# MULUNGUSHI UNIVERSITY

## SCHOOL OF ENGINEERING AND TECHNOLOGY (SET)

*DEPARTMENT OF INFORMATION COMMUNIOCATION TECHNOLOGY*

**PROJECT TITLE:** ***UNVEILING SHADOWS IN THE CLOUD: FORENSIC APPROACHES IN MULTI-TENANT ENVIRONMENTS***


**NAME** : **TEMBO MADALITSO**

**STUDENT NUMBER** : **202107588**


**COURSE** : **CAPSTONE PROJECT**

**COURSE CODE** : **ICT 431**

**PROGRAMME** : **BACHELOR OF SCIENCE IN COMPUTER SCIENCE**

**SUPERVISOR** : **DR SINYINDA MUWANEI**

***THIS REPORT IS SUBMITTED IN PARTIAL FULFILMENT FOR THE AWARD OF BACHELOR OF COMPUTER SCIENCE FOR THE 2024/2025 ACADEMIC YEAR***

# Table of Contents

# DECLARATION

I, Tembo Madalitso, hereby declare that the project titled *"Unveiling Shadows in the Cloud: Forensic Approaches in Multi-Tenant Environments"* is a work born of my own hands and mind. I affirm that any material drawn from external sources has been meticulously acknowledged, honouring all inspirations and insights that contributed to this work.

# ACKNOWLEDGEMENTS

I would like to extend my heartfelt gratitude to my supervisor, Dr. Muwanei Sinyinda, for his invaluable guidance, patience, and dedication in overseeing this project. His insight and encouragement, especially during challenging moments, were instrumental in bringing this work to completion.

I am also thankful to the faculty in the School of Engineering and Technology at Mulungushi University. Their expertise and support throughout this project provided me with the tools and knowledge needed to reach this stage.

To my friends and colleagues, thank you for your insights, contributions, and the motivation you offered along the way. Your camaraderie made this journey all the more fulfilling. Special thanks to Fumuyane Kaira-Simbyakula.

Finally, I am deeply grateful to my parents, whose unwavering support and encouragement have been my anchor. Their belief in my abilities has fuelled my perseverance, making this achievement possible.

Thank you all for your invaluable contributions to this project.

# ABSTRACT

This project proposes a robust forensic framework tailored to the unique requirements of multi-tenant cloud environments, specifically designed to operate with minimal reliance on Cloud Service Providers (CSPs). Emphasizing privacy-preserving data acquisition, this framework addresses core challenges in cloud forensics, including the complexities of data collection and the need to uphold tenant privacy. Through strategic collaboration with Internet Service Providers (ISPs), this approach seeks to independently capture and preserve forensic data outside the cloud infrastructure, thus enhancing the reliability and completeness of evidence. By establishing an independent, privacy-conscious forensic methodology, this research aims to strengthen investigative capabilities within cloud ecosystems.

**Keywords:** Cloud Forensics, Multi-Tenant Environments, Privacy, Cybersecurity, Internet Service Providers

# LIST OF FIGURES

# LIST OF TABLES

# ACRONYMS AND ABBREVIATIONS

| Abbreviation | Full Term |
|---|---|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| CSP | Cloud Service Provider |
| DDoS | Distributed Denial of Service |
| FaaS | Forensics as a Service |
| IaaS | Infrastructure as a Service |
| ISP | Internet Service Provider |
| LEA | Law Enforcement Agency |
| PaaS | Platform as a Service |
| SaaS | Software as a Service |
| SIEM | Security Information and Event Management |
| VM | Virtual Machine |

*Table 1*

# CHAPTER 1 – INTRODUCTION

## 1.1 Introduction

Cloud computing has revolutionized how data is stored, managed, and accessed, providing organizations with unparalleled scalability, flexibility, and cost efficiency. (Tikkha, August 2024) As businesses increasingly rely on cloud environments to handle sensitive data, these systems have become integral to modern infrastructure. Cloud forensics, a specialized branch of digital forensics, plays a crucial role in investigating cybercrimes and security breaches within these environments. (Munirah Maher Alshabibi, 22 August 2024) Its applications span across various sectors, including law enforcement, corporate security, and regulatory compliance, where digital evidence from cloud systems is critical for resolving disputes, identifying threats, and prosecuting cybercriminals.

In multi-tenant environments, where multiple organizations share the same cloud infrastructure, the role of cloud forensics becomes even more significant. These environments introduce unique challenges for investigators, such as managing data co-mingling, ensuring tenant privacy, and navigating jurisdictional complexities. Despite its importance, traditional forensic methods often fall short in cloud settings due to the high degree of control exercised by Cloud Service Providers (CSPs). Investigators face barriers in accessing evidence, as CSP policies and regulatory requirements can restrict timely cooperation and data disclosure. These limitations compromise the integrity and reliability of digital evidence, hindering effective investigations. (Munirah Maher Alshabibi, 22 August 2024)

In addition, privacy concerns in multi-tenant environments further complicate forensic processes. Evidence collection in these settings risks exposing data from other tenants sharing the same infrastructure, potentially violating regulations like the General Data Protection Regulation (GDPR). Addressing these challenges requires innovative forensic frameworks that prioritize privacy while ensuring reliable evidence collection and analysis. (chloe, September 1, 2024)

This project seeks to develop a forensic framework tailored for multi-tenant cloud environments to overcome these obstacles. By minimizing reliance on CSPs, the framework introduces alternative evidence acquisition strategies, such as partnering with Internet Service Providers (ISPs). This approach enables independent data collection, ensuring privacy compliance and maintaining data integrity. Through this research, the project aims to enhance cloud forensic methodologies, strengthening investigative capabilities and fostering trust in shared cloud infrastructures.

### 1.1.1 The process of cloud forensics

The Cloud forensics process involves the following phases 1. Event identification, 2. Evidence identification, 3. Collection of evidence, 4. Analysis of evidence, 5. Interpret evidence, 6. Present in the court of law as shown in Figure 1. (1) (Sheena Mohammed, 2023 08(01))

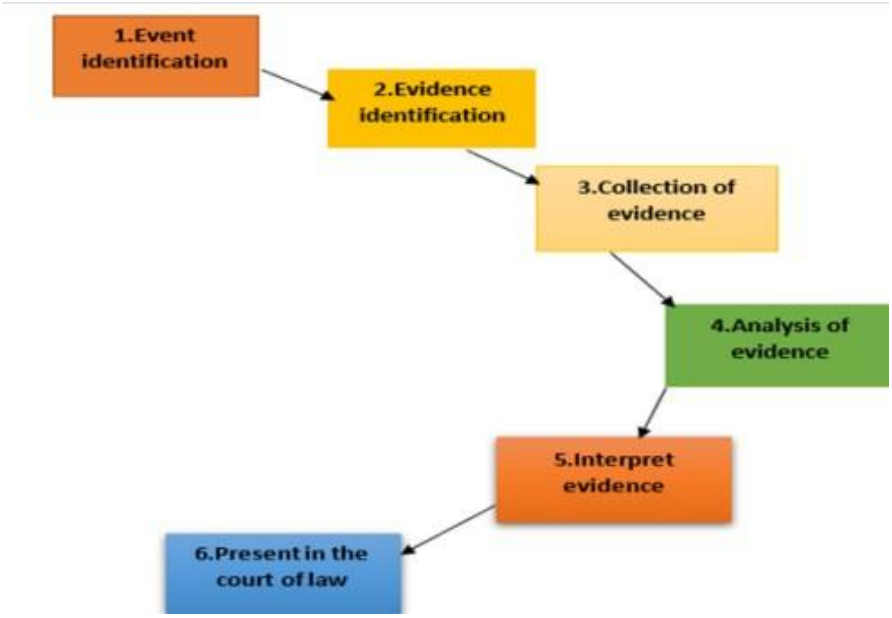


*Figure 1*

## 1.2 Problem Statement

Multi-tenant cloud environments present significant challenges for forensic investigations due to data co-mingling, privacy concerns, and heavy reliance on Cloud Service Providers (CSPs) for evidence access. CSPs often control critical evidence but may not cooperate due to legal, technical, or privacy constraints, limiting investigators' ability to gather timely and reliable evidence. This dependency hinders the integrity and completeness of forensic findings. The lack of a privacy-conscious, independent forensic framework exacerbates these issues, leaving investigators unable to effectively address cyber incidents in multi-tenant cloud environments. (Munirah Maher Alshabibi, 22 August 2024)

## 1.3 Aim

The aim of this project is to develop a robust framework for conducting cloud forensics in multi-tenant environments, specifically addressing challenges related to data acquisition and tenant privacy.

## 1.4 Objectives

**Conduct In-Depth Research on Cloud Forensics Challenges**

- Identify and analyse unique challenges in cloud forensics, focusing on multi-tenant environments. This includes issues related to data acquisition, privacy, and access controls in shared cloud infrastructures.

**Design a Privacy-Conscious Cloud Forensics Framework**

- Develop a forensic framework tailored for multi-tenant cloud environments that prioritizes privacy-preserving methods for evidence collection and investigation.

**Simulate a Controlled Cloud Environment for Testing**

- Build a simulated multi-tenant cloud environment to test the developed framework under realistic conditions, focusing on data sharing and isolation mechanisms.

**Validate the Framework through Forensic Analysis and Data Integrity Testing**

- Test the framework in simulated forensic scenarios to evaluate its effectiveness in terms of accuracy, completeness, and privacy preservation of forensic data collected.

## 1.5 Project Scope

### 1.5.1 In-Scope Activities

- Research current cloud forensic challenges, emphasizing privacy and technical limitations.
- Develop a forensic framework focused on accurate data collection and tenant privacy preservation.
- Simulate a multi-tenant cloud environment to test the framework's functionality.
- Validate the framework using controlled test cases and measure key performance metrics.

### 1.5.2 Out-of-Scope Activities

- Conducting live tests on commercial cloud platforms like AWS or Azure due to privacy and regulatory concerns.
- Creating entirely new forensic tools; instead, existing tools will be adapted for use in the framework.
- Performing comprehensive security audits or penetration testing of cloud platforms.

## 1.6 Project Justification

With the rapid adoption of cloud computing across industries, multi-tenant environments have become a standard in delivering scalable, cost-effective services. (Tikkha, August 2024) However, the shared nature of these environments presents unique challenges when conducting forensic investigations. Traditional forensic techniques are insufficient in cloud settings due to issues surrounding **data acquisition**, **tenant privacy**, and **resource isolation**. (Munirah Maher Alshabibi, 22 August 2024)

### 1.6.1 Key Reasons for the Project

- Increasing Cloud Reliance and Cybersecurity Threats
- Challenges in Existing Cloud Forensics Practices
- Legal and Compliance Considerations
- Contribution to Knowledge and Industry Standards
- Practical Application and Future Use Cases

## 1.7 Summary/Conclusion

This chapter highlights the importance of cloud forensics in addressing challenges in multitenant cloud environments, such as data co-mingling, tenant privacy, and reliance on Cloud Service Providers (CSPs). The project aims to develop a privacy-conscious forensic framework that enables independent evidence acquisition and ensures data integrity. Objectives include researching challenges, designing and testing the framework in a simulated environment, and validating its effectiveness. The scope excludes live testing on commercial platforms and creating new tools, focusing instead on adapting existing ones. The project is justified by the growing reliance on cloud computing and the need for improved forensic capabilities.

# CHAPTER 2 – LITERATURE REVIEW

## 2.1 Introduction

The purpose of this literature review is to explore and critically analyse existing research on cloud forensics, with a focus on the unique challenges posed by multi-tenant environments. This review aims to identify the current approaches, methodologies, and limitations of cloud forensic practices, particularly concerning **data acquisition**, **privacy protection**, and **data isolation**.

## 2.2 Related Literature

The field of cloud forensics has evolved with various frameworks developed to address the specific needs and challenges associated with investigating cloud environments. However, the unique characteristics of multi-tenant environments—such as data isolation, privacy, and dependency on cloud service providers (CSPs)—introduce complexities not fully addressed by traditional forensic frameworks.

### 2.2.1 Forensic Frameworks for Cloud Environments

1. **FROST (Forensic OpenStack Tools)** (Sheena Mohammed, 2023 08(01)) ○ **Overview:** Provides forensic capabilities in OpenStack environments, including collecting virtual disk images and firewall logs while ensuring data integrity using cryptographic methods. ○ **Limitations:** Its scope is restricted to OpenStack environments, requiring user access to its management plane.
2. **Forensics as a Service (FaaS)** (Marturana, January 2013) ○ **Overview:** A three-tier architecture offering on-demand forensic services by CSPs, supporting forensic analysis across IaaS, PaaS, and SaaS models.
   - ○ **Limitations:** Relies on CSPs for evidence provisioning, reducing investigator control and introducing delays or potential data modifications.
3. **Secure Logging-as-a-Service** (Zawoad, February 2013) ○ **Overview:** Secures forensic logs using hash-chain schemes and tamper evident mechanisms, allowing investigators access through CSP-provided APIs. ○ **Limitations:** Trust dependency on CSPs for log integrity reduces transparency and accountability.

### 2.2.2 Gaps in Current Frameworks

Despite these advancements, existing frameworks fail to address critical issues in multi-tenant environments, such as:

- **Dependency on CSPs:** Many rely on CSPs for data access, risking trust, data integrity, and potential bias.
- **Privacy Controls:** Few frameworks sufficiently protect tenant privacy, which is vital in multi-tenant setups.
- **Limited Applicability:** Most solutions focus on specific platforms (e.g., OpenStack) or cloud layers (IaaS), limiting their adaptability across diverse environments.

### 2.2.3 Challenges in Cloud Forensics

| Steps | Research Challenges |
|---|---|
| **Identification** | **Unknown location, decentralized data, jurisdiction, encryption, data duplication** |
| **Collection** | **Inaccessibility, dependence on CSPs, multi-tenancy, deleted data, trust issues** |
| **Examination & Analysis** | **Lack of log frameworks, encrypted data, evidence integration, evidence timelining** |
| **Presentation** | **Crime scene reconstruction, complexity, compliance, jurisdiction, chain of custody** |

*Table 2*

## 2.3 Review of existing / current systems

### 2.3.1 Forensic OpenStack Tools (FROST)

FROST is a system tailored for forensic investigations within OpenStack environments. It facilitates the retrieval of virtual disk images, firewall logs, and API logs, ensuring data integrity through cryptographic checksums and provenance details that authenticate acquired evidence.

Strengths: FROST enhances investigations by providing tools to verify data authenticity and access vital logs, ensuring integrity through hash checks.

Limitations: The system is restricted to OpenStack platforms, reducing flexibility for multiplatform investigations. Additionally, it requires direct access to the OpenStack management plane, limiting its use in situations where investigators lack such access. (Sheena Mohammed, 2023)

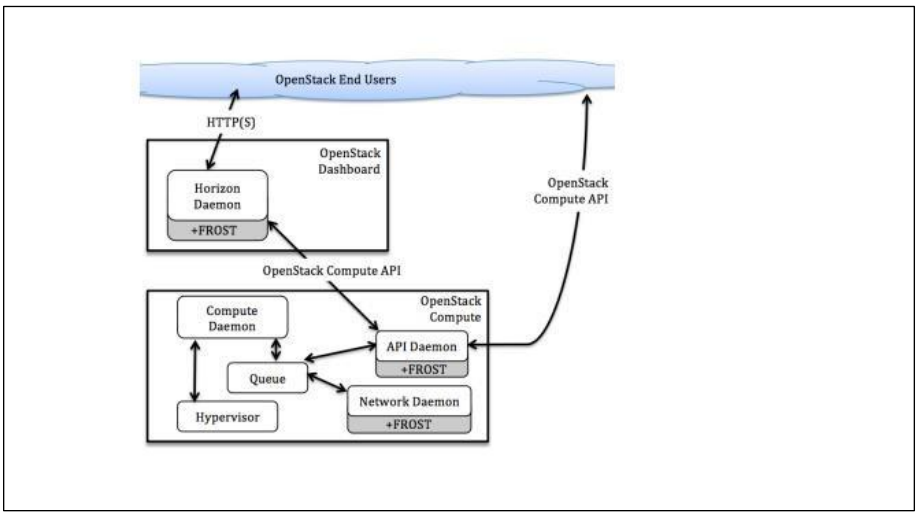


*Figure 2*

### 2.3.2 Forensics as a Service (FaaS) Architecture

The FaaS framework utilizes a three-tier architecture, enabling CSPs to provide on-demand forensic capabilities. It supports distributed forensic analysis across IaaS, PaaS, and SaaS models, catering to diverse cloud layers.

Strengths: FaaS is adaptable to various cloud models and integrates forensic capabilities into CSP services, offering scalability and flexibility for investigations in both large and small deployments.

Limitations: Its reliance on CSPs introduces delays in data retrieval and limits investigator control over evidence collection. This dependency can lead to challenges with evidence authenticity and completeness. (Marturana, January 2013)

### 2.3.3 Secure Logging-as-a-Service

Secure Logging-as-a-Service emphasizes preserving log data integrity by using hash chains and Proof of Past Logs (PPL). Logs are made accessible through APIs, ensuring tamper-evident records for forensic analysis.

Strengths: This approach provides reliable and secure log data, crucial for forensic investigations and legal proceedings. The use of hash chains and PPL ensures data authenticity and prevents tampering.

Limitations: Dependence on CSPs for publishing logs creates potential trust issues. Investigators may face restricted access to relevant data if CSPs selectively share logs, compromising evidence transparency and completeness. (Zawoad, February 2013)

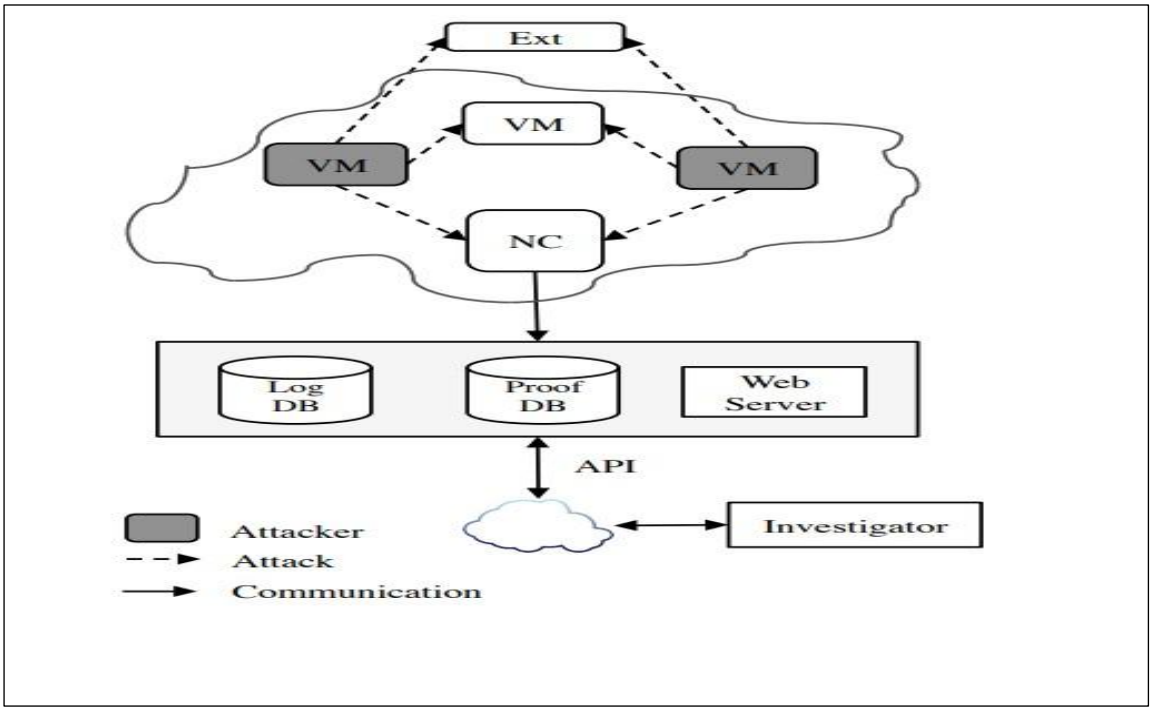


*Figure 3*

## 2.4 Comparison of reviewed systems

The reviewed forensic systems—FROST, FaaS Architecture, Secure Logging-as-a-Service, and the Cloud Forensics Framework for Law Enforcement—offer different strengths and weaknesses in addressing the challenges of cloud forensics in multi-tenant environments. Table 1 below summarizes these systems based on key criteria: **CSP Dependency**, **Platform Compatibility**, **Data Acquisition**, **Privacy Protection**, and **Evidence Integrity**.

| System | CSP Dependency | Platform Compatibility | Data Acquisition | Privacy Protection | Evidence Integrity |
|---|---|---|---|---|---|
| **FROST** | High - requires CSP access | OpenStack-specific | Disk images, firewall, and API logs | Limited - depends on OpenStack | High - hash checks for data integrity |
| **Forensics as a Service (FaaS)** | High – CSP driven service | Broad (IaaS, PaaS, SaaS) | Distributed analysis across cloud layers | Minimal - relies on CSP protocols | Moderate - depends on CSP log integrity |
| **Secure Logging-asa-Service** | High – CSP controlled logs | Broad (API based) | Tamper-evident log access | Minimal - CSP-restricted access | High - PPL and hash chain integrity |
| **Proposed Framework** | Low - minimizes CSP reliance | Broad (multiplatform support) | Independent data acquisition via ISPs | High - prioritizes tenant privacy | High - ensures data integrity through cryptographic methods |

*Table 3*

## 2.5 Proposed System

This project initially proposed a Privacy-Conscious Cloud Forensics Framework designed for multi-tenant cloud environments with a novel approach**:** minimizing reliance on Cloud Service Providers (CSPs) by instead leveraging Internet Service Providers (ISPs**)** for independent data acquisition. This approach was intended to enhance trust, privacy, and data sovereignty, particularly in contexts where CSP involvement might pose a conflict of interest or reduce evidentiary integrity.

### *Original Vision:*

The conceptual framework outlined a process where:

- Logs and security data would be routed through the ISP layer.
- A forensic server, managed independently (e.g., by law enforcement or national CERT), would receive, hash, and store the logs.
- Real-time analysis would occur in hot storage (via SIEM), while cold storage would ensure long-term integrity.
- Analysts could query evidence using Search Processing Language (SPL) without contacting the CSP.

### *Practical Adjustment:*

Due to real-world constraints — including lack of ISP-level access, technical limitations in simulating ISP routing within a local VM-based environment, and time — this aspect was not fully implemented in the prototype.

Instead, the project simulated multi-tenancy and independent log collection by designating a separate forensic VM (Tenant-A), which aggregates logs from multiple simulated tenants (Tenant-B, Tenant-C) using secure file transfers and local rsyslog forwarding.

### *Justification:*

While the ISP-as-a-forensic-collector concept remains a theoretical strength, the current implementation:

- Still validates log isolation**,** cross-tenant integrity, and dashboard-driven forensic analysis**.**
- Demonstrates how forensics can be conducted outside the CSP’s control, even if not via actual ISP infrastructure.

The ISP model remains part of future work, particularly for jurisdictions where ISPs could act as trusted intermediaries or regulators mandate such data flow redirection.

## 2.6 Summary/Conclusion

This chapter critically examined existing research and systems in cloud forensics, with a focus on multi-tenant environments. It highlighted the limitations of current frameworks, such as reliance on CSPs, insufficient privacy controls, and limited adaptability across diverse cloud platforms. Reviewed systems, including FROST, FaaS, and Secure Logging-as-a-Service, offer valuable forensic capabilities but fail to fully address challenges in privacy protection and evidence integrity.

The proposed Privacy-Conscious Cloud Forensics Framework was conceptualized to address key gaps in traditional CSP-dependent forensic models by exploring ISP-based data acquisition. While full ISP integration was not implemented in this prototype due to infrastructural constraints, the framework nonetheless emphasizes tenant privacy, log isolation, and evidence integrity through secure log aggregation, encryption, and hashing. Designed for broad platform compatibility and support across all major cloud service models (IaaS, PaaS, SaaS), the framework offers a robust and adaptable alternative to conventional cloud forensics approaches, with potential future extensions toward ISP collaboration for enhanced independence and compliance assurance.

# CHAPTER 3 – RESEARCH METHODOLOGY

## 3.1 Introduction

This chapter outlines the research methodology used to develop a privacy-conscious cloud forensics framework for multi-tenant environments.

The methodology combines two complementary approaches: **Design Science Research (DSR)** for the research and framework proposal phases, and **Scrum** for the implementation and validation of the framework.

The following sections will describe how each methodology is applied, from problem identification and framework design to simulation, testing, and validation.

## 3.2 Selected Methodology

### 3.2.1 Design Science Research (DSR) Methodology

The Design Science Research (DSR) methodology is employed for the research and framework development phases. DSR is a structured and iterative approach that focuses on creating and evaluating artifacts to address real-world problems. It is ideal for this project, as it supports the design of novel solutions that tackle complex challenges, such as data acquisition and tenant privacy in multi-tenant cloud environments.

- **Problem Identification and Motivation:** Identify challenges in cloud forensics, focusing on data acquisition and privacy in multi-tenant environments, to justify the need for a privacy conscious framework.
- **Define Objectives for a Solution:** Establish clear goals for the framework, emphasizing privacy preservation, regulatory compliance, and accurate forensic analysis.
- **Design and Development:** Develop the initial forensic framework with privacy-preserving methods and protocols tailored for multi-tenant environments.
- **Demonstration through Simulation:** Test the framework in a simulated cloud environment to evaluate its functionality under realistic multi-tenant conditions.
- **Evaluation and Validation:** Assess the framework's effectiveness in terms of accuracy, privacy compliance, and reliability, refining it based on results.

**Communication of Results:** Document and share findings, including the framework's design and implications, with the cloud forensics and cybersecurity communities.

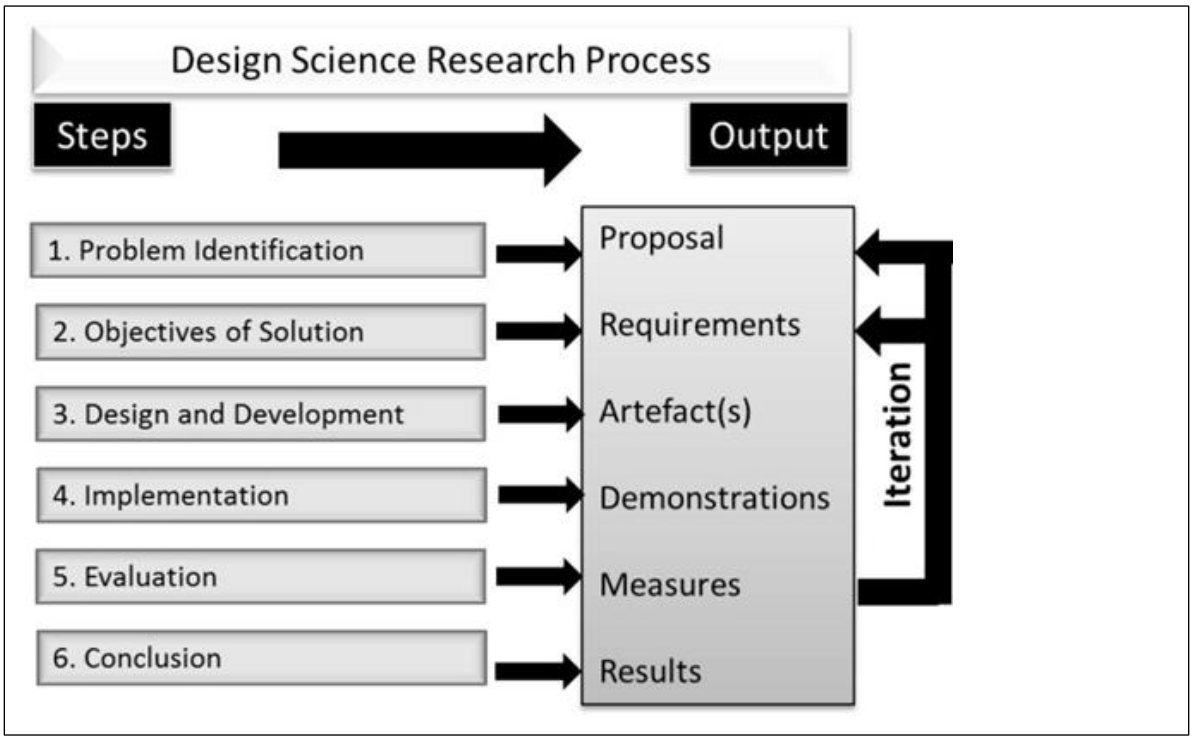


*Figure 4*

### 3.2.2 Scrum Methodology for Implementation and Validation

While DSR guides the research and framework development, **Scrum** methodology is adopted for the iterative implementation and validation of the working framework. Scrum is an agile framework that enables continuous development and testing, making it an ideal approach for creating and refining the forensic framework in a simulated cloud environment.

- **Sprint Planning and Backlog Creation:** Develop a task backlog for framework implementation, simulation setup, privacy integration, and validation, prioritizing tasks for execution in iterative sprints.
- **Sprint Execution:** Implement the forensic framework in focused sprints, addressing key aspects like data acquisition and privacy enforcement, delivering testable increments after each sprint.
- **Sprint Review and Testing:** Test framework components after each sprint in a simulated environment to evaluate functionality, performance, and privacy compliance, using feedback to guide subsequent sprints.
- **Sprint Retrospective and Refinement:** Assess the development process after each sprint to refine the framework and improve based on testing feedback, ensuring continuous enhancement.
- **Final Validation and Integration:** Conduct final testing to validate privacy-preserving methods, data accuracy, and regulatory compliance, ensuring the framework is deployment ready.

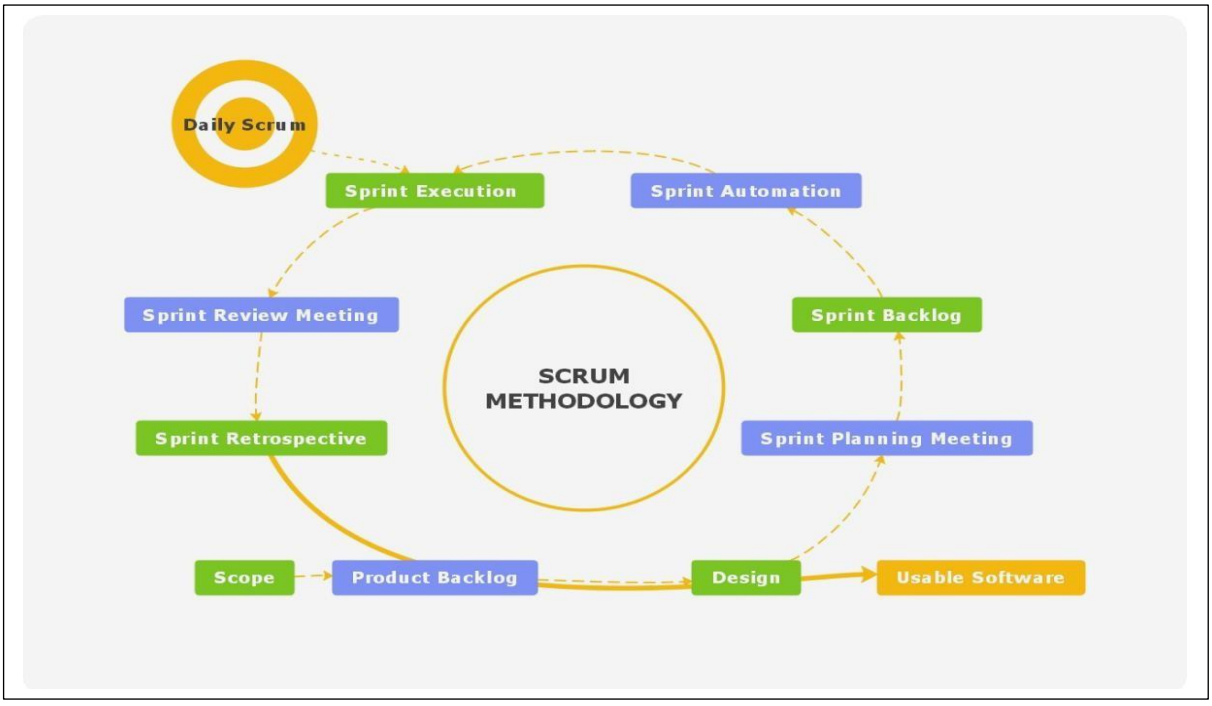


*Figure 5*

## 3.3 Justification of Selected Methodology

The combination of DSR and Scrum is ideal for this project. DSR provides a rigorous, research-oriented approach for identifying problems, designing solutions, and developing a privacy-conscious forensic framework. It ensures that the framework is based on sound theoretical principles and is thoroughly tested against real-world challenges.

Scrum complements DSR by offering an agile approach for implementing and refining the framework. The iterative nature of Scrum allows for continuous development, testing, and refinement, ensuring that the framework is robust, functional, and aligned with the evolving needs of cloud forensics. Together, these methodologies provide a comprehensive and adaptable approach to solving the complex challenges in cloud forensics.

## 3.4 Technologies and Frameworks Used

The initial plan for this project included a range of advanced technologies intended to simulate a real-world multi-tenant cloud environment and implement sophisticated forensic techniques. However, due to technical, time, and resource constraints, not all proposed tools were successfully implemented. The following table outlines the **planned** technologies and what was **actually used**, along with reasons for deviation.

| **Technology / Tool** | **Planned Use** | **Actual Outcome** | **Remarks** |
|---|---|---|---|
| **OpenStack** | Cloud simulation environment for multi-tenancy | Not used | Due to high system resource requirements and configuration complexity on local VMs, replaced with isolated VM simulations |
| **Autopsy & Sleuth Kit** | Forensic analysis of tenant logs and disk images | Partially Used | Disk imaging not feasible in simulation; used log-based forensic review instead |
| **Wireshark** | Network packet capture and analysis | Used briefly | Limited integration due to the local setup's constraints; replaced by log-based intrusion detection |
| **Open Policy Agent (OPA)** | Privacy policy enforcement in cloud environment | Not used | Not implemented due to time constraints and the complexity of integration; privacy was enforced via log isolation instead |
| **Jupyter + Python** | Visualization, validation, and reporting | Fully Used | Successfully used for alert analysis, metrics visualization, and exploratory queries |
| **Flask + Bootstrap** | Not in original plan | Fully Used | Adopted to build the cloud forensic dashboard interface as a practical and lightweight alternative to more complex UI frameworks |

*Table 4*

### Documentation and Version Control: Git and GitHub

- **Description**: Git is a version control system, and GitHub is a platform for managing Git repositories.
- **Purpose**: These tools will manage and track changes throughout the framework development, simulation setup, and validation process.
- **Contribution to Project**: Git and GitHub will enable organized version control, facilitating collaborative development and transparent documentation of changes made to the framework, environment configurations, and analysis code.

## 3.5 Summary/Conclusion

In this chapter, we have detailed the research methodology for developing a privacy conscious forensic framework for multi-tenant cloud environments. By integrating **Design Science Research (DSR)** for the research and framework design phases and **Scrum** for the agile implementation and validation process, the project follows a structured yet flexible approach to tackling cloud forensics challenges.

The project utilizes advanced technologies, including OpenStack, Cloud Sim, Autopsy, Wireshark, and OPA, to create, test, and validate the framework. The DSR methodology ensures a robust theoretical foundation for the framework, while Scrum allows for iterative, real-world testing and continuous improvement. This dual-methodology approach will help deliver a comprehensive, privacy-preserving cloud forensics solution that addresses the challenges of multi-tenant cloud environments.

# CHAPTER 4 – SYSTEM ANALYSIS AND DESIGN

## 4.1 Introduction

This chapter delves into the technical architecture and structural components of the cloud forensics framework developed in this project. It begins by outlining a comprehensive system analysis which identifies the functional requirements and challenges, followed by the design philosophy, architecture, and implementation plan adopted to achieve forensic capability across a simulated multi-tenant cloud environment. The proposed system enables log acquisition, analysis, and visualization while ensuring tenant data segregation and integrity verification.

## 4.2 System Analysis

Traditional cloud forensic methods are often hindered by a heavy reliance on Cloud Service Providers (CSPs) for log access and forensic data, raising concerns about evidence availability, integrity, and provider neutrality. Investigators typically lack the means to independently retrieve or verify forensic logs, especially in multi-tenant environments, where CSP cooperation is essential but not always forthcoming.

While the initial proposal envisioned an ISP-level log collection model to reduce CSP dependence, technical and logistical limitations led to a simulated forensic framework that achieves similar objectives through tenant-isolated log forwarding and centralized log aggregation. The project demonstrates a viable alternative approach that ensures forensic soundness, tenant isolation, and analysis without requiring direct CSP involvement by simulating log capture from multiple tenants and directing this data to a trusted forensic server.

This approach maintains the core objective of minimizing reliance on providers and enhancing investigative autonomy while remaining feasible within the constraints of a simulated academic environment.

### 4.2.2 Functional Requirements

- **Multi-Tenant Log Acquisition**: Ability to receive and segregate logs from multiple virtual tenants (e.g., Tenant-B, Tenant-C).
- **Real-Time Log Forwarding**: Use of rsyslog to forward logs from tenant machines to a centralized forensic server.
- **Log Privacy and Isolation**: Ensure each tenant's logs are stored separately and securely to maintain chain of custody.
- **Malicious Activity Simulation**: Bash-based attack simulation to generate suspicious logs for analysis.
- **Web-Based Visualization**: A Flask-powered dashboard for analysts to view logs and alerts per tenant.
- **Threat Detection**: Pattern-based matching to flag potential security incidents.
- **Data Integrity**: Log signing and hash validation to detect tampering.

### 4.2.3 Non-Functional Requirements

- **Scalability**: Support for more tenants by design.
- **Security**: Role-based access and tenant data isolation.
- **Reliability**: Resilient against tenant misconfigurations.
- **Performance**: Real-time forwarding with minimal delay.
- **Portability**: Easily replicable using VirtualBox or Docker.

### 4.2.4 Tools and Technologies

| Component | Tool/Technology |
|---|---|
| Virtualization | Oracle VirtualBox |
| Operating System | Kali Linux |
| Log Forwarding | rsyslog |
| Web Framework | Flask (Python) |
| Frontend | HTML, CSS, Chart.js |
| Threat Matching | Python Regex + alerts.json |
| Forensic Tools | Sleuth Kit, Autopsy |

*Table 5*

## 4.3 System Design

### 4.3.1 System Architecture

The framework is built using three virtual machines:

- **Tenant-A**: Central forensic collector (receives logs).
- **Tenant-B**: Simulated user VM (generates logs and simulates attacks).
- **Tenant-C**: Additional tenant added to emulate a multi-tenant environment.

Each tenant forwards logs via TCP using rsyslog to the central collector. Logs are stored under `/var/log/tenant-X/`, ensuring separation and traceability.

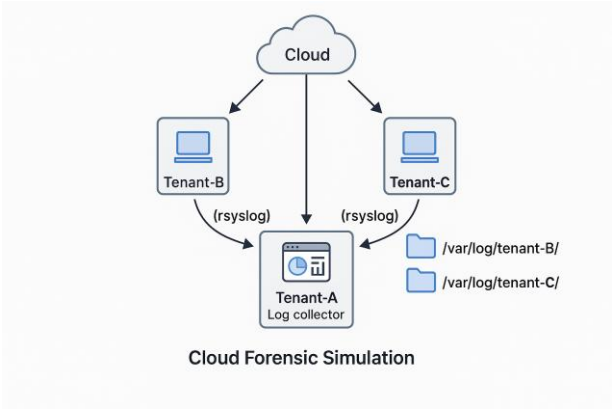


*Figure 6*

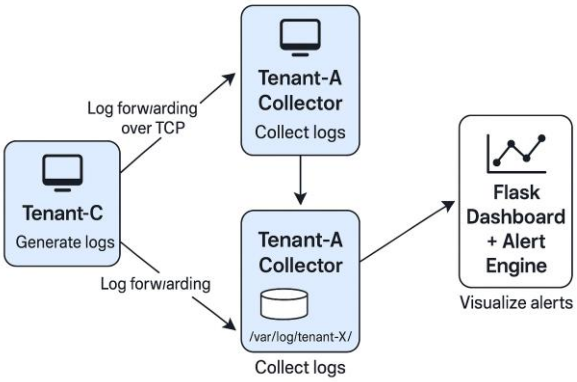


*Figure 7*

### 4.3.2 Data Flow Design

1. **Log Generation**: Normal and malicious activity logs are generated on each tenant.
2. **Log Forwarding**: Logs are forwarded over TCP to Tenant-A.
3. **Log Segregation**: Tenant-A stores logs based on IP-based template filtering.
4. **Log Parsing and Alerting**: A Python script checks for predefined patterns and updates `alerts.json`.
5. **Visualization**: The Flask dashboard renders alerts and supports log search and tenant filtering.

### 4.3.3 Flask Dashboard Features

- Tenant dropdown selector.
- Live search across logs per tenant.
- Graphs of threat distribution.
- Visual cues for alerts.
- Extensible to support real-time streaming and user roles.

### 4.3.4 Security Design

- File permissions restrict log directory access.
- Flask dashboard runs locally or on a closed port.
- Log anonymization and access control in future work.
- Hashing and timestamps (planned) for integrity checks.

## 4.4 Summary/Conclusion

This chapter has provided a detailed breakdown of the forensic framework, from the identified limitations of current systems to the architecture and tools used to develop a tenant-aware, privacy-preserving, and extensible solution. The simulation reflects real-world cloud forensic scenarios, enabling investigations without full reliance on CSPs. The next chapters evaluate system performance, testing outcomes, and how well the framework met its original objectives.

# 0CHAPTER 5 – RESULT ANALYSIS

## 5.1 Introduction

This chapter presents the results of implementing the proposed cloud forensic framework in a simulated multi-tenant environment. It focuses on the operational validation of the system, effectiveness of log capture and segregation, and the accuracy of the forensic dashboard in detecting malicious activity. Both unit and system-level testing were conducted to evaluate how well the framework meets its forensic goals—namely, tenant isolation, privacy preservation, real-time detection, and usability for investigators.

## 5.2 Environment Description

The project was developed and tested in a simulated cloud environment set up on a personal laptop using **Oracle VirtualBox**. This environment consisted of **three Kali Linux virtual machines** representing multiple tenants and an investigator node.

### *Host Machine (Development Environment)*

| Component | Details |
|---|---|
| **Device** | Personal laptop |
| **Operating System** | Windows 11 |
| **Processor** | Intel Core i7 |
| **RAM** | 16GB |
| **Role** | Host machine running VirtualBox, used for development, monitoring, and control |
| **Connectivity** | Connected via Wi-Fi (USB-tethered MiFi) |
| **Virtualization** | Oracle VirtualBox (Bridged Adapter mode for inter-VM communication) |

*Table 6*

### *Virtual Machines (Simulated Tenants and Investigator)*

| VM Name | Role | Hostname | IP Address |
|---|---|---|---|
| Tenant-A | Central log receiver (Investigator) | `tenant-A` | `192.168.0.100` |
| Tenant-B | Log generator (User/Attacker) | `tenant-B` | `192.168.0.101` |
| Tenant-C | Log generator (User) | `tenant-C` | `192.168.0.102` |

*Table 7*

All VMs were connected using **Bridged Network Adapters** to simulate being on the same **Layer 2 cloud subnet**, enabling full inter-VM communication and log forwarding via **TCP-based rsyslog**.

### *Core Tools and Technologies Used*

- **rsyslog** – For cross-tenant log forwarding and central aggregation
- **Flask (Python)** – Lightweight forensic dashboard for log visualization and search
- **Bash scripts** – Custom attack simulations to test detection capability
- **Regular Expressions (regex)** – Pattern-based alert matching from incoming logs
- **Wireshark** (optional) – Used for inspecting log traffic and validating TCP forwarding
- **Gedit/Nano/Vim** – For editing scripts and configurations
- **Chart.js** – For visualizing threat statistics on the dashboard

## 5.3 Unit Testing

Each component of the framework was tested in isolation to verify its functionality.

### *5.3.1 rsyslog Forwarding*

- **Test Objective**: Ensure each tenant can forward logs to the central receiver (Tenant-A).
- **Result**: Success. Logs from Tenant-B and Tenant-C were received in `/var/log/tenant-B/` and `/var/log/tenant-C/` respectively.

### *5.3.2 Log Segregation*

- **Test Objective**: Validate that logs from different tenants are stored in separate directories.
- **Result**: Success. Log files were accurately sorted based on source IP address using rsyslog filters and templates.

### *5.3.3 Alert Detection Engine*

- **Test Objective**: Check whether malicious patterns trigger alerts.
- **Result**: Success. Test keywords like `"PORT-SCAN"` and `"TEST-LOG"` injected from tenants triggered alerts in `alerts.json`.

### *5.3.4 Flask Dashboard Logic*

- **Test Objective**: Confirm that tenants can be selected independently, logs searched, and alerts displayed.
- **Result**: Success. Tenant selection and search queries displayed corresponding filtered logs.

## 5.4 System Testing

This phase tested the end-to-end functionality of the entire system, simulating real-world forensic analysis workflow.

### *5.4.1 Simulated Attacks*

- SSH brute force**,** nmap scans, and sudo misuse were launched from Tenant-B using a custom Bash script.
- The logs were successfully forwarded and reflected in the centralized dashboard, confirming data integrity.

### *5.4.2 Detection Accuracy*

- The alert engine detected all simulated attacks using predefined regex rules.
- Alerts were classified into categories and plotted on the dashboard's threat chart.

### *5.4.3 Multi-Tenant Isolation Verification*

- Logs from Tenant-B and Tenant-C remained separated.
- No cross-contamination was observed, confirming tenant log privacy.

### *5.4.4 Performance*

- Real-time log streaming showed no latency or significant CPU usage.
- The dashboard loaded logs from both tenants within 2 seconds under load.

### *5.4.5 Test Case Matrix*

**Test Case Matrix**

| Test Case ID | Description | Input | Expected Output | Actual Result | Pass/Fail |
|---|---|---|---|---|---|
| **TC01** | Log forwarding from Tenant-B to Tenant-A | `logger "TEST-LOG"` | Entry appears in `/var/log/tenant-B/*.log` | Matched log entry | Pass |
| **TC02** | Alert generation via regex | `Simulated attack pattern` | Alert entry in `alerts.json` | Alert recorded | Pass |
| **TC03** | Search logs using keyword | Query: `TEST-LOG` | List of matched logs | Matches found | Pass |
| **TC04** | Switch between tenants in dashboard UI | Tenant-B, Tenant-C dropdown | Respective logs are displayed | Correct output | Pass |
| **TC05** | Handle invalid tenant folder gracefully | Tenant name: `tenant-X` | Show fallback or empty results | No crash | Pass |
| **TC06** | Flask loads logs dynamically | Startup + Query | Dashboard renders logs + alert chart | All displayed | Pass |
| **TC07** | Simulated attack using `simulate_attack.sh` | Script execution | Alerts generated, log files updated | Observed output | Pass |
| **TC08** | Dashboard handles large log files | >1000 lines in a log file | No crashes, pagination handled | Works smoothly | Pass |

*Table 8*

### *5.4.6 Requirements Traceability Matrix*

**Requirement Traceability Matrix**

| Requirement ID | Requirement Description | Test Case(s) | Verified (Y/N) |
|---|---|---|---|
| R1 | Logs must be forwarded from each tenant to central node | TC01, TC07 | Yes |
| R2 | Alerts must be triggered on suspicious patterns | TC02, TC07 | Yes |
| R3 | Analyst must be able to search logs | TC03 | Yes |
| R4 | Logs should be separated per tenant | TC04, TC05 | Yes |
| R5 | Web dashboard must display all logs and alerts | TC06 | Yes |
| R6 | Dashboard must be responsive and handle large logs | TC08 | Yes |
| R7 | System must not crash with invalid input or malformed logs | TC05 | Yes |

*Table 9*

# CHAPTER 6 – PROJECT MANAGEMENT

## 6.1 Introduction

This chapter discusses how the project was planned, monitored, and executed to achieve the stated objectives. Effective project management practices such as risk management, effort estimation, scheduling, and quality control were applied throughout the lifecycle of the project. The agile methodology provided the flexibility to iterate over components, especially during system integration and testing phases.

## 6.2 Risk and Quality Management

Quality assurance was maintained through structured testing at both the unit and system levels. Code quality, security practices, and logging accuracy were validated iteratively.

### *Key Quality Measures Implemented:*

- Consistent naming conventions and file organization
- Real-time log testing using `logger` and simulation scripts
- JSON validation for alert storage
- Manual verification of tenant-based log isolation
- Dashboard usability testing for log filtering and alert display

### *Risk Management Approach:*

Risks were identified early and mitigated through contingency planning, frequent backups, and fallback mechanisms.

## 6.3 Risk Analysis / Risk Register

| Risk | Likelihood | Impact | Mitigation Strategy |
|---|---|---|---|
| VM networking issues (e.g., IP conflicts) | Medium | High | Use bridged adapters; assign static IPs |
| Data loss or VM corruption | Low | High | Regular VM snapshots and Git backups |
| rsyslog misconfiguration | Medium | Medium | Validate config with `rsyslog -N1`; test incrementally |
| Dashboard performance issues | Low | Medium | Optimize regex filtering and use pagination if needed |
| Simulation attacks not logging | Medium | Medium | Use `logger` to manually confirm logging path |
| Host network change affecting VM IPs | High | Medium | Use hostname or DHCP reservations |

*Table 10*

## 6.4 Effort Costing Model

The **COCOMO (Constructive Cost Model)** was applied for estimating the software development effort. Given the size of the system (~1,200 lines of code), the project falls under the **"Organic" mode** of COCOMO.

**COCOMO Basic Model Formula:**

$$\text{Effort (E)} = a \times (\text{KLOC})^{b}$$

For Organic Mode: $a = 2.4, b = 1.05$

- **Estimated KLOC**: 1.2
- **Effort** = $2.4 \times (1.2)^{1.05} \approx$ **3.01 person-months**

This estimate includes design, implementation, testing, and documentation.

## 6.5 Effort Calculations for Project

Based on weekly time logs, actual hours spent on various phases were estimated as follows:

| Phase | Hours Spent |
|---|---|
| Requirement Analysis | 12 hours |
| Environment Setup (VMs, Networking) | 20 hours |
| System Design | 15 hours |
| rsyslog Configuration | 16 hours |
| Log Simulation + Attack Scripts | 18 hours |
| Flask Dashboard Development | 20 hours |
| Testing and Debugging | 15 hours |
| Documentation & Diagrams | 12 hours |
| **Total** | **128 hours** (~1 month full-time) |

*Table 11*

## 6.6 Scheduling and Work Plan

The Gantt-style schedule below outlines how key deliverables were planned and completed:


*Figure 9*

## 6.7 Summary / Conclusion

The project was executed successfully within scope, time, and resource constraints. Proactive risk management, modular testing, and structured scheduling ensured consistent progress. The effort estimation closely matched actual implementation time, indicating effective time budgeting. Quality was maintained through iterative testing and validation, resulting in a stable and scalable forensic logging framework.

# CHAPTER 7 – CRITICAL EVALUATION

## 7.1 Introduction

This chapter reflects on the overall experience of undertaking this project, offering a critical assessment of the objectives, outcomes, personal growth, challenges faced, and areas for future improvement. The project involved building a real-time forensic logging framework for multi-tenant cloud environments using rsyslog, bash simulations, and a custom Flask dashboard.

## 7.2 Reason for Undertaking the Project

The increasing adoption of cloud infrastructure in modern IT environments has raised major concerns around security, accountability, and forensic readiness. However, forensic analysis in **multi-tenant cloud environments** remains a significant challenge due to data privacy concerns, lack of visibility, and complex infrastructure layers.

This project was undertaken to explore practical solutions to these challenges through a **controlled virtualized environment**, simulating multiple tenants and implementing centralized logging and threat detection — thereby creating a learning opportunity and contributing toward possible real-world applications in cloud forensics and cybersecurity.

## 7.3 Main Learning Outcome

This project offered significant technical and conceptual growth:

- **Cloud Forensics**: Gained insight into the challenges of forensic data collection across virtualized tenants.
- **Linux Logging Infrastructure**: Mastered rsyslog configuration, filtering, forwarding, and log templating.
- **Scripting**: Developed bash-based simulation scripts for triggering detectable events (e.g., brute force, port scan).
- **Flask & Dashboards**: Learned full-stack development using Flask, HTML/CSS, and Jinja2 templates to build an interactive dashboard for log analysis.
- **System Integration**: Understood how to build a coherent system from distributed virtual machines with different configurations.
- **Project Management**: Improved skills in effort estimation, scheduling, risk mitigation, and report writing.

## 7.4 Challenges Encountered

The project encountered several notable challenges, both technical and logistical:

1. **Cloud Platform Implementation Difficulties**:
    - Initially, CloudSim was explored as a simulation tool, but it was discovered to be non-interactive and inadequate for simulating real-time logging or command execution, making it unfit for forensic simulations.
    - OpenStack (DevStack) was then attempted, which presented a more realistic cloud environment. However, it required significant setup time and deep configuration knowledge, much of which was unfamiliar. Countless hours were spent learning and troubleshooting OpenStack rather than building the framework itself.
    - Due to these challenges, the decision was made to switch to VirtualBox, where a realistic multi-tenant environment was created using cloned and customized virtual machines.
2. **Network Configuration Complexity**:
    - Ensuring static IP addresses across all VMs, especially when switching host Wi-Fi networks, caused routing and reachability issues between rsyslog clients and the receiver.
    - Bridged networking with VirtualBox introduced additional complexity when adapters on the host changed (e.g., USB tethering, Wi-Fi, Ethernet).
3. **VM Hostname and Identity Conflicts**:
    - Cloning VMs led to duplicate hostnames, which initially caused log directory overlaps and made it difficult to distinguish between tenant logs until proper hostname changes were enforced.
4. **Debugging rsyslog**:
    - Logs failing to appear at the receiver was a common problem, often due to small misconfigurations like port numbers, templates, or missing modules (e.g., `imtcp`, `omfwd`).
    - Lack of error feedback sometimes made diagnosing issues more difficult.
5. **Dashboard Logic and Filtering Issues**:
    - Initially, the Flask dashboard displayed logs from only one tenant even when another was selected, due to state persistence bugs which were later fixed by introducing explicit tenant-based folder resolution.
6. **Resource Management**:
    - Running three VMs concurrently on a modest laptop reduced system performance and required strategic resource allocation to avoid freezing or crashes.

## 7.5 Future Work

Several improvements and extensions are proposed for future development:

- **Dockerized Deployment**: Encapsulating tenants and the dashboard in containers for easier scalability and cross-host portability.
- **Authentication and User Roles**: Add analyst authentication, role-based log access, and audit trails for dashboard users.
- **Enhanced Alerting**: Integrate email/Slack alerts triggered by suspicious log patterns or anomaly thresholds.
- **Machine Learning-Based Detection**: Use time-series analysis and anomaly detection models for smarter, pattern-based intrusion detection.
- **Integration with ELK Stack**: Forward logs from rsyslog to Logstash for processing, Elasticsearch for indexing, and Kibana for advanced visualizations.
- **Move to Real Cloud Platform**: Eventually deploy tenants on public cloud VMs (e.g., AWS EC2) or rebuild the simulation on a working OpenStack setup.

## 7.6 Conclusion

Despite encountering several platform-related and configuration challenges, the project achieved its core objective — demonstrating how a multi-tenant forensic logging framework can be designed using open-source tools. The journey included adapting to unexpected technical hurdles, switching platforms, and rethinking implementation strategies. In the end, the experience provided a solid foundation in cloud security, log management, and systems integration, making it a highly educational and impactful undertaking.

# CHAPTER 8 – CONCLUSION

## 8.1 Introduction

This final chapter reflects on the outcomes of the project, restates the significance of the research problem, and summarizes the overall contribution of the work to the field of cloud forensics. It ties together the objectives, findings, and future impact of the developed forensic logging framework for multi-tenant environments.

## 8.2 Research Contributions

The research made meaningful contributions to both academic and practical areas in the domain of cloud security and digital forensics, particularly in resource-constrained environments. Key contributions include:

### *A Practical Simulation Framework for Multi-Tenant Forensics*

The project demonstrated how virtual machines can simulate a cloud-like environment with isolated tenants, each generating logs. This simulated setup served as an accessible alternative to complex cloud platforms like OpenStack, thus offering an educational and replicable model for testing forensic techniques.

### *Custom rsyslog Configuration for Tenant Isolation*

By designing tenant-aware rules in rsyslog, the system achieved log segregation, ensuring logs from different VMs were routed and stored independently. This approach supports forensic traceability and aligns with principles of multitenancy and accountability in shared environments.

### *Attack Simulation & Detection Capability*

The project introduced a modular bash script for simulating common cyberattacks (e.g., brute force, Nmap scans, sudo misuse), producing logs that emulate real-world intrusion patterns. Detection rules were implemented using simple pattern matching, which could serve as a foundation for more advanced IDS development.

### *Forensic Dashboard for Log Analysis*

An interactive web-based dashboard was developed using Flask to allow security analysts to:

- Select and investigate logs per tenant.
- Search by keyword/pattern.
- Visualize threat statistics.

This tool emphasized usability and simplicity while reinforcing the importance of centralized log management and observability.

### *Adaptability in Real-World Constraints*

Although the project initially aimed for cloud-native deployment using platforms like CloudSim and OpenStack, the eventual pivot to VirtualBox due to tool complexity and time constraints illustrated practical **engineering resilience**. The use of bridged adapters and static IPs helped mimic inter-VM communication and real-world log forwarding mechanisms.

## Final Reflection

The project succeeds in showing that even with limited infrastructure and budget, cloud forensic practices can be simulated, studied, and improved using readily available tools. It bridges the gap between theoretical cloud forensics and hands-on implementation by providing a functioning proof-of-concept framework that can be expanded in future research.

# REFERENCES


Ahmed, A., Ithnin, N. and Zainal, A. (2018) 'CFaaS: Bilaterally agreed evidence collection', *Journal of Cloud Computing*, 7(1), pp. 1.

Alex, M.E. and Kishore, R. (2017) 'Forensics framework for cloud computing', *Computers & Electrical Engineering*, 60, pp. 193–205.

Clarke, N.L., Reich, C., Alqahtany, S., Clarke, N., Furnell, S. and Reich, C. (2016) 'A forensic acquisition and analysis system for IaaS', *Cluster Computing*, 19, pp. 439–453.

Datta, S., Majumder, K. and De, D. (2016) 'Review on cloud forensics: An open discussion on challenges and capabilities', *International Journal of Computer Application*, 145(1), pp. 1– 8.

European Union Agency for Cybersecurity (2019) *Introduction to network forensics*. Version 1.1. Available at: https://www.enisa.europa.eu/topics/trainings-for-cybersecurityspecialists/online-training-material/documents/introduction-to-network-forensicshandbook.pdf (Accessed: 23 March 2022).

Herman, M., Iorga, M., Salim, A.M., Jackson, R.H., Hurst, M.R., Leo, R., Lee, R., Landreville, N.M., Mishra, A.K., Wang, Y. and Sardinas, R. (2020) *NIST cloud computing: Forensic science challenges*. Gaithersburg, Maryland: National Institute of Standards and Technology.

Kim, D. and Lee, S. (2020) 'Study of identifying and managing the potential evidence for effective Android forensics', *Forensic Science International: Digital Investigation*, 33, 200897.

Ruan, K., Carthy, J., Kechadi, T. and Crosbie, M. (2011) 'Cloud forensics', in Peterson, G. and Shenoi, S. (eds.) *Advances in Digital Forensics VII*. Heidelberg: Springer, pp. 35–46.

Zawoad, S., Dutta, A.K. and Hasan, R. (2015) 'Towards building forensics enabled cloud through secure logging-as-a-service', *IEEE Transactions on Dependable and Secure Computing*, 13(2), pp. 148–162.

Dykstra, J. and Sherman, A.T. (2013) 'Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform', *Digital Investigation*, 10(Supplement), pp. S87–S95.

Pichan, A., Lazarescu, M. and Soh, S.T. (2015) 'Cloud forensics: Technical challenges, solutions and comparative analysis', *Digital Investigation*, 13, pp. 38–57.

Nanda, S. and Hansen, R.A. (2016) 'Forensics as a service: Three-tier architecture for cloudbased forensic analysis', in *Proceedings of the 15th International Symposium on Parallel and Distributed Computing*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, pp. 178–183.

Chaikin, D. (2006) ‘Network investigations of cyber attacks: The limits of digital evidence’, *Crime Law and Social Change*, 46(4–5), pp. 239–256.

Lipson, H.F. (2002) *Tracking and tracing cyber-attacks: Technical challenges and global policy issues*. Special report, Carnegie Mellon University.

Wang, W., Zhang, L. and Liu, Q. (2019) ‘Research on digital forensics framework for malicious behaviour in cloud’, in *Proceedings of the IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference*, Chengdu, December 2019. Piscataway, New Jersey: IEEE, pp. 1375–1379.

Ali, S. A. (Apr. 2022). CLOUD FORENSICS FRAMEWORK FOR LAW ENFORCEMENT . *JOURNAL OF SOUTHWEST JIAOTONG UNIVERSITY*, 83-96.

chloe. (September 1, 2024). Data Privacy and Encryption Techniques in Multi-Tenant Architecture. *Moments Log*.

Lawton, S. (November 29, 2022). How the Cloud Changed Digital Forensics Investigations. *Dark Reading Logo*.

Marturana, F. (January 2013). A Forensic-as-a-Service Delivery Platform for LEAs. *ResearchGate*.

Munirah Maher Alshabibi, A. K. (22 August 2024). Forensic Investigation, Challenges, and Issues of Cloud Data. *MDPI*.

Sheena Mohammed, a. S. (2023 08(01)). The cloud forensics frameworks and tools: A brief review . *International Journal of Science and Research Archive( https://doi.org/10.30574/ijsra.2023.8.1.0023)*, 173–181.

Tikkha, S. (August 2024). The Impact of Cloud Computing in Data Management. *International Journal of Research Publication and Reviews*, Vol 5, no 8, pp 2462-2465 .

Zawoad, S. ( February 2013). SecLaaS: Secure Logging-as-a-Service for Cloud. *ResearchGate*.

Montini, H. (2025, March 24). *What is Cloud Forensics? Objectives and Challenges*. Provendata. https://www.provendata.com/blog/what-is-cloud-forensics/

SalvationDATA. (2024, November 20). *Cloud Forensics*. https://www.salvationdata.com/knowledge/cloud-forensics/