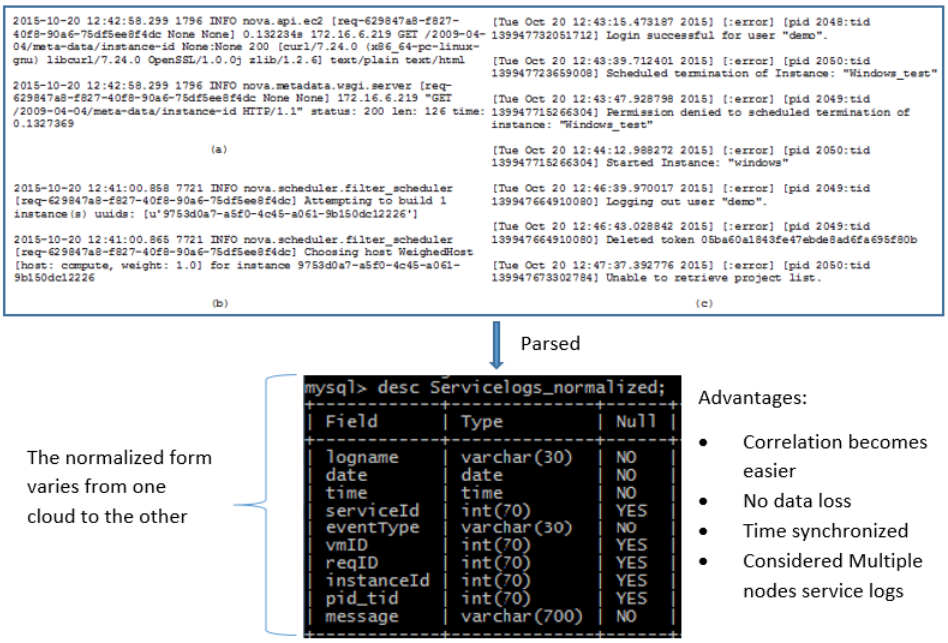



Figure 6.4: Applying normalization phase to the cloud logs in different formats

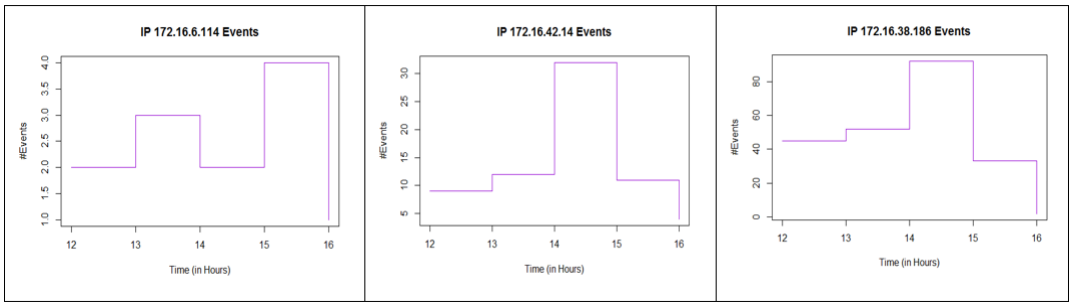



Figure 6.5: Prioritized host IPs used for accessing the target cloud instance

node as base. This log stores the host IP addresses from which the cloud is accessed. The problem is that the information as to which VMs are accessed from which host is not explicitly mapped. With our correlation approach, we achieve this mapping. The basic idea is, say there are *n* IPs from which the VMs have been accessed. We considered the host IP address and identified all its access times to the cloud. The target VM events and their timings were identified and compared with the access times of the current host IP using exact brute force pattern matching algorithm. The same process is repeated for all host IP addresses present in *access.log*. Here, we show only first three correlation results (Figure 6.5) which are having more close association with the target VM.

*Ranking and visualization:* Based on the visual representation, one can easily identify the host ip which has accessed the target VM. For example, in Figure 6.5, $IP_z$(172.16.6.186) has the highest rank (close timely relation) using which we can confirm that the instance named "windows_test" was accessed from $IP_z$. If a tenant accesses multiple VMs from the same host IP then all the instance details under the project can be extracted from MongoDB. We need to follow a similar procedure to plot and get to know the host IP.

*Report generation:* The outcome of this homogeneous correlation emphasizes on listing and validating the host IP from which the target VM was accessed. This is then submitted to the user or legal authority for further steps.

## 6.4 Performing Heterogeneous Correlation in the Incident Known Case

If a VM has been compromised, then there will be many artifacts which should be considered for cloud incident handling. We consider two major artifacts for heterogeneous correlation i.e. vRAM and vDisk.

### 6.4.1 Proposed approach for heterogeneous event correlation in cloud

When investigating the cloud incidents, there are two possibilities. An incident may not be known or may be known. In this section, we discuss the latter case. For any known incident $I_x$, we can know the symptoms and these can be represented and correlated by using various techniques like Bipartite graphs, Code based techniques, Bayesian networks, Causality graphs, Context free grammars etc., [183]. In this chapter, we used the codebook technique.

Codebook is an event correlation technique which uses matrix based representation. Each cell contains either 1 or 0 with row indicating the symptom and each attribute indicating the fault/incident. If there is a 1 in the cell then it implies that the symptom is one of the reasons to cause the corresponding fault. The reasons behind using code based approach are:

- Similarities exist among many event correlation approaches. For example, causality graph, bipartite graphs etc., can be converted and represented using codebook technique.
- Codebook based correlation approach can be simplified by preprocessing the events which needs to be analyzed.
- An effective construction of the codebook will reduce the correlation approach to minimal distance calculation.

The main disadvantage with the codebook is, it may not always be a best fault descriptor. Taking this aspect into consideration, we avoided symptoms which are common across many faults.

### 6.4.2 Results and Analysis

To illustrate the application of the codebook for performing event correlation, we considered an incident named *rootkit*. The behavior of the rootkits changes depending on its type. For example, in this experiment we considered four major and popular variations of rootkits i.e. I1: Kbeast, I2: Phalanx2, I3: Averagecoder, I4: Sutursu. Table 6.2 shows the possible symptoms for detecting each of these rootkits. Since the incident was known, ideal codebook is constructed from the possible symptoms identified (Table 6.3). To confirm the incident (I1 or I2 or I3 or I4) which affected the target victim VM, we followed a two step process: (1) The data pertaining to each symptom is collected from the vRAM and the vDisk of the target virtual machine and transferred to the incident handler's environment (Table 6.2) (2) We group all of them based on the existing similarity measures.

Table 6.1: Time taken for transferring evidences from the cloud to incident handler's (IH) environment

| **Evidence Size** | **Time taken** |
|---|---|
| 10 MB | 2 sec |
| 200 MB | 39 sec |
| 1 GB (Heavy traffic) | 90 sec |
| 10 GB (Heavy traffic) | 750 sec |
| 100 GB (Heavy traffic) | 6200 sec |

For example, if we want to confirm whether the rootkit has the symptom S9 or not. We extracted features to search for the symptom S9 on both vRAM and vDisk. We applied euclidean distance based grouping on that data and the result is shown in Figure 6.6. Comparing the resultant points present in Figure 6.6 (a) and 6.6 (b), the following observations are made:

- There should not be any entries in the second group for baseline case as we are trying to extract the same features for each symptom (here S9) from both the vRAM and vDisk. It is important to note that, for the baseline system there are some points in the second group. These can be ignored as they are not anomaly entries and related to LiME (this is used to image the target VM's vRAM).

- After the rootkit was affected, we have seen more entries in the second group of Figure 6.6 (b) when compared to the baseline system second group. This happened due to the rootkit dynamic behavior which can create feature differences even though we try to extract the similar symptom data from both the artifacts. This confirms that, symptom S9 was present in the target VM.

Table 6.2: List of possible symptoms for rootkit identification

| Symptom | Description |
|---|---|
| S1 | system calls |
| S2 | Sequence operations structures |
| S3 | Loadable kernel modules |
| S4 | The arguments of the processes |
| S5 | Bash shell details during the terminal session |
| S6 | Processes and its hierarchy |
| S7 | Process mappings |
| S8 | Cred structure of the terminal session |
| S9 | Kernel debug messages buffers |
| S10 | Function pointers |
| S11 | Network connections |
| S12 | Process list metadata |
| S13 | File descriptors |
| S14 | Code hoking |
| S15 | Kernel opened files |

The same process is applied for all the symptoms mentioned in Table 6.2 and the final codebook after the incident is shown in Table 6.4. Now this newly created codebook has to be compared with the ideal codebook. For comparison, we calculated the hamming distance between the two. We observed that I2 has minimal hamming distance when compared with others (I1, I3, I4) and this confirms that the target VM is affected with I2 i.e. phalanx2 rootkit.

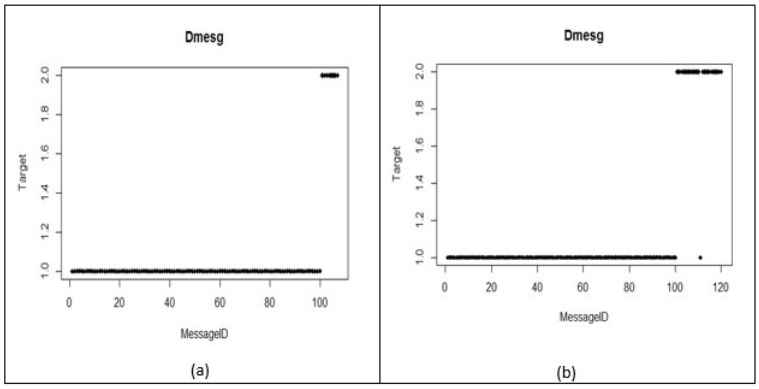



Figure 6.6: Features representing the kernel buffer messages for baseline and affected system (here, Dmesg indicates events in kernel message buffers)

Table 6.3: An ideal codebook for rootkit

| S/I | I1 | I2 | I3 | I4 |
|---|---|---|---|---|
| s1 | 1 | 1 | 0 | 1 |
| s2 | 1 | 1 | 0 | 0 |
| s3 | 1 | 1 | 1 | 0 |
| s4 | 1 | 1 | 0 | 0 |
| s5 | 0 | 0 | 1 | 1 |
| s6 | 1 | 1 | 0 | 1 |
| s7 | 1 | 1 | 0 | 0 |
| s8 | 0 | 0 | 1 | 0 |
| s9 | 0 | 1 | 0 | 0 |
| s10 | 1 | 0 | 1 | 0 |
| s11 | 0 | 0 | 0 | 1 |
| s12 | 1 | 1 | 0 | 0 |
| s13 | 0 | 1 | 0 | 0 |
| s14 | 1 | 0 | 0 | 1 |
| s15 | 0 | 0 | 0 | 1 |

Table 6.4: Codebook after incident(rootkit)

| S | Ix |
|---|---|
| s1 | 1 |
| s2 | 1 |
| s3 | 1 |
| s4 | 1 |
| s5 | 0 |
| s6 | 1 |
| s7 | 1 |
| s8 | 0 |
| s9 | 1 |
| s10 | 0 |
| s11 | 0 |
| s12 | 1 |
| s13 | 1 |
| s14 | 0 |
| s15 | 0 |

## 6.5 Heterogeneous Event Correlation when the Incident is not Known

In the incident unknown case, the incident handler may face difficulties during incident handling (worst case). For example, time spent on the case may be very high, directions to proceed further may not be clear. To reduce these sort of difficulties, we suggest to use the approach of *Event Correlation*. We would illustrate the proposed approach with a typical scenario.

### 6.5.1 Correlation of cloud VM artifacts-incident not known

Since the incident is not known, approaches like Causality graphs, Context Free Grammars, and Case Based Reasoning will not fit our requirement including the approach of codebook. We use anomaly based detection to meet our objective. Due to the multi-tenant and virtualized environment of cloud, anomaly based detection may give more false positives [157]. Thus we divide our approach into multiple stages so that the interpretation and reliability of the correlation results may be increased i.e.

- Stage 1: Correlation among the events present in a single artifact.
- Stage 2: Correlation of events present in multiple artifacts of a cloud virtual machine.

We illustrate the above two stages with the following scenario:

- The tenants performed some malicious activity on the private cloud.

- No proactive solution enabled (like capturing the packets between the tenant host system and cloud) by which the knowledge of abnormal activities that the tenants are going to perform is not present with the CSP.

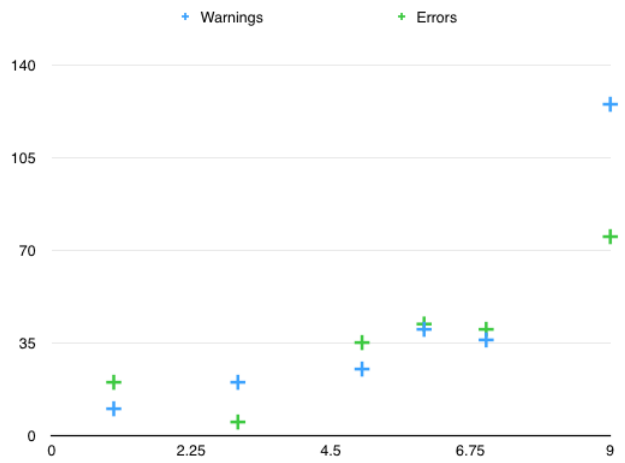



Figure 6.7: Checking for event bursts in target suspicious VM

In this case, we should rely only on the evidences left after the incident and then accordingly perform incident handling. Among all the major evidences (vRAM, Service logs, Snapshots and vDisk), acquisition of *service logs* will be more reliable for the Incident Handler. However, there are certain challenges involved in service logs acquisition like segregation of events pertaining to each tenant which is addressed using the Filter module. This segregation allows the CSP to give only the target tenant events to the incident handler. Considering this, we have taken service logs artifact for handling Stage 1.

For Stage 2, we consider all the artifacts (vRAM, Service logs and vDisk). The correlation results from the first stage can help the incident handler in enhancing the interpretation on the acquired forensic data which helps in second stage correlation.

### 6.5.2 Results and Analysis

**Correlation of events from single artifact (Stage 1)**

We identified that any artifact can be considered for incident handling at the semantic level and non-semantic level. In general, based on the outcome of the non-semantic analysis, the input size of the events to the semantic level can be reduced. The artifact to be analyzed varies depending on the entity involved and even according to the case. As said, we considered service logs because of its high availability and reliability.

*Non-semantic level:* We used event bursts for non-semantic analysis and applied it on the cloud service logs. Each service log can record seven types of events: DEBUG, INFO, AUDIT, WARNING, ERROR, CRITICAL, and TRACE. We considered only event types - Error and Warnings as including all types of events will reduce the extent of interpretation.

All logs were parsed for the target VM events and plotted as shown in Figure 6.7. From it, one can observe that the access and usage behavior of the target tenant VM contains some anomaly (we checked the event bursts for all the VMs connected to the victim VM. We have shown only the suspicious event burst of a VM). To increase the accuracy of the event bursts value, we considered nine months of service log events.

Simply looking at the event bursts values, the incident handler may not be able to identify the exact reason for the anomaly. To confirm the abnormality involved, we used semantic based event correlation on the target artifact (here service logs).

*Semantic-Level:* To perform correlation at this level, the prerequisite for the incident handler is to have the knowledge about when an event will be logged. If so, in which log the event traces can be found. Thus, we found a log where DOS incident traces are present (Figure 6.8).

```
ERROR: apport (pid 11542) Thu Oct 22 06:47:05 2015: called for
pid 58379, signal 11, core limit 0
ERROR: apport (pid 11542) Thu Oct 22 06:47:05 2015: executable:
/usr/lib/squid3/pinger (command line "(pinger)")
ERROR: apport (pid 11542) Thu Oct 22 06:47:05 2015:
is_closing_session(): no DBUS_SESSION_BUS_ADDRESS in environment
ERROR: apport (pid 11542) Thu Oct 22 06:47:05 2015: apport:
report /var/crash/_usr_lib_squid3_pinger.0.crash already exists
and unseen, doing nothing to avoid DoS
```

Figure 6.8: Observed traces of the anomaly in cloud service logs

Obviously, from both the above levels we can clearly say that, the *anomaly observed in event burst is due to the DOS incident*. We extended the event correlation across multiple artifacts so that details like how the incident (DOS) has been performed and what is its impact in each artifact can be known. The details about the same is discussed as stage 2.

### Correlation of events across multiple artifacts (Stage 2)

As the incident in the current case was not known initially, we did not emphasize on the techniques which vary from one incident to the other [187]. Instead, our approach works irrespective of the incident i.e. Timeline Generation and Analysis. The other advantages of using timeline for correlating the events are: (1) Visualizing the timeline of events would reveal many interesting facts in less time (2) The artifact to select and the events to analyze may not be known in all the cases which can be known using timeline.

For correlating multiple evidences, we considered three major artifacts of cloud VM namely vRAM, vDisk and Service logs to generate the timeline. We divide the entire process of timeline based correlation in to two levels namely *generating timeline* and *analyzing the events using the created timeline*.

*Generating the timeline:* The challenges we faced during timeline generation include lack of unique time format, privacy violation and absence of tools for comprehensive timeline generation.

Table 6.5: Supported formats for Disk analysis (do not support cloud vDisk analysis)

| S. No. | Disk forensic analysis tool | Supported formats |
|---|---|---|
| 1 | Encase | AFF, AF4, gfzip, raw, .e01 |
| 2 | FTK | dd, .e01, SMARTS |
| 3 | Autopsy | raw, .e01 |
| 4 | DFF | raw, .ewf, .aff |

a. *Time format variations:* Considering a single artifact itself like service logs, there are differences in the time format. For example, the *access.log* of controller stores time in the format of *DD/Mon/Year hh:mm:ss + zone* which is quite different from nova service logs: *Year-MM-DD hh:mm:ss.ms*. We observed many variations in the time format even across multiple artifacts.

b. *Privacy violation:* The timeline generated should be specific to a VM or tenant but it should not represent any events from other tenants. Without the actual segregation of the events, it leads to privacy violation especially in the case of service logs.

c. *Existing tools do not support the generation of timeline:* For the timeline generation of the target virtual machine's vDisk, we initially tried to use existing tools like DFF, sleuthkit, encase, and FTK (Table 6.5). None of them supports timeline generation for the cloud VMs as they have different formats.

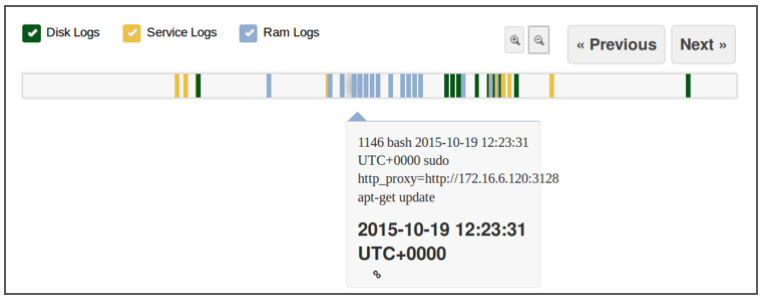



Figure 6.9: Timeline showing the description of VRAM events

We resolved these issues and the methodology followed is briefed as below.

Irrespective of the time format of events, all of them are converted to same format. Coming to the privacy preservation of the tenants, we identified certain parameters. For example, uuid, req-id pairs for each instance, pid and tid pair etc., helps in better segregation of target VM events. To resolve the third issue, we used a library called as *libqcow* which can mount the cloud images of *qcow2* format to the incident handler machine. The mounted image is given as input to the sleuth kit timeline commands - *fls* and *mactime* to get a file with events along with the proper timing information. Similarly, for getting the timeline for VRAM we used various plugins of volatility framework. After following the above methods


Figure 6.10: Timeline showing the description of Service logs events


Figure 6.11: Timeline showing the description of vDisk events

and combining those, we are able to generate the timeline as shown in Figure 6.9, Figure 6.10, Figure 6.11.

*Analyzing the events using timeline:* Time based events play a crucial role in incident handling (traditional, cloud), especially in the case of handling unknown incidents. In the current case, till now we only identified that DOS incident has been performed by a tenant. If the incident handler wants to know more information about the incident then we extend the process of event correlation using timeline analysis. From the visual timelines, we had the following observations:

- The virtual machine of the intruder used a proxy to achieve this incident as it may increase his/her chances of hiding (Figure 6.9)
- The intruder achieved DOS by over utilizing the resources (Figure 6.10)
- We found an orphan file after the incident. We recovered it and observed that it has a script to perform DOS. After the incident, the intruder deleted that file which we can clearly see from the Timeline in Figure 6.11.

These are only few important observations pertaining to the incident. Also, we identified that, using the proposed timeline based correlation across various artifacts, the incident handler can get the responses for many forensically relevant queries easily which can lead to effective incident handling. Some of them are shown in Table 6.6.

Table 6.6: Role of cloud service logs in incident handling

| **S. No** | **Query on timeline?** | **Artifact-Location(s)** |
|---|---|---|
| 1 | Is there is any event which tried privilege escalation, if so at what time? | Keystone-controller |
| 2 | When authentication failures occurred in the target VM? | Keystone-controller |
| 3 | At time $t_x$, what are the VM's of the suspected tenant which are in active state? | Nova-conductor |
| 4 | Give sequence of access times to the console of target VM from login time to logout time? | Nova-consoleauth and error.log of apache in controller |
| 5 | At specified time $t_x$, is there are any VM creation in the suspected tenant? May be the new VM may get the same IP of previously deleted VM (in our case, with the fixed pool it happens) | Nova-scheduler |
| 6 | At time $t_x$, what is the availability zone of the target VM? (very useful as it increases the chances of retrieving the data even though the VM is migrated) | Cinder-api log (Other information like provider location, attached host, creation time, launched time etc) |
| 7 | What is the host IP from which the cloud has been accessed through secure shell (ssh) and when? (For detecting malicious insiders) | Auth.log of compute node |

## 6.6 Summary

In this chapter, we took up one of the technical challenges of cloud incident handling namely event correlation and proposed a novel segregation model for the same. We validated the proposed model in two stages using a private cloud IaaS test bed: (1) homogeneous correlation and (2) heterogeneous correlation. Homogeneous correlation has been described with the help of a relevant case study- finding the host system from which the cloud VM was accessed. We have shown the process of heterogeneous correlation to achieve incident detection with the aid of codebook technique. The incident-unknown case of heterogeneous correlation is discussed using single artifact as well as multiple artifacts. Our approach of cloud event correlation is comprehensive as it it facilitates the correlation of evidences of same type as well as different types.

The work described here is an accepted and published work in [PUB1] and [PUB3] (refer page no. 136-137).

# Chapter 7

# Conclusion and Future Scope

In this thesis work, we proposed various models for effective incident handling using which multiple artifacts like vRAM, Service logs, VM snapshots and vDisk can be analyzed by addressing the architectural aspects of it (CIH).

## 7.1 Summary of Contributions

In this work, we have proposed a mechanism for incident handling in the IaaS cloud environment. The major contributions of the work are:

- Proposed a Trigger based Introspection approach for handling volatile evidences by addressing the lack of transparency between incident handler and CSP.
- Devised approaches for Event Reconstruction (specifically, Hypothesis generation) using cloud service logs and built Provenance System from VM snapshots for effective analysis during cloud incident handling.
- Designed Event Correlation approaches considering various cloud VM artifacts (Homogeneous and heterogeneous correlation in the incident known and unknown cases)

Volatile evidences like, vRAM, have a lot of forensically relevant information. However, the challenge is to increase its availability irrespective of the cloud VM memory incidents. To address this, we used Virtual Machine Introspection (VMI) technique. We proposed a trigger based introspection approach which can detect *known*, *variation of known* incidents and understand the root cause of an incident at memory level. The incident handler is required to access the hypervisor for performing VMI. Cloud, being a multi-tenant environment, the Cloud Service Provider (CSP) needs to be alerted about the suspicious activities if any, performed by the incident handler. We achieved this through the proposed ALTRA model.

Event Reconstruction (ER) has many capabilities which can be used to handle the cloud incidents effectively. In this work, we used the Hypothesis Generation capability of ER.

Generating hypothesis in cloud is challenging due to its multi-tenant nature. Also, the number of events in the target cloud evidence is huge. Taking into account these challenges, we proposed a model named SEASER which can help the incident handler arrive at accurate hypothesis generation. For the segregation phase of the model, we proposed parameter-based and session-based approaches. Then, various machine learning were used for identifying incident handler relevant events. For the aggregation phase of the model, we came up with two algorithms namely $LF_{v1}$ and $LF_{v2}$.

Cloud VM snapshots are one of the richest sources of evidences for incident handling. We proposed a model namely iCFR to improve the availability of cloud VM snapshots. Since cloud VM snapshots are generally of huge size, it is desirable to have an automatic analysis approach. For this, we proposed a model, SNAPS, which can build a provenance system from multiple cloud VM snapshots. SNAPS cannot retrieve the deleted objects from snapshots and we addressed this challenge using NLP techniques.

Event correlation helps in effective incident handling in cloud. The traces of an incident can spread across multiple evidences. Analyzing all of those individually and correlating the logical findings manually is time consuming. Thus, we proposed three ways of correlating the artifacts- *Homogeneous incident unknown, Heterogeneous known and unknown cases* to handle the cloud incidents better.

## 7.2 Future Scope of the Work

The proposed approaches for "Incident Handling in IaaS Cloud Environment using Digital Forensic Practices" are validated in the Openstack private cloud test bed. The solutions can also be tested and validated on other cloud platforms like, OpenNebula, Eucalyptus and CloudStack.

In future, approaches for performing homogeneous correlation in the incident known case can be devised. Advanced NLP techniques can be used to analyze the deleted files in the acquired VM snapshots. We proposed a root cause identification technique for volatile events. As a future work, approaches for root cause analysis for various other cloud evidences can be devised. The introspection capabilities can be used to detect more memory level incidents. Also, the recommendation engine devised for snapshot analysis can be integrated with more features. The capabilities of the proposed models can be extended to address other challenges of cloud incident handling for example, live migration, multiple availability zones etc. Additional cloud incidents may be used to validate the proposed models.

## 7.3 Concluding Remarks

Incident handling in cloud is always challenging. In this work, we presented ways in which the incident handler can deal with the occurred incidents at the IaaS cloud environment.

Collaboration among the cloud actors would help in effective incident handling in cloud. This in turn helps in improving the trust of the users in cloud systems. We believe that this research work would help the cloud community to facilitate incident handling services to the end user.

_____

_____

# List of Publications

## International Journals

**[PUB1]** BKSP Kumar Raju and G Geethakumari, SCORPIO: Segregation and Correlation for Virtual Machine Artifacts of IaaS Cloud, Journal of Network and Computer Applications (JNCA), Elsevier Publishers, 2018 (Accepted). SCI Indexed. Impact Factor: 3.5

**[PUB2]** BKSP Kumar Raju and G Geethakumari, SNAPS: Towards Building Snapshot based Provenance System for Virtual Machines in the Cloud Environment, Journal of Computers and Security, Elsevier Publishers, December 2017. SCI indexed, Impact Factor: 2.849

**[PUB3]** BKSP Kumar Raju, and G. Geethakumari. Event correlation in cloud: a forensic perspective, Springer Computing Journal, 98.11 (2016): 1203-1224. SCI Indexed, Impact Factor: 0.872

**[PUB4]** BKSP Kumar Raju, and G. Geethakumari. A trigger-based introspection approach for cloud incident handling. Inderscience, International Journal of Big Data Intelligence, 3.3 (2016): pp 163-175. Indexed in DBLP Computer Science Bibliography.

## International Conferences

**[PUB5]** BKSP Kumar Raju, Nikhil Bharadwaj Gosala and G Geethakumari, ”CLOSER: Applying Aggregation for Effective Event Reconstruction of Cloud Service Logs, Proceedings of 11th ACM International Conference on Ubiquitous Information Management and Communication ACM IMCOM 2017, Beppu, Japan, January 5-7.

**[PUB6]** BKSP Kumar Raju, Bhupendra Moharil and G Geethakumari, ”FaaSeC: Enabling Forensics-as-a-Service for Cloud Computing Systems, Proceedings the 9th IEEE/ACM International Conference on Utility and Cloud Computing (UCC), Shangai, China, Dec 6-9, 2016.

**[PUB7]** BKSP Kumar Raju and G Geethakumari, ”Timeline based Cloud Event Reconstruction Framework for Virtual Machine Artifacts”, Proceedings the 4th Springer International Conference on Advanced Computing, Networking, and Informatics (Springer ICACNI 2016), 22-24 September 2016, India.

**[PUB8]** BKSP Kumar Raju, Meera G and G Geethakumari, Cloud Forensic Investigation: A sneak peak into acquisition", IEEE, Proceedings of the Symposium on Emerging Topics in Computing and Communications (IEEE SETCAC'2015), December 16 - 19, 2015, India.

**[PUB9]** BKSP Kumar Raju and G Geethakumari, A Digital Forensic Model for Introspection of Virtual Machines in Cloud Computing, Proceedings of the IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems 2015, (IEEE SPICES 2015), February 19-21, 2015, India.

**[PUB10]** BKSP Kumar Raju and G Geethakumari, A Model for Trust Enhancement in Cloud Computing, Proceedings of the IEEE International Conference on Computing and Communication Technologies IEEE ICCCT2014, December 11-13, 2014, India, Proceedings in IEEE Explore.

**[PUB11]** BKSP Kumar Raju and G Geethakumari, A Novel Approach for Incident Response in Cloud Using Forensics, Proceedings of the ACM COMPUTE 2014, October 9-11, 2014, India; Proceedings in ACM Digital Library.

# Bibliography

[1] Grobauer, Bernd, and Thomas Schreck. Towards incident handling in the cloud: challenges and approaches. *In Proceedings of the 2010 ACM workshop on Cloud computing security workshop. ACM, 2010*.

[2] Yahoo Says 1 Billion User Accounts Were Hacked. https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html?_r=0 *Last Accessed. 30-12-2016*.

[3] Data Breach At Oracles micros Point-of-Sale Division. https://krebsonsecurity.com/2016/08/data-breach-at-oracles-micros-point-of-sale-division/ *Last Accessed. 30-12-2016*.

[4] eBay hacked: 230 million told to change passwords after major cyber attack http://metro.co.uk/2014/05/21/ebay-hack-128-million-told-to-change-password-after-major-cyber-attack-where-customer-details-were-stolen-4735660/#ixzz4cstbKAQV *Last Accessed. 21-04-2015*.

[5] E-Business, online shoppers, job-seekers urged to be suspicious. http://www.cybersecurity.my/data/content_files/44/1309.pdf *Last Accessed. 07-02-2017*.

[6] From Events to Incidents . https://www.sans.org/reading-room/whitepapers/incident/events-incidents-646 *Last Accessed. 24-01-2014*.

[7] Patrick Kral , Incident Handlers Handbook, SANS, 2011. *Accessed. 27-01-2014*.

[8] Mobile Incident Response Overview. https://books.nowsecure.com/mobile-incident-response/en/overview/index.html *Accessed. 12-02-2016*.

[9] Best practices for incident response in the age of cloud http://www.networkworld.com/article/3116011/cloud-computing/best-practices-for-incident-response-in-the-age-of-cloud.html *Last Accessed. 17-11-2016*.

[10] Planning and executing an IoT incident response. https://www.safaribooksonline.com/library/view/practical-internet-of/9781785889639/ch10s02.html *Last Accessed. 03-01-2016*.

[11] Foster, Ian, et al. Cloud computing and grid computing 360-degree compared. *In IEEE Grid Computing Environments Workshop, 2008. GCE'08. 2008*.

[12] Vaquero, Luis M., et al. A break in the clouds: towards a cloud definition. *In ACM sigcomm Computer Communication Review 39.1*, 2008. pp: 50-55.

[13] Mell, Peter, and Tim Grance. The NIST definition of cloud computing. 2011.

[14] Takahashi, Takeshi, Youki Kadobayashi, and Hiroyuki Fujiwara. Ontological approach toward cybersecurity in cloud computing. *In Proceedings of the 3rd ACM international conference on Security of information and networks*, 2010.

[15] Cloud Computing Trends. http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey *Last Accessed. 21-03-2016*.

[16] CVW Group. Cloud computing vulnerability incidents: A statistical overview. Technical Report, March, 2013

[17] Lance Whitney, https://www.cnet.com/news/amazon-ec2-cloud-service-hit-by-botnet-outage/. *Last Accessed. 12-04-2014*

[18] Higgins K.: Dropbox, wordpress used as cloud cover in new apt attacks,In: https://www.threatconnect.com/in-the-news/july-11-2013-dark-reading-dropbox-wordpress-used-as-cloud-cover-in-new-apt-attacks/, (2013), *Last accessed 03-09-2016*.

[19] Inci, Mehmet Sinan, et al,.: Seriously, get off my cloud! Cross-VM RSA Key Recovery in a Public Cloud. *In IACR Cryptology ePrint Archive*, 2015.

[20] Dow Jones Newswires, http://www.foxbusiness.com/features/2016/10/21/denial-service-attack-hits-amazon-twitter-others.html. *Last accessed. 12-11-2016*.

[21] Munteanu, Victor Ion, et al. Cloud incident management, challenges, research directions, and architectural approach. *In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014.

[22] Ab Rahman, Nurul Hidayah, Niken Dwi Wahyu Cahyani, and KimKwang Raymond Choo. Cloud incident handling and forensicbydesign: cloud storage as a case study. *In Concurrency and Computation: Practice and Experience of Wiley*, 2016.

[23] Analyst firm Forrester. https://www.snowsoftware.com/blog/2016/10/26/cloud-spend-23/#.WHXTVlV97IU *Last Accessed. 21-11-2016*.

[24] Birk, Dominik, and Christoph Wegener. Technical issues of forensic investigations in cloud computing environments. *In IEEE sixth Systematic Approaches to Digital Forensic Engineering (SADFE) workshop*, 2011.

[25] Reith, Mark, Clint Carr, and Gregg Gunsch. An examination of digital forensic models. International Journal of Digital Evidence 1.3 (2002): 1-12.

[26] Pollitt, Mark M. An ad hoc review of digital forensic models. *In IEEE Second International Workshop on Systematic Approaches to Digital Forensic Engineering*, 2007.

[27] Carrier, Brian. Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of digital evidence*, pp. 1-12, 2003.

[28] Kent, Karen, et al. Guide to integrating forensic techniques into incident response. *NIST Special Publication 10*, pp. 800-86, 2006.

[29] Cohen, M. I. PyFlagAn advanced network forensic framework. *Journal of Digital investigation 5, Elsevier*, pp. S112-S120, 2008.

[30] Kohn, Michael Donovan, Mariki M. Eloff, and Jan HP Eloff. Integrated digital forensic process model. *Journal of Computers & Security 38, Elsevier*, pp. 103-115, 2013.

[31] Zatyko, Ken and Bay, John. The Digital Forensics Cyber Exchange Principle, http://www.dfinews.com/articles/2012/02/digital-forensics-cyber-exchange-principle#.Uq83puLEqrU *Last Accessed. 12-02-2014*.

[32] Crosbie, M., Hack the Cloud: Ethical Hacking and Cloud Forensics, Cybercrime and Cloud Forensics: Applications for Investigation Processes, *IGI Global Journal*, December 2012.

[33] Ruan K., James I.J., Carthy, J., Kechadi, T. Key Terms for Service Level Agreement to Support Cloud Forensics, *Advances in Digital Forensics VIII*, Springer, pp. 201-212.

[34] Gonsowski D. Compliance in the Cloud and Implications on Electronic Discovery, Cybercrime and Cloud Forensics: Applications for Investigation Processes, *IGI Global Journal*, December 2012.

[35] Grivas S.G., Kumar, T.U., Wache H., Cloud Broker: Bringing Intelligence into the Cloud -An Event-based Approach, IEEE 3rd International Conference on Cloud Computing, pp. 544-545, 2010.

[36] Ruan K., Carthy, J., Cloud Computing Reference Architecture and its Forensic Implications: a Preliminary Analysis, *In Proceedings of the 4th International Conference on Digital Forensics & Cyber Crime*, Springer Lecture Notes, October 25-26, Lafayette, Indiana, USA.

[37] Orton I., Alva A., Endicott-Popovsky B., Legal Process and Requirements for Cloud Forensic Investigations, Cybercrime and Cloud Forensics: Applications for Investigation Processes, *IGI Global Journal*, December 2012.

[38] Ficco, Massimo, Massimiliano Rak, and Beniamino Di Martino. An intrusion detection framework for supporting SLA assessment in cloud computing. *In proceedings of fourth IEEE International Conference on Computational Aspects of Social Networks (CASoN)*, 2012.

[39] Dolan-Gavitt, Brendan, Bryan Payne, and Wenke Lee. Leveraging forensic tools for virtual machine introspection. Georgia Institute of Technology, 2011.

[40] Roussev, Vassil, et al. Cloud forensicsTool development studies & future outlook. Digital Investigation, Elsevier, pp. 79-95, 2016.

[41] Mohay, George. Technical challenges and directions for digital forensics. *International Workshop on IEEE Systematic Approaches to Digital Forensic Engineering*, 2005.

[42] British Standards Institution. BIP 0107:2008 foundations of IT service management based on Itil V3, UK. 2007.

[43] Cichonski P et. al, Computer security incident handling guide recommendations of the National Institute of Standards and Technology (NIST). Gaithersburg, 2012.

[44] West-Brown MJ, Stikvoort D, Kossakowski K-P, Killcrere G, Ruefle R, Zajicek M. Handbook for computer security incident response teams (CSIRTs). 2nd ed. Pittsburgh: Carnegie Mellon/ SEI; 2003.

[45] Alberts C, Dorofee A, Killcrece G, Ruefle R, Zajicek M. Defining incident management processes for CSIRTs: a work in progress. 2004. Pittsburgh

[46] European Network and Information Security Agency (ENISA). Good practice guide for incident management. Athens: ENISA; 2010.

[47] International Standard for Organisation. ISO/IEC 27035:2011 information technology e security techniques e information security incident management. 2011. Geneva.

[48] Grance, Tim, Karen Kent, and Brian Kim. ”Computer security incident handling guide.” NIST Special Publication 800 (2004): 61.

[49] Mitropoulos S, Patsos D, Douligeris C. On incident handling and response: a state-of-the-art approach. Computer Security, 2006; 25(5):351e70.

[50] Monfared A, Jaatun MG. Handling compromised components in an IaaS cloud installation. Journal of Cloud Computing Advance System Applications 2012.

[51] FBI Defends Disruptive Raids on Texas Data Centers. https://www.wired.com/2009/04/data-centers-ra/. *Last Accessed. 12.05.16.*

[52] List of National CSIRTs http://www.cert.org/incident-management/national-csirts/national-csirts.cfm? *Last Accessed. 21.12.2015*.

[53] Takabi, Hassan, James BD Joshi, and Gail-Joon Ahn. Security and privacy challenges in cloud computing environments. IEEE Security & Privacy 8.6, pp: 24-31, 2010.

[54] AWS suffers a five-hour outage in the US. http://www.datacenterdynamics.com/content-tracks/colo-cloud/aws-suffers-a-five-hour-outage-in-the-us/94841.fullarticle *Last Accessed. 09.03.2017*.

[55] Cloud Computing Vulnerability Incidents: A Statistical Overview. https://www.cert.uy/wps/wcm/connect/certuy/abfd80ca-3142-4d28-b99c-e8f841568dde/Cloud_Computing_Vulnerability_Incidents.pdf?MOD=AJPERES *Last Accessed. 12.03.2015*.

[56] Sen, Jaydip. Security and privacy issues in cloud computing. Architectures and Protocols for Secure Information Technology Infrastructures, pp. 1-45, 2013.

[57] Khorshed T, Ali ABMS, Wasimi SA. A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. *Future Generation Computer Systems*, Elsevier journal, 2012.

[58] Rong C, Nguyen ST, Jaatun MG. Beyond lightning: a survey on security challenges in cloud computing. Computer Electrical Eng 2013;39(1):47e54.

[59] Zissis D, Lekkas D. Addressing cloud computing security issues. Future Generation Computer Systems, Elsevier, 2012.

[60] FBI Cyber Division. https://en.wikipedia.org/wiki/FBI_Cyber_Division *Last Accessed. 21-03-2014*.

[61] Martini B, Choo K-KR. An integrated conceptual digital forensic framework for cloud computing. *Digital Investigation Journal, Elsevier*, 2012;9(2):71e80.

[62] Quick D, Choo K-KR. Digital droplets: Microsoft SkyDrive forensic data remnants. *Future Generation Computer Systems, Elsevier*, 2013a;29(6):1378e94.

[63] Quick D, Martini B, Choo K-KR. Cloud storage forensics. *Syngress*; 2014.

[64] Dennis Radar. https://en.wikipedia.org/wiki/Dennis_Rader *Last Accessed. 18.03.2015*.

[65] Computer Forensics: The new fingerprinting. http://www.popularmechanics.com/technology/security/how-to/a630/2672751/ *Accessed. 16-09-2015*.

[66] Computer Evidence. http://www.justiceforbrad.com/evidence/computer/overview.html *Last Accessed. 28-03-2015*.

[67] The people of state of New York. http://www.courts.state.ny.us/Reporter/3dseries/2010/2010_07364. *Last Accessed. 30-11-2014*.

[68] Hillary Clinton BOMBSHELL. http://www.express.co.uk/news/world/726407/Hillary-Clinton-email-scandal-FBI-Trump-president-elections *Last Accessed. 21.11.2016*.

[69] Cyber Forensics: Case studies from India. http://prateek-paranjpe.blogspot.in/p/cyber-forensics-case-studies.html *Last Accessed. 12.09.2015*.

[70] Cusumano, Michael. Cloud computing and SaaS as new computing platforms. Communications of the ACM 53.4, pp. 27-29. 2010.

[71] Genez, Thiago AL, Luiz F. Bittencourt, and Edmundo RM Madeira. Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels. Network Operations and Management Symposium (NOMS), 2012 IEEE, 2012.

[72] Hay, Brian, Kara Nance, and Matt Bishop. Storm clouds rising: security challenges for IaaS cloud computing. *44th Hawaii International Conference on System Sciences (HICSS)*, 2011.

[73] Dikaiakos, Marios D., et al. Cloud computing: Distributed internet computing for IT and scientific research. *IEEE Internet computing 13.5*, 2009.

[74] Bruneo, Dario. A stochastic model to investigate data center performance and qos in iaas cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems 25.3*, pp. 560-569, 2014.

[75] Bhardwaj, Sushil, Leena Jain, and Sandeep Jain. Cloud computing: A study of infrastructure as a service (IAAS). *International Journal of engineering and information Technology*, pp. 60-63, 2010.

[76] Grobauer, Bernd, Tobias Walloschek, and Elmar Stocker. Understanding cloud computing vulnerabilities. *IEEE Security & Privacy* 9.2 (2011): 50-57.

[77] Armbrust, Michael, et al. A view of cloud computing. *Communications of the ACM 53.4* (2010): 50-58.

[78] Khan, Suleman, et al. Cloud Log Forensics: Foundations, State of the Art, and Future Directions. *ACM Computing Surveys (CSUR) 49.1*,(2016): 7.

[79] Marty, Raffael. Cloud application logging for forensics. *In Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011.

[80] Zaferullah, Z., Anwar, F., Anwar, Z.: Digital forensics for eucalyptus. *In IEEE Frontiers of Information Technology*, pp. 110116, 2011.

[81] Zawoad, Shams, Amit Kumar Dutta, and Ragib Hasan. SecLaaS: secure logging-as-a-service for cloud forensics. Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security. ACM, 2013.

[82] Guo, H., Jin, B., Shang, T.: Forensic Investigations in Cloud Environments. In: 2012 International Conference on Computer Science and Information Processing (CSIP). pp. 248251. IEEE.

[83] Dykstra, J., Sherman, A.T.: Acquiring forensic evidence from infrastructure-as-a-service cloud computing: exploring and evaluating tools, trust, and techniques. Digit. Investig. 9, S90S98 (2012)

[84] Ruan, K.: Designing a forensic-enabling cloud ecosystem. In: Cybercrime and cloud forensics, pp. 331344. IGI Global, USA (2013)

[85] J. Dykstra, Cybercrime and Cloud Forensics, in Cybercrime and cloud forensics, K. Ruan, Ed. USA: IGI Global, 2013, pp. 156185.

[86] Ruan, Keyun, Joe Carthy, and Tahar Kechadi. "Survey on cloud forensics and critical criteria for cloud forensic capability: A preliminary analysis." Proceedings of the Conference on Digital Forensics, Security and Law. Association of Digital Forensics, Security and Law, 2011.

[87] Delport, Waldo, Michael Khn, and Martin S. Olivier. "Isolating a cloud instance for a digital forensic investigation." ISSA. 2011.

[88] Li, J., Chen, X., Huang, Q., Wong, D.S.: Digital provenance: enabling secure data forensics in cloud computing. Future Gener. Comput. Syst. (2013)

[89] Reilly,D., Wren,C., Berry,T.: Cloud computing?: pros and cons for computer forensic investigations. Int. J. Multimed. Image Process. 1, 2634 (2011)

[90] Marangos, N., Rizomiliotis, P., Mitrou, L.: Time synchronization: pivotal element in cloud forensics. Secur. Commun. Netw. (2014)

[91] Zawoad, S., Hasan, R.: Digital Forensics in the Cloud (2013)

[92] Ruan, K., Carthy, J., Kechadi, T., Crosbie, M.: Cloud forensics?: an overview. Adv. Digit. Forensics VII 1526 (2011)

[93] Chen, G., Du, Y., Qin, P., Du, J.: Suggestions to digital forensics in Cloud computing ERA. In: 2012 3rd IEEE International Conference on Network Infrastructure and Digital Content, pp. 540544. IEEE, Beijing (2012).

[94] Al Fahdi, M., Clarke, N.L., Furnell, S.M.: Challenges to digital forensics: a survey of researchers & practitioners attitudes and opinions. In: 2013 Information Security for South AfricaProceedings of the ISSA 2013 Conference, pp. 18 (2013)

[95] Ficco, Massimo. "Security event correlation approach for cloud computing." International Journal of High Performance Computing and Networking 1 7.3 (2013): 173-185

[96] Geethakumari, G., Belorkar, A.: Regenerating cloud attack scenarios using LVM2 based system snapshots for forensic analysis. Int. J. Cloud Comput. Serv. Sci. 1, 134141 (2012)

[97] Grivas S.G., Kumar, T.U., Wache H. (2010) 'Cloud Broker: Bringing Intelligence into the Cloud - An Event-based Approach', 2010 IEEE 3rd International Conference on Cloud Computing, pp. 544-545.

[98] X-Ways: X-Ways technology. http://www.x-ways.net/ Accessed. 12-04-2014.

[99] Sleuthkit: Open Source Digital Forensics. http://www.sleuthkit. org/index.php Accessed. 19-04-2014.

[100] Taylor,M., Haggerty, J.,Gresty, D.,Hegarty,R.: Digital evidence in cloud computing systems. Comput. Law Secur. Rev. 26, 304308 (2010)

[101] Almulla, S., Iraqi, Y., Jones, A.: A state-of-the-art review of cloud. In: 2014 ADFSL 9, pp. 728 (2014).

[102] Raghavan, S.: Digital forensic research: current state of the art. CSI Trans. ICT. 1, 91114 (2012).

[103] Kumar, M.: Computer Investigations. http://thehackernews.com/ 2011/09/offline-windows-analysis-and-data.html Accessed. 25.03.2014.

[104] Dykstra, J., Sherman, A.T.: Design and implementation of FROST:digital forensic tools for the openstack cloud computing platform.Digit. Investig. 10, S87S95 (2013)

[105] Damshenas, M., Dehghantanha, A., Mahmoud, R., Shamsuddin, S.: Forensics investigation challenges in cloud computing environments. cyber security. In: 2012 International Conference on Cyber Warfare and Digital Forensic (CyberSec), pp. 190194. IEEE, Kuala Lumpur (2012)

[106] Sang, T.: A log based approach to make digital forensics easier on cloud computing. In: 2013 Third International Conference on Intelligent System Design and Engineering Applications, pp. 91 94. IEEE (2013)

[107] 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, 2011 IEEE World Congress on Services, 2011, pp. 584588.

[108] J. Dykstra and A. T. Sherman, Understanding Issues In Cloud Forensics: Two Hypothetical Case Studies, in Proceedings of the 2011 ADFSL Conference on Digital Forensics Security and Law, 2011, pp. 110.

[109] D. Birk, Technical Challenges of Forensic Investigations in Cloud Computing Environments, Workshop on Cryptography and Security in Clouds, 2011, pp. 16

[110] Zawaod S, Hasan R. Cloud forensics: a meta study of challenges, approaches and open problems. Distrib Parallel Clust Comput 2013. arXiv:1302.6312v1 [cs.DC].

[111] AWS Security Centre A. AWS cloud trail, user guide. 2013. Available at, https://aws.amazon.com/documentation/cloudtrail/ [Accessed 12.12.14].

[112] CSA.Mapping the forensic standard ISO/IEC 27037 to cloud computing. 2013. Available at, https://downloads.cloudsecurityalliance.org/initiatives/imf/Mapping-the-Forensic-Standard-ISO-IEC-27037-to-Cloud- Computing.pdf [Accessed 07.02.14].

[113] INCSR. International narcotics control strategy report (INCSR): treaties and agreements. United States of America: Department of State; 2012. Available at, http://www.state.gov/j/inl/rls/nrcrpt/2012/vol2/184110. htm [Accessed 08.12.2014].

[114] Taylor M, Haggerty J, Gresty D, Lamb D. Forensic investigation of cloud computing systems. Netw Secur 2011:4e10

[115] Biggs S, Vidalis S. Cloud computing: the impact on digital forensic investigations. In: Proceedings of the 2009 International Conference for Internet Technology and Secured Transactions, (ICITST). London, UK: IEEE; 2009. p. 1e6.

[116] VMI Definition Payne, Bryan D. ”virtual machine introspection.” Encyclopedia of Cryptography and Security. Springer US, 2011. 1360-1362.

[117] Wang, Lianhai, Ruichao Zhang, and Shuhui Zhang. ”A model of computer live forensics based on physical memory analysis.” Information Science and Engineering (ICISE), 2009 1st International Conference on. IEEE, 2009.

[118] Hay, Brian, and Kara Nance. ”Forensics examination of volatile system data using virtual introspection.” ACM SIGOPS Operating Systems Review 42.3 (2008): 74-82.

[119] Nance, Kara, Matt Bishop, and Brian Hay. ”Investigating the implications of virtual machine introspection for digital forensics.” Availability, Reliability and Security, 2009. ARES’09. International Conference on. IEEE, 2009.

[120] FlorianWestphal, Stefan Axelsson, Christian Neuhaus, and Andreas Polze. Vmi-pl: A monitoring language for virtual platforms using virtual machine introspection. Digital Investigation, 11:S85S94, 2014.

[121] Bryan D Payne. Simplifying virtual machine introspection using libvmi. Sandia Report, 2012

[122] Jonas Pfoh, Christian Schneider, and Claudia Eckert. A formal model for virtual machine introspection. In Proceedings of the 1st ACM workshop on Virtual machine security, pages 110. ACM, 2009.

[123] Asit More and Shashikala Tapaswi. Virtual machine introspection: towards bridging the semantic gap. Journal of Cloud Computing, 3(1):114, 2014.

[124] Saberi, Alireza, Yangchun Fu, and Zhiqiang Lin. "HYBRID-BRIDGE: Efficiently bridging the semantic gap in virtual machine introspection via decoupled execution and training memoization." Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS14). 2014.

[125] Alan L Herrmann and Greg P Nugent. Embedded logic analyzer for a programmable logic device, May 14 2002. US Patent 6,389,558.

[126] Subodh Nimkar Umesh Bellur. List of operators for comprehensive complex event processing language framework. IITB, 2010.

[127] Grobler, C. P., C. P. Louwrens, and Sebastiaan H. von Solms. "A multi-component view of digital forensics." Availability, Reliability, and Security, 2010. ARES'10 International Conference on. IEEE, 2010.

[128] Alessandro Margara, Gianpaolo Cugola, and Giordano Tamburrelli. Learning from the past: automated rule generation for complex event processing. In Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, pages 4758. ACM, 2014.

[129] Raju, BKSP Kumar, G. Meera, and G. Geethakumari. "Cloud forensic investigation: A sneak-peek into acquisition." Computing and Network Communications (CoCoNet), 2015 International Conference on. IEEE, 2015.

[130] Cloudlytics. https://www.cloudlytics.com/main Accessed. 12.02.2016.

[131] Clifton, Brian. Advanced web metrics with Google Analytics. John Wiley and Sons, 2012

[132] Russ Anthony. Detecting Security Incidents Using Windows Workstation Event Logs. SANS Institute Reading Room, 2013.

[133] Pearl, Judea. Causality: models, reasoning and inference. Econometric Theory 19 (2003): 675-685

[134] Textor, Johannes, Juliane Hardt, and Sven Knppel. DAGitty: a graphical tool for analyzing causal diagrams. Epidemiology 22.5 (2011): 745.

[135] Dempster, Arthur P. Upper and lower probabilities induced by a multivalued mapping. The annals of mathematical statistics (1967): 325-339.

[136] Openstack admin guide for analyzing log files. In: http://docs.openstack.org/admin-guide/cli-analyzing-log-files-with- swift.html. Accessed: 07-12-2016.

[137] Casey, Eoghan. Digital evidence and computer crime: Forensic science, computers, and the internet. Academic press, 2011.

[138] Carrier, Brian D., and Eugene H. Spafford. "Defining event reconstruction of digital crime scenes." Journal of Forensic Science 49.6 (2004): JFS2004127-8.

[139] Liao, Yi-Ching, and Hanno Langweg. "Resource-Based Event Reconstruction of Digital Crime Scenes." Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint. IEEE, 2014.

[140] T. King and P. M. Chen, Backtracking intrusions, in Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, ser. SOSP 2003. New York, NY, USA: ACM, 2003, pp. 223 to 236.

[141] C. Verbowski et al., Flight data recorder: Monitoring persistent-state interactions to improve systems management, in 7th Symposium on Operating Systems Design and Implementation (OSDI 2006), November 6-8, Seattle, WA, USA.

[142] Kristinn.: Mastering the super timeline with log2timeline. In: SANS Institute (2010).

[143] Olsson, Jens, and Martin Boldt.: Computer forensic timeline visualization tool. In: digital investigation, pp. S78-S87, Elsevier, 2009.

[144] Buchholz, Florian P., and Courtney Falk.: Design and Implementation of Zeitline: a Forensic Timeline Editor. In: DFRWS. 2005

[145] Hargreaves, Christopher, and Jonathan Patterson.: An automated timeline reconstruction approach for digital forensic investigations. In: Digital Investigation, pp. S69-S79, 2012.

[146] Gladyshev, Pavel, and Ahmed Patel. Finite state machine approach to digital event reconstruction. Digital Investigation 1.2 (2004): 130-149.

[147] Khan, M. N. A., Chris R. Chatwin, and Rupert CD Young. A framework for post-event timeline reconstruction using neural networks. *digital investigation* 4.3 (2007): 146-157.

[148] Case, Andrew, et al. FACE: Automated digital evidence discovery and correlation. *digital investigation* 5 (2008): S65-S75.

[149] Thorpe, Sean, Indrajit Ray, and Tyrone Grandison. A secure Log Cloud Forensic Framework. *Proceedings of the International Conference on Cybercrime, Security and Digital Forensics*. Glasgow, UK. 2011.

[150] Pearl, Judea. Causality: models, reasoning and inference. *Econometric Theory* 19 (2003): 675-685.

[151] Openstack4j-Fluent Openstack SDK for java, http://www.openstack4j.com/javadoc/. *Last Accessed: 03-04-2014*.

[152] Batista, Gustavo EAPA, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data, *ACM Sigkdd Explorations Newsletter* 6.1 (2004): 20-29.

[153] Wang, Wei. A graph oriented approach for network forensic analysis. IOWA state university, Digital Repository, 2010.

[154] Backup Azure virtual Machines, July 2016. Available at: https://azure.microsoft.com/en-in/documentation/articles/backup- azure-vms/ Accessed. 12.09.2016.

[155] Amazon EBS snapshots, (Accessed) June 2016. Available at: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSSnapshots.html Accessed. 12.07.2016.

[156] Belorkar, Abha, and G. Geethakumari. Regeneration of events using system snapshots for cloud forensic analysis. *Annual IEEE India Conference*. IEEE, 2011.

[157] Almulla, Sameera, Youssef Iraqi, and Andrew Jones. A Distributed Snapshot Framework for Digital Forensics Evidence Extraction and Event Reconstruction from Cloud Environment. *IEEE 5th International Conference on Cloud Computing Technology and Science* (CloudCom), . Vol. 1. IEEE, 2013.

[158] Roussev, Vassil, et al. Cloud forensicsTool development studies & future outlook. *Digital Investigation*, 2016.

[159] De Marco, Lucia, M-Tahar Kechadi, and Filomena Ferrucci. Cloud forensic readiness: Foundations. *Digital Forensics and Cyber Crime*. Springer International Publishing, 2013. 237-244.

[160] Kebande, Victor R., and Hein S. Venter. A Cloud Forensic Readiness Model Using a Botnet as a Service. *The International Conference on Digital Security and Forensics* (DigitalSec2014). The Society of Digital Information and Wireless Communication, 2014.

[161] Raju, BKSP Kumar, and G. Geethakumari. An advanced forensic readiness model for the cloud environment. *International Conference on IEEE Computing, Communication and Automation* (ICCCA), 2016.

[162] Dept. of statistics and operations, Spatio-temporal models, University of North Carolina at Chapel Hill. Available at: http://www.stat.unc.edu/faculty/rs/s321/spatemp.pdf Accessed. 25.07.2014.

[163] Pelekis, Nikos, et al. Literature review of spatio-temporal database models. *The Knowledge Engineering Review* 19.03 (2004): 235-274.

[164] Fernndez, Delia Cabrera, Rolando Grave de Peralta Menndez, and Sara L. Gonzlez Andino. Some limitations of spatio temporal source mod- els. *Brain topography* 7.3 (1995): 233-243.

[165] Gehani, Ashish, and Dawood Tariq. SPADE: support for provenance au- diting in distributed environments. *In proceedings of the 13th International Middleware Conference*. Springer-Verlag New York, Inc., 2012.

[166] Somak Das, ProTrack: A Simple Provenance tracking Filesystem, MIT.

[167] Wagner, Alan. ProFS: A lightweight provenance file system, MIT, 2012.

[168] Muniswamy-Reddy, Kiran-Kumar, Peter Macko, and Margo I. Seltzer. Provenance for the Cloud. FAST. Vol. 10. 2010

[169] Amerini, Irene, et al. A sift-based forensic method for copymove attack detection and transformation recovery. *IEEE Transactions on Information Forensics and Security* 6.3, pp. 1099-1110, 2011.

[170] Sarala, R. et.al. Information security risk assessment under uncertainty using dynamic Bayesian networks. *International Journal of Research in Engineering and Technology*, pp. 304-309, 2014.

[171] Roger A. Grimes, 7 sneak attacks used by today's most devious hack- ers, Available at: http://www.infoworld.com/article/2610239/malware/7-sneak-attacks-used-by-today-s-most devious devious-hackers.html Accessed. 12.05.2016.

[172] Battistoni et.al. CURE: Towards enforcing a reliable timeline for cloud forensics: Model, architecture, and experiments. *Computer Communications*, 2016.

[173] Carrier, Brian. File system forensic analysis. *Addison-Wesley Professional*, 2005.

[174] Spyridopoulos T. and Katos V. (2013) Data Recovery Strategies for Cloud Environments, Cybercrime and Cloud Forensics: Applications for Investigation Processes, Ed. 623 Ruan K, *IGI Global*, December 2012.

[175] Calhoun W, Coles D. Predicting the types of file fragments. *In: Proceedings of the 2008 Digital Forensics Research Conference (DFRWS)*, 2008.

[176] Axelsson S. The normalized compression distance as a file fragment classifier. *In: Proceedings of the 2010 Digital Forensics Research Conference* (DFRWS), 2010.

[177] Conti G, Bratus S, Sangster B, Ragsdale R, Supan M, Lichtenberg A, et al. Automated mapping of large binary objects using primitive fragment type classification. *In: Proceedings of the 2010 Digital Forensics Research Conference* (DFRWS), 2010

[178] Li Q, Ong A, Suganthan P, Thing V. A novel support vector machine approach to high entropy data fragment classification. *In: Proceedings of the South African Information Security Multi Conference* (SAISMC 2010); 2010.

[179] Jeon, Jiwoon, W. Bruce Croft, and Joon Ho Lee. Finding semantically similar questions based on their answers. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005.

[180] Zhang, Wen, Taketoshi Yoshida, and Xijin Tang. A comparative study of TF-IDF, LSI and multi-words for text classification. *Expert Systems with Applications* 38.3 pp.2758-2765, 2011.

[181] Kavulya, Soila P., et al. Failure Diagnosis of Complex Systems. Resilience Assessment and Evaluation of Computing Systems. *Springer Berlin Heidelberg*. pp 239-261, 2012.

[182] NIST cloud computing forensic science challenges. (2014). [Online]. Available: http://csrc.nist.gov/publications/drafts/nistir-8006/draft nistir 8006.pdf *Last Accessed. 27.01.2015*

[183] Tifiany, Michael. A survey of event correlation techniques and related topics. Research paper, *Georgia Institute of Technology*, 2002.

[184] Dayan, Tal. Event correlation in cloud computing. *Google U.S. Patent* Application 12/841,371, 2012.

[185] Bohra, Ata E. Husain, and Vipin Chaudhary. VMeter: Power modelling for virtualized clouds. *IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum* (IPDPSW), 2010.

[186] Yi, Sangho, Derrick Kondo, and Artur Andrzejak. Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud. *3rd International Conference on Cloud Computing (CLOUD)*, IEEE.

[187] Ahmad, Mushtaq. Security risks of cloud computing and its emergence as 5th utility service. *Information Security and Assurance*. Springer Berlin Heidelberg, pp. 209-219, 2011.

[188] Garfinkel, Simson L. Forensic feature extraction and cross-drive analysis. *digital investigation* 3 : PP 71-81, 2006.

[189] Case, Andrew, et al. FACE: Automated digital evidence discovery and correlation. *digital investigation*: S65-S75, 2008.

[190] Zeng, Yuanyuan, Xin Hu, and Kang G. Shin. Detection of botnets using combined host-and network-level information. *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2010.

[191] Flaglien, Anders, Katrin Franke, and Andre Arnes. Identifying malware using cross-evidence correlation. *Advances in Digital Forensics VII*. Springer Berlin Heidelberg, pp 169-182.

# Appendix I
# Openstack Cloud Test Bed

Many organizations are using Openstack cloud for managing their daily business activities. Openstack or any cloud reduce the operational and maintenance costs. In this thesis, we validated the proposed approaches using Openstack private cloud test bed.

Openstack is a cloud computing platform which contains many inter-operable services.

- Dashboard: This is an interface between the user and the cloud. Using dashboard, the user can receive the services from the cloud environment.
- Compute (nova): The complete cycle of an instance is managed using this service starting from scheduling to termination.
- Object storage: All the unstructured objects are stored and retrieved via RESTful API.
- Block Storage (cinder): This provides persistent capabilities for each instance in terms of volumes. The volumes can be disassociated from an instance and can be attached to any other instance at the tenant level.
- Identity service (keystone): This service authorizes and authenticates all the running cloud services.
- Image service (Glance): This is used to store and retrieve the images and this is mainly used during instance provisioning.
- Telemetry (Ceilometer): This service monitors the resource usage at the instance and tenant level and helps in scaling.

Openstack icehouse can be deployed using two different types of architectures. (1) Legacy networking and (2) Neutron networking. Both the type of architectures provide the same services to the end user and only the number of nodes required for the installation differs. In this thesis, we use Openstack cloud testbed with legacy networking architecture (see below).

Details of configuration are as follows:

- controller node : hostname - controller
  - em1 : external network (172.16.6.xxx)
  - em2 : management network (10.0.0.11)

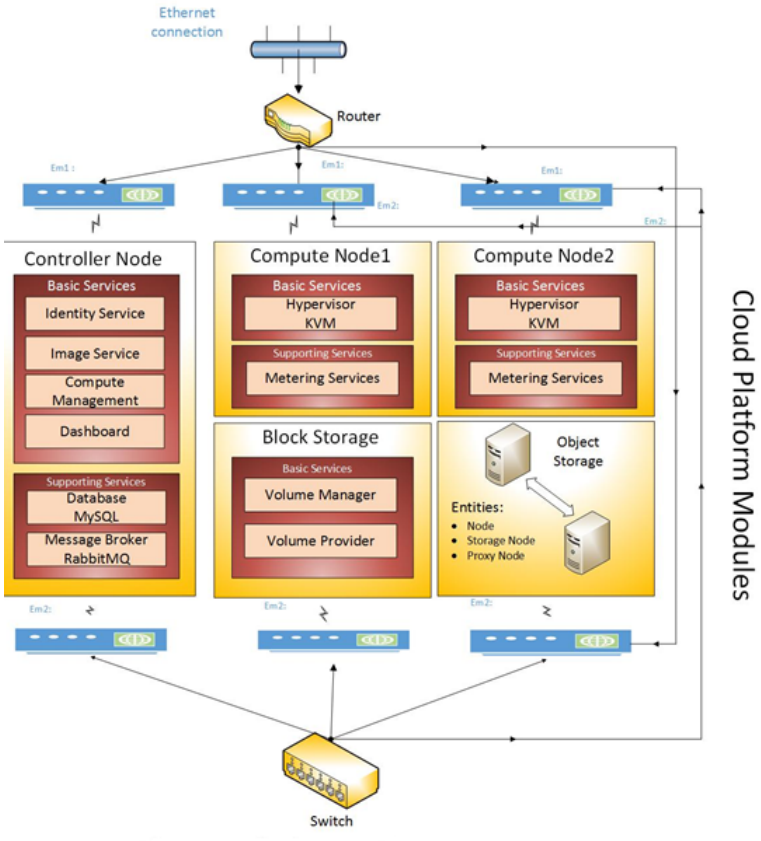



Figure A.1: Openstack cloud test bed setup in our lab

- compute node : hostname  compute
  - em1 : external network (172.16.6.xxx)
  - em2 : management network (10.0.0.31)
- Storage node : hostname  Block Storage
  - em1 : external network (172.16.6.xxx)
  - em2 : management network (10.0.0.41)
- Storage node : hostname  Object Storage
  - em1 : external network (172.16.6.xxx)
  - em2 : management network (10.0.0.51)

Compute node uses a FlatDHCP manager. After installing the specified cloud services, we can create instances using the openstack dashboard and then use the cloud capabilities.

# Appendix II
# Developed CFI Tool

We developed a tool named Cloud Forensic Investigator (CFI) which is first of its kind to perform forensic investigation in the IaaS cloud environment.

## Design of the CFI tool

Before we start developing CFI, we had many designs of the tool at multiple stages. Here, we present the main flow of the tool in terms of its various phases.

The crime scene is the virtual and physical infrastructure of the cloud provider where the evidences of the crime exist. The investigation includes five major stages - identification, acquisition, preservation, analysis and presentation. Proposed conceptual diagram of the process of forensic investigation in cloud systems is given in the below figure. For any investigation to proceed, the investigator must have access to the cloud infrastructure. If the investigator is a trusted third party, the CSP provides credentials to log in to the infrastructure, thereby also ensuring access control over him via roles, privileges and other policies.

The first step of investigation is to identify artifacts to be collected and analyzed. The type of artifacts chosen depends on the type of incident reported. In cloud forensics, the artifacts relevant to investigation form a superset of those required for traditional digital forensics. As virtualization is the backbone of multi-layered cloud systems, artifacts collected from the virtual layer are as important as those from the underlying software and hardware layers. This includes artifacts from the Virtual Machine (VM) instances as well as those from the hypervisor or Virtual Machine Monitor (VMM). Once the artifacts are identified, they should be cloned or imaged perfectly without causing any change to their original state. Preservation ensures integrity of the artifacts collected. Usually an investigator generates a one-way checksum of the original artifact as well as its copy to ensure that its state is preserved. Any modification to the artifact will change the checksum, which can be identified during verification.

Acquisition is the phase during which we collect chosen artifacts for analysis. As most of the artifacts relevant to cloud forensics are at the virtual layer, static analysis alone is not sufficient. The investigator might need memory images for analyzing running processes and other volatile information. As shown in the below figure, various types of artifacts can be collected from various layers of the cloud. From the Host OS, the investigator can collect access logs, event logs and so on. Also, all the services of the cloud provider run at the host OS. Hence service logs pertaining to the specific cloud architecture can also be found at

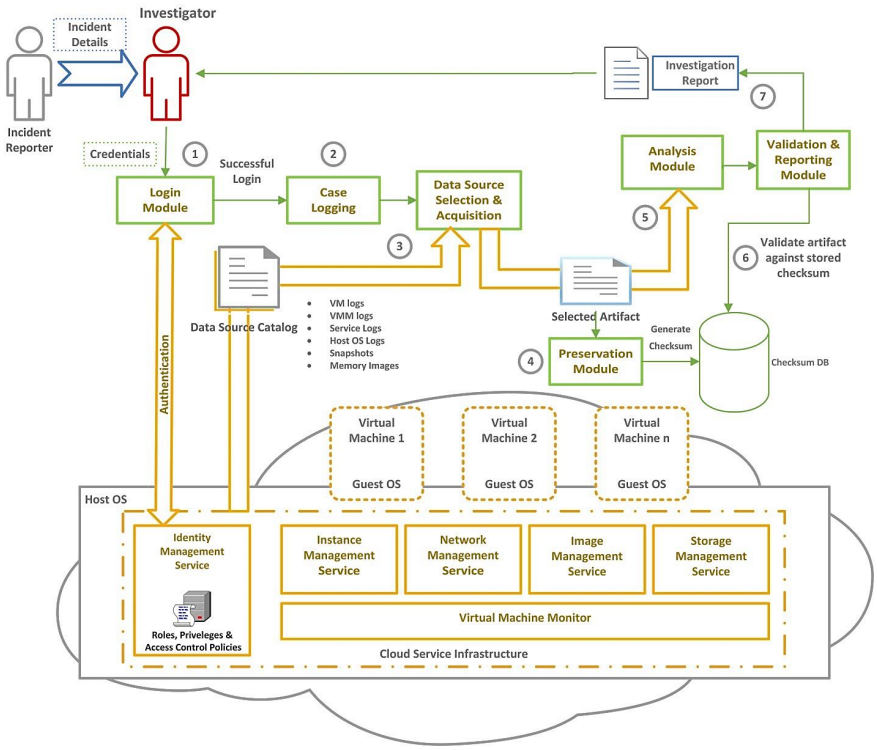



Figure A.2: Conceptual model for the developed CFI tool

the host OS. The hypervisor logs and metadata regarding VMs are also available at the host OS. The guest OS of the virtual machines contain logs pertaining to user access and usage. These along with snapshots, images and other artifacts obtained by live acquisition provide the baseline for forensic analysis.

During the analysis phase the data collected during the acquisition phase is extracted and scrutinized for identifying relevant information suggesting malicious activity. Cloud forensics investigation typically involves huge volumes of data. Hence we have to filter the content based on the reported incident. For example, if we know the details of the user suspected in the investigation we can eliminate log entries from other users and focus only on the person of interest.

In some scenarios, the data may be hidden or even deleted. To recover such obstructed content, the investigator may have to employ recovery techniques. In cloud forensics, due to the presence of volatile data, we might need to use live forensics along with static forensic analysis.

## Evidences considered by the CFI tool

The developed Cloud Forensic Investigator (CFI) tool capabilities can be categorized into three phases- Acquisition, Analysis and Reporting. The CFI tool can acquire the following evidences from the IaaS cloud environment.

## Artifact Acquisition Phase

- vDisk: Each virtual machine will have its own persistent storage and it is securely acquired and transferred to the investigators environment.
- vRAM: The target virtual machines RAM is imaged using light weight kernel imager and it is then transferred to the system where CFI is running. The tool ensures the evidence integrity at multiple levels.
- Service logs: The selected logs by the investigator are pre-processed at the cloud side and the tool facilitates to query them using multiple filters.
- Snapshots: All the available snapshots of the target virtual machine are shown to the investigator. Then the selected snapshots were transferred to the investigators environment. We calculate the checksum of the snapshot at the receiver end and verify that against with the actual checksum at the cloud end.

## Artifact Analysis Phase

Once the evidences are acquired to the system where the CFI tool is running then the tool provides various analysis capabilities i.e.

- vDisk: The acquired vDisk format is Qcow2 and its file system is completely extracted. The tool can tag the files according to the investigators requirement, can detect the obfuscated files, can sort the files according to the file type, can generate the visual timeline, can detect the hidden files, can show the complete metadata associated with the file system level and file level etc.
- vRAM: The tool provides various networking, system and process level information. For example, the process level data that the tool can provide is, list of active processes, prints a parent/child relationship, detecting hidden processes, cross-reference of processes based on multiple sources, open file descriptors and their paths for each running process etc.
- Service logs: The tool provides events in the service logs which pertains to the selected instance from the target tenant. This module also facilitates the investigator with tuple and attribute filters.
- Snapshots: Each loaded snapshot into the CFI tool can perform indexing, cross artifact analysis, viewing capabilities for various file types, gallery view of the images, query engine etc.

Finally, CFI can generate comprehensive report including all the logical conclusions made during the process of investigation. The report also stores all the case details, tenant details, evidences acquired and associated metadata.

# Glossary

**Incident**: This violates the existing security policies of the underlying cloud environment and then the nodes/virtual machines hosted in the cloud behaves unexpectedly.

**Cloud Incident Handler (CIH)**: In this thesis, we consider CIH as an entity who can handle cloud incidents using digital forensic practices.
**Cloud Tenant**: He/she is a consumer of cloud services where it can have multiple users and many virtual machines (or instances)

**Evidences**: To conduct investigation, the forensic examiner needs the traces of the occurred incident which are available as a form of evidence. The four major evidences in the IaaS cloud environment are vRAM, Service logs, Snapshots and vDisk. The evidence (or artifact) to choose depends on the type of the occurred incident.

**Actors**: In the entire process of cloud incident handling, various actors are involved i.e. Cloud Service Provider (CSP), is an on-demand service facilitator to the customer on rental basis ; Incident Handler in our case, is a forensic expert and he/she can be given support by the CSP organization; Cloud carrier, is the network provider that facilitates the customer to receive configured cloud services.

**Virtual Machine Monitor (VMM)**: This is also called as Hypervisor. VMM hosts all the virtual machines and provides the guest operating system to manage its operations using virtual operating platform.

**Virtual Machine Introspection (VMI)**: This technique monitors the target virtual machine and captures the guest operating system events from the VMM level. The acquired events through VMI are less contaminated and they increase the chance of legal admissibility in the court of law.

**Hypothesis generation**: In general, the cloud evidences are of huge size and it is difficult for the investigator to generate the hypothesis about the occurred incident. But once the framed hypothesis is accurate, it further benefits the investigator to generate quick logical conclusions about the current case being investigated.

**Provenance**: It is metadata which can represent the history of an object in the target evidence.

**Event Correlation**: It provides the association among two or more events and helps the incident handler to get more comprehensive picture about the occurred incident.

**Cloud Forensic Readiness**: This can identify the possible sources of evidences before the incident or while the incident is happening.

**Digital Forensics**: Scientific principles used by the investigator to handle the digital crime scene.

**Encase**: This is a forensic tool which is used to conduct digital investigation in a forensically sound manner.

**FTK**: This is a standard digital forensic tool which can help the investigator to have quick logical conclusions and can increase the admissibility of the generated report.

**Sluethkit**: This is an open source digital forensic tool available for three major operating systems- Linux, Windows and Mac.

# Biography: BKSP Kumar Raju Alluri

Mr. BKSP Kumar Raju is a Research Scholar in BITS Pilani Hyderabad Campus. He joined in BITS in the academic year 2013-14. His research interests include Cloud Computing, Digital Forensics, Operating systems, and Distributed Systems. Previously, he completed his M.Tech from University College of Engineering, JNTUK, VZM in 2013.

Mr. Kumar Raju is a part of DIT funded project on *Design and development of Cloud Forensic Toolkit*. Recently, the project has successfully completed and then deployed to DIT, Central Government of India. The tool developed has taken appreciation from various Cyber experts from industries and government agencies.

He also published papers in many international conferences (IEEE, ACM, Springer) and peer-reviewed journals (Springer, Elsevier) as part of his Ph.D work. Also, during M.Tech, he had published his work in IEEE international conferences. He is an organizing member for the two events conducted for showcasing the developed tool and its capabilities.

# Biography: Dr. G. Geethakumari

Dr Geetha joined the Dept. of Computer Science, BITS Pilani, Hyderabad Campus in 2008. Prior to this, she worked as a faculty at the National Institute of Technology, Warangal. Dr Geetha received Ph.D. in Computer Science from University of Hyderabad. Her Ph.D. thesis was titled Grid Computing Security through Access Control Modelling. Dr. Geetha has many international publications (in reputed conferences and peer-reviewed journals) to her credit. Her present areas of research interests include: information security, cloud computing - cloud security challenges and IoT security concerns like data protection and privacy.

Dr Geetha has successfully executed a funded project on Cloud Forensics sponsored by DeitY, Govt. of India and is also a Project Review Steering Group (PRSG) member for a DIT-funded project on Cloud Interoperability. She organized a Symposium on Cloud Computing and Digital Forensics (SoC-D 2015) on September 4, 2015 in BITS Pilani, Hyderabad Campus. As part of the funded project on Cloud Forensics, Dr Geetha organized a tech showcase event on January 30th, 2017 at BITS Pilani, Hyderabad Campus. Titled ”Unveiling CFIT: Cloud Forensic Investigator Toolkit - A Technology Showcase”, the event was aimed at bringing out the design and development aspects of CFIT.

Dr Geetha got elevation to Senior Member Grade, IEEE from February, 2018. Her publications have reached over 1200 reads and 160 citations in ResearchGate. Dr Geetha has produced four Ph.Ds. Presently she is guiding two scholars. A research article written by Dr Geetha and Dr K P Krishnakumar titled Detecting misinformation in online social networks using cognitive psychology, was cited in a book Truth Decay: An Initial Exploration of the Diminishing Role of Facts and Analysis in American Public Life, authored by Jennifer Kavanagh and Michael Rich and published in January 2018. The research paper titled An Efficient Secure Data Aggregation Technique for Internet of Things Network: An Integrated Approach Using DB-MAC and Multi-path Topology and co-authored by Dr Geetha and Sruthi Sagi has been cited in a paper in the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, June 2017 Edition.

Dr Geetha has been in the forefront of technical activities at BITS-Pilani, Hyderabad Campus. She has been the Faculty Advisor for Computer Science Association during 2008-2011. Presently she is the IEEE Student Branch Counselor, BITS-Pilani, Hyderabad Campus. She is also the Coordinator for the Linux User Group, BITS Pilani, Hyderabad Campus. Dr. Geetha is a Senior Member, IEEE as well as Member, IEEE Computer Society. She is also a Professional Member, ACM. She was the Organizing Committee Member for the IEEE INDICON Conference conducted in BITS Pilani, Hyderabad Campus during December

16 - 18, 2011. Dr Geetha was the Publicity Co-Chair for the IEEE Prime Asia Conference hosted by BITS Pilani, Hyderabad Campus during December 5-7, 2012. She was the Organizing Committee Member for the Workshop on Advances in Image Processing and Applications held in BITS Pilani, Hyderabad Campus during October 26 - 27, 2013. She was in the Organizing Committee for the National Seminar on Indian Space Technology - Present and Future (NSIST-2014) held at BITS Pilani Hyderabad Campus on 1st May, 2014.

Dr Geetha has given many guest lectures and tutorial sessions on topics in emerging areas such as Internet of Things (IoT) security, cyber security, and cloud computing security. She has been a member of the Technical Program Committees of various IEEE International Conferences. An extract from the paper 'A taxonomy for modelling and analysis of diffusion of (mis)information in social networks', co-authored by Dr Geetha and published in the International Journal of Communication Networks and Distributed Systems, Vol. 13, No. 2, 2014, pp.119-143, by Inderscience Publishers, was selected for a press release on *Semantic attacks in online social media*.