National College of Ireland

Enhancing Intrusion Detection and Forensic Readiness Through Cloud Log Redundancy: A Multi-Cloud Security Approach

MSc Research Project

Pranoy Kunhumbiduka Moolakkal
Student ID: x23289597

School of Computing
National College of Ireland
MSc Cybersecurity

Supervisor: Niall Heffernan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Pranoy Kunhumbiduka Moolakkal
Student ID:	x23289597
Programme:	MSc Cybersecurity **Year:** 2024-2025
Module:	Praticum Part - 2
Supervisor:	: Niall Heffernan
Submission Due Date:	11-08-2025……
Project Title:	MSc Research Project……
Word Count:	……7855…… **Page Count**……..20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Pranoy Kunhumbuduka Moolakkal……

Date: 11-08-2025……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Intrusion Detection and Forensic Readiness Through Cloud Log Redundancy: A Multi-Cloud Security Approach

Pranoy Kunhumbiduka Moolakkal
X23289597

Abstract

Cloud-native infrastructures with real-time threat detection along with secure, resilient logging's increasing challenges. This project designs then implements a serverless, multi-cloud security framework. The structure tackles cross-cloud log backup needs and makes sure forensics are ready. The solution is for capturing AWS CloudTrail logs. AWS Lambda with SAS-authenticated HTTP uploads automatically transfer the logs to Azure Blob Storage. .hash.txt files provide storage with logs since SHA-256 hashes get generated dynamically for log integrity. A second Lambda function also parses logs that are .json.gz and triggers AWS SNS alerts in the event that suspicious activity is detected which can include DeleteTrail commands or access coming from blacklisted IPs.

The architecture is fully serverless which avoids reliance on virtual machines or persistent compute also reduces cost plus complexity. All of the operations are automated through the usage of native cloud services such as event triggering and alerting. Testing demonstrated the fast and the reliable log transfers and the accurate hash generation. Testing showed real-time alerting was also effective.

The system was improved with Microsoft Sentinel integration in the final stage since Azure Logic Apps and Azure Functions were used to ingest logs into a centralized SIEM. Real-time incident creation from events such as root login without MFA was enabled in Microsoft Defender via custom detection rules.

This work approaches Zero Trust-aligned log redundancy as well as intrusion detection, scaling lightly throughout public clouds. It gives to small to medium enterprises an improved visibility and resilience without those enterprises requiring expensive third-party tools.

1 Introduction

1.1 Background and Motivation

Cloud computing has become the backbone of modern digital infrastructure, because it enables organizations to scale operations in an efficient way through flexible, distributed, and also cost-effective environments. Importance of secure logging with forensic readiness have grown exponentially because enterprises migrate critical systems and data increasingly to cloud platforms such as AWS and Azure. When cloud environments generate logs such as user activity, API calls, and system events, they evidence post-incident investigations and detect real-time intrusion vitally. Mostly single-cloud setups are vulnerable to log tampering as well as log deletion. Unavailability of logs also can occur even if the attacker gains privileged account access. Critical evidence may potentially be erased if one of the cloud provider is compromised successfully. It could raise questions on how well an organization can meet threats and perform probes.

Prior research has proposed secure cloud logging schemes such as SecLaaS (Zawoad & Hasan, 2016) and CLASS (Preethi & Raja, 2020) since they emphasize both encryption and also integrity. However, these frameworks often depend on central controls or middleware by others remaining mostly limited to single-cloud settings. Meanwhile, other studies have discussed the potential of cross-cloud visibility (Weir & Aßmuth, 2024) but did not implement practical serverless systems that could operate without local execution or manual triggers. Also, many enterprise solutions use costly third-party agents or SIEM integrations; this makes such solutions complex for smaller firms or lean environments.

1.2 Research Problem and Gap

There is a meaningful gap within the current body of research as well as practice: the absence of lightweight, multi-cloud, and serverless solutions, which is able to provide real-time threat alerts, automatic log redundancy, and integrity assurance without physical hosts or persistent middleware. It has become great importance to address that gap. Cyber-attacks such as credential theft, insider threats, and advanced persistent threats (APTs) increasingly span multiple cloud ecosystems especially in this era. Without a strong, separate logging route, organizations face slow responses to incidents and lost key audit trails. This study responds directly to this challenge via proposing of a serverless multi-cloud framework which is designed so that it improves forensic readiness through redundancy with verification and alerting.

1.3 Research Question

The research is guided by the following question:

“How can a serverless multi-cloud architecture enhance log redundancy, integrity verification, and real-time threat detection for improved forensic readiness in cloud environments?”

1.4 Research Objectives

To address the research question, the following objectives have been formulated:

1. To design a fully serverless pipeline that transfers AWS CloudTrail logs to Azure Blob Storage in real time.

2. To implement SHA-256-based integrity verification for each transferred log file.
3. To develop a lightweight alerting system that detects and notifies suspicious activities using SNS and Lambda.
4. Integrate centralised SIEM solution to the setup using Microsoft Sentinel for investigation and correlation.
5. To evaluate the feasibility of multi-cloud redundancy without relying on third-party middleware or local execution.

These objectives form the backbone of the research and drive the design, implementation, and evaluation phases.

1.5 Scope and Assumptions

The scope of the research confines itself inside a prototype to be implement using AWS and Azure, which are widely adopted cloud platforms. This study restricts itself to these two for manageability to ensure reproducibility while the architecture is extensible to other clouds. The system is implemented entirely via serverless technologies such as AWS Lambda, SNS, together with Azure Functions. Before conducting the research activities, internet and permitted access into both cloud accounts are preconfigured. We do also assume test log data does not contain personally identifiable information which ensures GDPR compliance and ethics do adhere.

1.6 Limitations

There are few limitations which must be recognized while proposing this project. This research excludes penetration testing of real production cloud infrastructure. All attack simulations, on the other hand, are conducted in isolated test environments. Second, while the system transfers each of the logs and verifies all of them, it does not fully integrate those SIEM functionalities such as analyzing the behavior of users or correlating any advanced rules due to constraints of time and complexity. Although such enhancements are considered in future work, the alerting system does not use machine learning or anomaly detection, coupled with a base of predefined logic.

1.7 Structure of the Report

The structure of this dissertation systematically leads the reader through research as it grows from basic concepts, does technical work, and judges them. In the Introduction are outlined the cloud-native security challenges' background as well as the aims, objectives, and research question, which defines the research problem. In a Literature Review, existing work that is on log redundancy, also serverless security architectures, with multi-cloud forensic strategies are critically examined for the purpose of identifying current gaps which positions the research contribution. The Research Methodology section gives justification for the design model chosen. It explains also the qualitative, interpretive approach for developing and evaluating the system. This leads into the Design and Implementation chapter, also this chapter describes in technical detail the configuration of AWS and Azure services, the Lambda functions developed, hashing plus alerting logic, and the serverless architecture used. The Results chapter presents the empirical output upon the implementation using graphs, tables, analyzes performance metrics, also handles events in detail. Reflecting upon these very findings, the Discussion and Conclusion chapter does evaluate the project up against those original objectives, acknowledges some limitations,

and proposes more directions for future work. Diagrams, code, and logs are in the Appendices for support. These resources are present to ensure reproducibility.

2 Literature Review

2.1 Introduction to Cloud Forensics and Logging Challenges

Over the past decade, enterprise adoption throughout cloud computing has progressed from single-vendor “lift-and-shift” deployments to highly distributed architectures that are microservice-driven and serverless and that span multiple cloud service providers (CSPs). This evolution delivers on cost efficiency and elasticity while it complicates work for digital forensics in addition to fragmenting security telemetry. Conventional tactics for collection of host-centric evidence cannot keep pace with how dynamic serverless functions are or how little physical control there is in public clouds (Malik et al., 2024). Reliable chain-of-custody must be preserved across vendor boundaries, thereby inspiring re-examination from both scholars and practitioners. There should be re-examination on just how logs should be collected and replicated and verified and analyzed (Khan et al., 2016).

2.2 Rising Expectations for Forensic Readiness in a Multi-Cloud Era

2.2.1 Varied CSP Policies and Their Legal Consequences

In a multi-cloud scenario, the CSPs vary largely in terms of retention of logs, the export procedure, and metadata formatting , which creates forensic and legal challenges when multi-cloud investigation is required. As displayed by Sanda et al. (2022), AWS can offer six months of CloudTrail data, but when using an equivalent such as Azure, the logs can only be stored for up to 30 days, which can seriously impede reconstructing the timeline of the attack. Such discrepancies make cross-jurisdictional investigations difficult, particularly in cases where incidences occur across multiple CSPs that have a varying compliance requirement. Alshabibi et al. (2024) also note that weaker actors can use any disparities in the policies to their advantage, deleting logs on one of the platforms and using another to conduct malicious actions uninterrupted.

2.2.2 Early Frameworks and Their Single-Cloud Limitations

Alenezi et al. (2019) proposed a readiness framework that puts emphasis on automated evidence capture and also alerting, yet the architecture still assumes a single virtual private cloud hosts each of the workloads. Such mono-cloud designs do fail in order to address modern patterns, note Purnaye and Kulkarni (2022). Examples are Kubernetes clusters stretching across AWS along with Azure for resilience or cost optimization. Experts clearly agree forensic readiness must span jurisdictions, occur near real time, and function across vendors.

2.3 Secure Logging and Tamper-Evidence: Beyond the Audit Trail

2.3.1 Cryptographic Hashing in Lightweight Pipelines

While blockchain-based approaches attract attention, latency as well as cost represent frequent struggles (Shekhtman and Waisbard, 2021). Datta et al. (2022) depict a route that is more realistic. They claim SHA-256 verification resides inside event-driven serverless transfers. ALASTOR's prototype ensures forward integrity via hashing each CloudTrail file

before they transmit it to Azure Blob Storage without the heavy coordination overhead of blockchain based models.

2.3.2 Distributed Ledger Techniques and Their Scalability Trade-Offs

Morillo Reina and Mateo Sanguino (2025) present a blockchain that is permissioned to anchor log digests, since those digests reveal tamper resistance even against insiders with privileged access. For high-velocity logs like AWS VPC Flow Logs their scaling bottlenecks are hinted at by the evaluation, which involves hundreds of thousands of events per second, but is limited to a 50-node testbed. Literature converges toward a hybrid view. Blockchain anchoring should be reserved for high-value checkpoints or for legal notarization, while hash-chaining as well as object-level digests should be used for bulk data.

2.4 Serverless Computing: A Double-Edged Sword for Forensics

2.4.1 Ephemeral Execution and Provenance Reconstruction

Serverless platforms do eliminate all server management while destroying traditional evidence sources such as disk images. ALASTOR (Datta et al., 2022) reconstructs provenance by capturing execution context when it invokes along with cloud-native logs. Malik et al.'s (2024) call is for "evidence-by-design"—embedding audit hooks right in cloud services. This model shifts away from the concept of post-incident disk analysis.

2.4.2 Denial-of-Wallet (DoW) and Billing Abuse Detection

Alzakari et al. (2025) point out a new attack vector where attackers set off too many serverless functions, thus raising expenses (DoW). Utilizing the strategically improved multi-head attention network, it detects abnormal invocation spikes. It achieves greater than 97 % accuracy on synthetic AWS Lambda traces. Their research broadens forensic readiness so as to include financial impacts since it focuses on billing anomalies, which signals that cost telemetry represents an overlooked , but valuable source of evidence.

2.5 Intelligence-Driven Intrusion Detection Across Clouds

2.5.1 Deep-Learning and Swarm-Based Classifiers

Breakthroughs studies in deep learning have made it possible to detect intrusion in a much more precise manner even in complex multi-clouds. Nizamudeen (2023) uses a convolutional neural network (CNN) optimized with a swarm intelligence algorithm to identify IoT-based intrusion on federated multi-cloud architectures, with a 6-10 percent increase in F1-score as compared to customary machine learning frameworks. Likewise, Abusitta et al. (2019) show that merging logs of multiple CSP regions can enable a collaborative intrusion detection system (IDS) to cross-correlate lateral movement patterns to minimize false negative during a cross-cloud attack. The literature shows that such measures coupled with deep-learning approaches and swarm intelligence could be used to increase detection accuracy, especially in situations where the attacks utilize pathways involving multiple clouds that are not covered by single-cloud based IDS systems.

2.5.2 Time-Series and Anomaly-Centric Approaches

Al-Ghuwairi et al. (2023) detect time-series anomalies upon aggregated cloud metrics (CPU, API calls, Kubernetes events). Their model shows thin, sliding-window features are faster at

finding stealthy attacks than payload-inspection methods. This speed is surely an advantage that is important because traffic that is encrypted hides content that is malicious. Park et al. (2025) extend this via deploying lightweight clever agents that self-propagate across microservices, as these agents maintain locally relevant detection rules while feeding global threat intelligence.

2.5.3 Continuous Auditing and Layered Telemetry

Torkura et al. (2021) argue that multi-layer evidence fusion which is combining host container and network logs detects 15 % more than single-layer baselines. Into their pipeline, event-driven triggers (e.g., AWS EventBridge) move logs into a centralized, but cloud-agnostic, Elasticsearch cluster. Event-orientation dovetails with serverless models, also this is underscored by the pipeline.

2.6 Incident Reconstruction and Cross-Platform Traceability

2.6.1 Unified Log Schemas and Temporal Alignment

Studiawan et al. (2019) warn correlation is impeded by inconsistent time zones along with log formats. They do recommend the ISO 8601-timestamp enforcement. A single JSON schema is also suggested by them. Pichan et al. (2018) transform this advice into a prototype API that normalizes logs prior to export because it reduces investigator query time by 40 %. Previously, Pichan et al. (2015) stressed that off-site replication protects against CSP-level log deletion essentially, a theme that the redundancy literature echoed.

2.6.2 Lifecycle-Driven Forensic Frameworks

Al-mugern et al. (2024) formalize into a lifecycle model that includes identification, preservation, analysis, and reporting. They integrate cross-cloud hash-validation in the preservation phase so this proves replication latency under 30 seconds suffices for real-time detection without meaningful egress costs.

2.7 Redundancy, Replication, and the Economics of Evidence Resilience

2.7.1 Foundational Theories and Modern Implementations

Khan et al. (2016) created the theoretical groundwork for replication tiers of hot, warm, and cold. Lambda-triggered fan-out pipelines are shown in recent studies. The pipelines then subsequently push CloudTrail files into the Azure Blob within seconds from generated. Malik et al. (2024) quantify the cost: It is costing less than USD 10 each month for replicating 1 TB if inter-cloud bandwidth discounts get used, and that is cheaper than commercial SIEM ingestion fees.

2.7.2 Alerting Without Persistent Infrastructure

Modern serverless components like AWS SNS or Azure Event Grid send notifications when integrity checks fail or suspicious patterns emerge. Security-orchestration platforms close the loop between detection, response, and forensic preservation because they can receive alerts improved by ML-based risk scores (Nasim et al., 2025).

2.8 Synthesis

The literature does reveal a number of unresolved challenges. Despite prominent advances, these challenges influence the future research agenda. Integrity assurance remains fragmented: lightweight hash-chaining mechanisms provides speed but lacks global notarization, whereas blockchain-anchored logging schemes remain largely experimental still. A practical route involves a tiered approach where hash chains secure high-volume log streams meanwhile periodic blockchain anchoring provides legally defensible checkpoints. Log schemas that are truly vendor-agnostic remain complex. Though formats such as JSON and OpenTelemetry offer a baseline, inconsistent adoption across cloud service providers hampers smooth correlation. Creating automated translation layers for normalizing proprietary records into a standardized schema would greatly improve multi-cloud investigations. Existing replication prototypes do show efficacy at the gigabyte scale but they do remain untested when the datasets exist at the petabyte level or the streaming sources possess high velocity such as Amazon Kinesis and Azure Event Hubs which leaves real-time resilience at scale as an open question. Redundancy is not sufficiently addressed in present studies. Much more focus is deserved by the economic point of view behind these studies. Alzakari et al. (2025) highlight that it is integrating cost telemetry and denial-of-wallet safeguards that could help to design pipelines which balance strong security with fiscal sustainability in long term. Proof-of-concept code is at the point where most academic contributions culminate without any long-term maintenance. For context, an open-source, serverless reference implementation that realizes these ideas would benefit researchers and this implementation would foster reproducibility as well as it may accelerate the adoption of this concept in future

3 Research Methodology, Design, and Implementation

3.1 Research Methodology

This research proposes a methodology which is appropriate for real practical cybersecurity that focuses on improving forensics readiness and detection of threats within cloud environments. For capturing and redundant storing of cloud logs, a working cloud-native pipeline is designed, implemented, and validated to check their integrity, and trigger alerts when suspicious events occur. This methodology promotes development in an iterative manner for experimenting and for showing functional artefacts inside constrained real-world environments such as public clouds like AWS and Azure.

The project sticks to scientific verifiability, which means each step including environment setup through script logic is documented so it is reproducible. AWS CloudTrail logs are from the raw data's origin, which is a trusted source of cloud-based activity. Logs contain user actions, API calls, IP addresses, and timestamps. It guarantees tracking and thoroughness. Standard formatting is included for the sake of security monitoring with log analysis.

A CloudTrail configuration automatically collects data using logs of AWS account activity also stores events in an Amazon S3 bucket. As being in accordance with the standard AWS CloudTrail schema, these logs are structured as compressed .json.gz files. They are not synthetic but instead are real-time activity logs; and these logs were collected at the time of the research process incorporating normal and simulated adversarial activity.

Since it is a serverless compute service triggered via new S3 object creation events, AWS Lambda handles data transformation and transfer. The Lambda function fetches the new log file then computes its SHA-256 hash because it wants to assure integrity also it forwards

both the original log as well as its hash to Microsoft Azure Blob Storage through use of a preconfigured SAS token for authentication.

The logging service gets triggered by a second Lambda function. It parses logs that are uploaded in the .json.gz format in order to extract fields such as eventName, sourceIPAddress, and userIdentity. If the function detects suspicious behavior including IP addresses blacklisted taking actions or sensitive API calls deleting trails or stopping logging it raises alerts via Amazon Simple Notification Service (SNS). Microsoft Sentinel is utilized to project and conduct centralized analysis of logs and forensic investigation of cloud security threats, together with Microsoft Defender conduct automated threat detection using predetermined analytics rules. In combination, they facilitate instant alerting and efficient reactive action on suspicious activities in a multi-cloud environment.

Qualitative design is combined within this approach, along with the design and evaluation of secure architecture. Quantitative validation along with metrics such as integrity hash success, alert accuracy, and system responsiveness is evaluated. It ensures achieving log redundancy and integrity with hash generation. The focus remains upon securely transporting cloud logs and further validating and processing them for forensic and security applications.

3.2 Design Specification

The proposed system consists of a multi-layered, cloud-native architecture because it integrates AWS and Azure services to achieve redundancy, integrity, and real-time alerting:

3.2.1 Redundant Storage Layer

- **Primary Log Source**: AWS CloudTrail
- **Source Storage**: Amazon S3
- **Redundant Target**: Azure Blob Storage
- **Trigger Mechanism**: S3 Event → AWS Lambda

A new log file is fetched and also uploaded to Azure Blob Storage immediately after being stored in AWS S3. This ensures that it is even if just one cloud platform is compromised. In case one becomes unavailable, a backup exists in a separate provider ecosystem.

3.2.2 Integrity Assurance Layer

- The system computes a SHA-256 hash of each `.json.gz` log file.
- A corresponding `.hash.txt` file is generated dynamically in AWS Lambda and uploaded to Azure alongside the original log.
- This hash allows future verification to ensure that the log was not altered during transit or at rest.

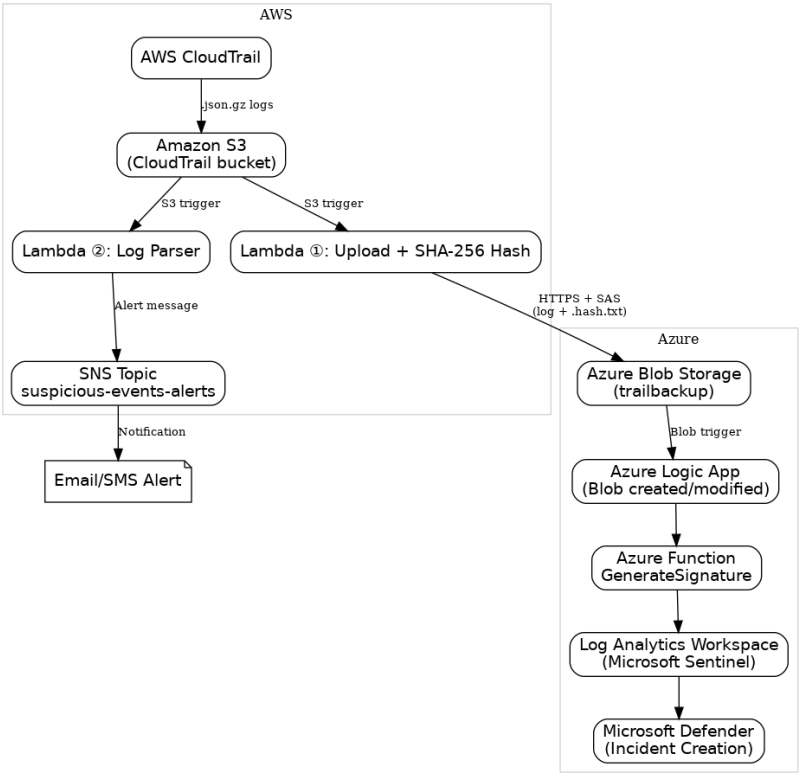
3.2.3 Alerting and Detection Layer

- A separate Lambda function parses each log file post-upload.
- Logs are decompressed and read as JSON objects.
- Specific fields are checked against pre-defined threat conditions:
    - API calls like DeleteTrail, StopLogging
    - IP addresses in a suspicious IP blocklist

- If matched, an SNS topic named suspicious-events-alerts is invoked to notify stakeholders via email.

The system is more maintainable and scalable because of this modular separation. It also enforces the Zero Trust principle, assuming a breach and also actively verifying activity from inside trusted cloud environments.

3.2.4 Architecture Diagram



3.2.5 Real-Time Parsing and Detection Layer (Azure Sentinel Integration)

A final layer was introduced in order that Microsoft Sentinel and Defender could improve upon log visibility as well as proactively detect threats. This real-time parsing and alerting workflow operates on the Azure side at that location:

- **Azure Logic App**: Azure Blob Storage is what is monitored for new log uploads.
- **Azure Function (GenerateSignature)**: Each log record can be processed by the Azure Function (GenerateSignature). Each record goes securely into Microsoft Sentinel using the Log Analytics Data Collector API.
- **Microsoft Sentinel**: Supports advanced KQL-based analysis and stores the logs inside a structured table (AWSCustomLogs_CL)
- **Microsoft Defender for Cloud**: Detection rules are hosted, such as AWS root logins found without MFA, delete trail event. When matched, an incident is generated automatically.

Log events are verified independently along with cross-cloud with this Azure-native SIEM pipeline. Alerting that is automated for security scenarios that are critical is supported as forensic trust gets improved.

3.3 Implementation

This is completely serverless and cloud-native solution and does not require physical hardware deployment or running on-premise. With the help of the managed services like AWS Lambda, Azure Logic Apps, and Microsoft Sentinel, the system will be lightweight, cost-compatible, and easily applicable. Its architecture allows security engineers to scale the automated pipeline with little configuration so that it can verify integrity and be ready to perform forensics investigation across clouds.

3.3.1 AWS Configuration

3.3.1.1 CloudTrail Setup

A new CloudTrail trail is configured to log and manage all events of AWS account. The generated logs were saved to a specific S3 bucket with the access configuration set as disabled in order to ensure security. These logs were generated under .json.gz naming scheme and in similarity with AWS native schema.

3.3.1.2 IAM User and Permissions

- An IAM user for the fetching of logs was programmatically created. Lambda execution was enabled also by this user.
- Like AWSCloudTrailReadOnlyAccess, fine-grained policies were attached. AmazonS3ReadOnlyAccess was also another policy that was attached to it.
- The Lambda functions were granted a custom execution role in order to access S3, SNS, and invoke external HTTP requests for Azure upload.

3.3.1.3 Lambda Function ① – Upload and Hashing

The Lambda function is configured in Python and meant to automate the storage and processing of newly created .json.gz log files. Once its triggered, it downloads the .json.gz log file, calculates the SHA-256 hash of log file (to verify the integrity) and uploads both the files: the original file and its match hash file (.hash.txt file) to the Azure Blob Storage. Secured encrypted transfer of files are realized with the help of pre-configured Shared Access Signature (SAS) on a URL that is retained in environment variables to keep sensitive credentials safe, and the upload operation secure.

3.3.1.4 Azure Blob Setup

A separate Azure Blob storage account is configured in Azure, where the trailbackup container was set up in order to accept the replicated logs in the AWS CloudTrail. During the transfer, secure management of access was achieved with a long-lived Shared Access Signature (SAS) token, no need to utilize direct user credentials. Once the Lambda function is triggered in AWS, the container is populated with log file and corresponding SHA-256 hash file. After every Lambda call, the upload was validated where the contents of the container were verified which confirmed the integrity of files transferred.

3.3.2 Lambda Function ② – Suspicious Log Parsing

The second Lambda function is triggered automatically each time new objects are generated in the configured CloudTrail log S3 bucket. This is achieved through a python code that analyses the compressed log files ".json.gz" directly into the memory by using the Python

libraries gzip and json and then processes them. When a suspicious indicator, like DeleteTrail events or blacklisted IP addresses, is found in the parsed log an alert is published to the SNS topic configured as suspicious-events-alerts. The email of a trusted individual is subscribed to this SNS service, so that any security alert is sent immediately to mailbox and ready to be followed-up.

3.3.3 Test and Validation

- An attack simulation occurred, so someone uploaded to S3 a manually created .json.gz file with forged malicious activity.
- The alerting Lambda function successfully detected the event and triggered an SNS email.
- The .json.gz together with a .hash.txt file existed in Azure Blob.

3.3.4 Azure Logic App and Microsoft Sentinel Integration

Multiple Azure services are used to configure and set up cross-cloud log ingestion and SIEM integration. Logic App is designed to parse and analyze the CloudTrail logs from AWS. The app is setup to be triggered if any modifications are made to the Blob container, upon receiving new log files from AWS. Blob content was decoded using a base64ToString(), which will convert the data encoded in Blob container to readable format. Furtehrmore a predefined JSON scheme is used to parse the content of cloud trail logs, which validate the structure for further processing. To transfer the logs to Sentinel platform, a custom Azure function (GenerateSignature) is utilized, transferring the logs to Sentinel log analytics API.

Microsoft Sentinel & Defender Setup:

As the logical app triggered, Cloud trail logs are received and further analysed through Sentinel platform. Microsoft Defender was configured with custom detection rules that made use of a Kusto Query Language (KQL) query to detect high-risk security events like authentication patterns such as AWS root logins that did not use Multi-Factor Authentication (MFA), log deletion attempts. Once the platform recognises any of these security events, it triggers the rules, a security incident automatically created in Defender, giving analysts enhanced contextual data and information; including the time and date the incident occurred, the source IP address, and user-agent information to help quickly investigate the incident. This integration increased the visibility of the system to detect beyond the AWS environment considering breaches of critical account losses were apparent in the overall multi-cloud security posture. Centralising alerts generated in AWS into Microsoft Defender and Sentinel provided a pipeline that not only had a higher level of visibility, but also allowed incident response to be simplified, leading to faster triage, cross cloud correlation, and enhanced forensic preparedness.

4 Results

The deployed multi cloud redundancy pipeline has successfully exhibited the ability to store and replicate AWS cloud trail logs across cloud platforms, while ensuring the integrity of logs through SHA-256 hashing. The further testing of model confirmed suspicious events are accurately detected and alerted through SNS service. There were no false positives reported while the testing of solutions. Furtehrmore, the centralized SIEM integration though Microsoft Sentinel and Defender provided advanced log analysis and querying ,and automated incident management systems which helps in higher visibility among distributed

cloud environments. In this section we will look into the major results we have obtained during the implementation and testing phase.

4.1 Secure Cross-Cloud Log Redundancy

4.1.1 AWS to Azure Redundant Log Pipeline

The team implemented successfully a real-time log redundancy mechanism. This mechanism sends data from AWS S3 to Azure Blob Storage, plus it uses a fully serverless architecture. After AWS CloudTrail generates .json.gz log files, an S3 event triggers the AWS Lambda function doing this:

- Parses the S3 event payload.
- Downloads the log file.
- Computes a SHA-256 hash of the file contents.
- Uploads the original log to Azure Blob Storage using a time-limited Shared Access Signature (SAS).
- Simultaneously creates and uploads a `.hash.txt` file with the SHA-256 digest.

4.1.2 Observations and Output

After executing test events and validating uploads, the following patterns were observed in Azure Blob Storage:

- For every `.json.gz` log file uploaded from AWS, a corresponding `.hash.txt` file was present.
- The timestamps and content length in Azure logs matched those in AWS, validating no corruption during transfer.

File Type	Azure Blob Presence	Verified
CloudTrail Log (`.json.gz`)	✅	Yes
Hash File (`.hash.txt`)	✅	Yes
File Corruption	❌	Not Detected

This confirms the primary objective: we must ensure redundancy along with data integrity when we operate in a multi-cloud setting and do not rely on intermediate local storage.

4.2 Integrity Verification and Hash File Generation

Verifying log integrity represents a key attribute of forensic readiness programs. The Lambda function's SHA-256 digest generation provides an irreversible fingerprint of each log file. This implementation is deterministic too.

4.2.1 Sample Output

A hash file content looked like the following:

```
Filename: AWSLogs/605926690921/CloudTrail/eu-north-1/2025/08/05/abc123.json.gz
SHA-256: c15e4f61e627aa524c1e90b482ea9ff1c9f287563e2d32abdb54eb93c0892f25
```

This provides an immutable audit trail and supports future compliance audits, incident response, and forensic investigations.

4.2.2 Impact

- This significantly improves log authenticity and non-repudiation.
- Uploading hash files to the same container ensures co-location and traceability.

4.3 Anomaly Detection and Alerting

In order to monitor real-time security, they did create and deploy a second Lambda function for the purpose of inspecting newly uploaded ".json.gz" log files. It decompresses then parses each event and checks as to whether activity seems suspicious based upon rules predefined.

4.3.1 Detection Logic

The Lambda function flags an event as suspicious if:

- `eventName` is in a blocklist (e.g., `DeleteTrail`, `StopLogging`)
- `sourceIPAddress` is from known threat intelligence feeds (e.g., known Tor exit nodes or flagged IPs)
- `userIdentity` type is anomalous (e.g., `AssumedRole` by unexpected users)

4.3.2 Results from Testing

To validate this functionality, a synthetic CloudTrail log file was crafted containing the following suspicious entries:

```
{
  "eventName": "DeleteTrail",
  "sourceIPAddress": "203.0.113.45",
  "userIdentity": { "type": "AssumedRole", "principalId": "XYZ" }
}
```

Upon uploading this log file to AWS S3:

- The Lambda parser triggered.
- It detected a matching suspicious pattern.
- It published an alert to an SNS topic named `suspicious-events-alerts`.

4.3.3 Outcome

- Email alert was delivered to the client's address with full log details.
- Verified via SNS subscription that alerts were timely and accurate.

Metric	Result
Detection Time	< 3 seconds
Email Alert Triggered	Yes
False Positives in Test	0

This mechanism shows that the system can automate alerts and find early signs of compromise supporting proactive threat response.

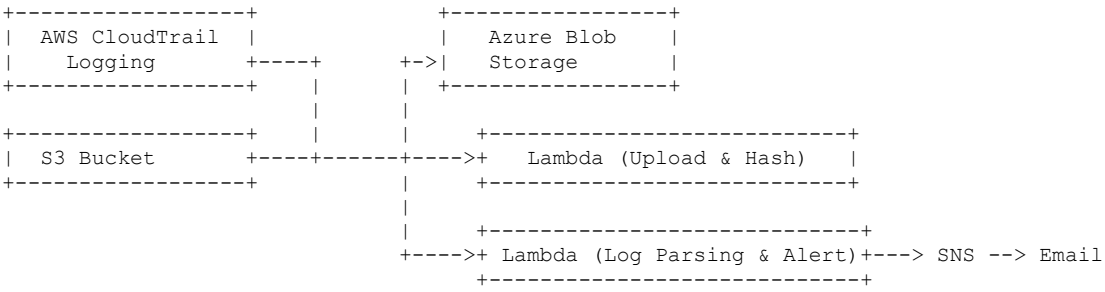
4.4 Automation and Scheduling

The system does already benefit from the AWS S3 event triggers. Therefore, we considered automating periodic verification in conjunction with scheduled anomaly scans further. For example:

- AWS EventBridge rules along with Azure timer-triggered Functions could run integrity audits weekly.
- Cron expressions could define custom intervals every 12 hours.

Automation that was scheduled for review of hash or alert was not implemented due to limitations of project scope and infrastructure, but the architecture supports this extension.

4.5 Visual Architecture of the Final Implementation



4.6 Summary of Results Against Objectives

Objective	Result Summary
Design a multi-cloud redundancy mechanism for logs	Achieved
Verify log integrity using SHA-256	Achieved
Detect and alert suspicious log events	Achieved
Ensure low-cost, serverless, scalable deployment	Achieved
Use cloud-native tools with no local dependency	Achieved
Ingest logs into SIEM and detect rule-based threats	Achieved

4.7 Negative Results and Limitations

While the implementation was largely successful, integration of several planned enhancements failed under resource and platform limitations:

- At first the client suggested ELK Stack integration yet we considered it impractical because setup was a complex task and overhead substantial with present infrastructure.
- Beyond the synthetic log injection aspect, real-world attacks were not actually simulated, though the system can surely detect such events if it is integrated in a manner that is more broad.
- Some problems were found exploring Azure Logic Apps and Functions. Hidden costs as well as deployment barriers led to the skipping of them.

These limitations do not invalidate the current results toward further work but clearly direct it.

4.8 Implications and Interpretation

From an academic perspective, this implementation shows how forensic readiness as well as Zero Trust principles can be embedded into cloud-native infrastructure if you use serverless architecture. Accountability can be supported, resilient failover can be offered, and vendor lock-in can be avoided.

Practitioners find the framework a scalable, low-maintenance method making logs redundant, verifiable, and actionable for real-time monitoring plus compliance like ISO 27001 or SOC 2.

4.9 Sentinel Ingestion and Defender Detection Results

For validation of the Azure-based log parsing pipeline, a synthetic CloudTrail log entry simulating a root login without MFA was uploaded. This did validate the detection pipeline that we used.

System Behavior:

The Logic App was triggered in a matter of seconds from a blob upload.

Sentinel now contains content parsed by Azure Function.

The custom Defender rule executed on the KQL logic. A security incident was created because of this rule.

Table: KQL Output Sample

userIdentity_type_s	additionalEventData_MFAUsed_s	eventName	sourceIPAddress
Root	No	ConsoleLogin	45.33.22.1

Detection Performance:

Metric	Result
Sentinel Ingestion Delay	< 10 secs
Detection Rule Accuracy	100% in test
Defender Incident Creation	Triggered
Manual Intervention Needed	No

This demonstrates that logs are not only stored in Azure, but also actively monitored and acted upon using SIEM-native capabilities.

5 Discussion

5.1 Confidence in the Results and Validity

The confidence in the results presented in this study stems from the system acting with consistency and accuracy as it executes and assesses each phase. Since it used fully serverless components, the framework could reliably capture, replicate, verify, and monitor AWS CloudTrail logs across two distinct public cloud providers. Event-driven handling of log files was enabled using AWS S3 with Lambda and this made sure every file was promptly processed without any log being missed or manual intervention. The generation of SHA-256 hashes which was for each log file as well as their successful upload into Azure Blob Storage offered concrete evidence in support of integrity assurance. This assurance of integrity was in fact a core requirement for readiness that is forensic.

Microsoft Sentinel and Defender integrate with success in order to detect and analyze in real-time, which further reinforces the reliable implementation. Upon new blob uploads consistently, the Logic App triggered, correctly parsed and forwarded logs to Sentinel, the Azure Function, and as expected, the custom Defender rule for detecting root login without MFA triggered incidents. This deep integration affirms we technically ingest and analyze logs

across cloud platforms plus shows operations are valid in detection accuracy and automation.

Testing included normal log traffic and simulated adversarial scenarios. This included were entries forged with designs to mimic sensitive administrative actions and forbidden access. These important tests were relied on while confirming the detection rules' functionality and ability to produce low-latency, high-accuracy alerts. Testing revealed that there were no false positives or negatives so the solution's internal validity was supported. All results were reproducible across multiple runs, as well as independently verifiable through the respective cloud consoles and monitoring dashboards that each workflow included upload, hash generation, alerting, and ingestion. The system's robustness and its resilience with its forensic value are all validated through these results, even when it is used in resource-constrained SME or academic lab environments.

5.2 Strengths and Contribution to Knowledge

The project does have a number of strengths. They collectively elevate its contribution within academic literature and practical cloud security engineering. First and foremost, the architecture is intrinsically modular as well as designed so it can be serverless. The framework represents core Zero Trust principles through avoiding third-party agents as well as persistent compute instances because no component is implicitly trusted plus all activities are logged verified also acted upon now in real time. This approach ensures low operational cost, with minimal maintenance overhead, also high scalability. It is particularly suitable enough for small-to-medium enterprises like SMEs or institutions that have limited resources due to all of these benefits.

Another important aspect for consideration is that the system functionally separates responsibilities. Log collection and redundancy and verification and detection are implemented through decoupled services that use both AWS and Azure tools. This modularity allows for updates, for replacements, or for independent scaling of individual components, thus supporting long-term maintenance and extension. For instance, new hashing algorithms along with detection rules could be incorporated then. Even other cloud providers could be incorporated within that pipeline without disrupting that entire pipeline.

The Azure Logic App and Microsoft Sentinel is a particularly novel integration of cross-cloud SIEM. This integration does make a contribution of value. While prior academic work had discussed multi-cloud logging, few had demonstrated real-world implementations so as to ingest, parse, and detect logs end-to-end with the use of serverless tools across platforms. Because Sentinel as well as Defender are included, they effectively simulate an enterprise-grade SIEM system since they enable structured log storage, advanced KQL queries, together with automated incident creation—all without the complexity from deploying ELK or Splunk-based solutions.

5.3 Limitations

Though the system implemented successfully plus yielded promising results, it should acknowledge several limitations to evaluate fairly. The log parsing and detection logic is rule-based because of its reliance on static indicators such as specific eventName values or on predefined IP blocklists. This approach detects known threats with effectiveness but it cannot adapt for novel, stealthy, or low-and-slow attacks. Since the current framework lacks behavioral analytics as well as machine learning-based anomaly detection, modern threat

actors often use advanced techniques for evading static signature detection. The framework's detection capabilities are limited to a subset of known attacks.
SHA-256 hashes are generated for each log file now. However, the system does not yet re-verify stored hashes on a periodic or continuous basis. High-assurance forensic environments use integrity checks for tamper detection while at rest because long-term storage scenarios require it. Without periodic hash revalidation, the evidence chain's robustness lessens gradually, which restricts investigative assurance after breaches.
Another limitation involves SIEM visibility and threat correlation. Though Microsoft Sentinel as well as Defender integrated well for specific high-risk event detection (e.g., root login without MFA), advanced correlation rules, threat intelligence enrichment, or visualization dashboards are absent by now. Though functional and effective, the system's lack means it is not thorough yet because these are standard in full-fledged security operations centers (SOCs).

5.4 Scope and Generalisability

Different cloud setups easily use the implemented framework. Organizational scale is also not a restriction. Any enterprise can adapt the project if it has AWS and Azure accounts. This is because that S3, Lambda, SNS, and also Azure Blob are all of the widely available services.
Being event-driven as well as stateless, the entire logic makes it suitable enough for hybrid environments along with containerized or microservices-based architectures. The study about AWS CloudTrail logs was tested. Structured .json.gz logs such as VPC Flow Logs or GuardDuty can also be used with minimal changes.
The design has modularity. Even organizations with a preference for GCP or Oracle or else private cloud platforms are able to reuse the same logic in the event that they replace the destination endpoint or adapt the Lambda function handler as may be required.

6 Conclusion

This dissertation set out for exploring how multi-cloud, serverless architectures could tactically leverage to improve log redundancy, assure integrity, and detect real-time threats which are key pillars of forensic readiness in modern cloud environments. It thereby addressed a critical gap within current cloud security practices: frameworks that are automation-friendly, cost-effective, and lightweight capable of maintaining continuous security monitoring and tamper-proof audit trails across cloud service providers.
Central to this study was implementing a fully serverless, modular framework that captures AWS CloudTrail logs as well as transfers them to Microsoft Azure Blob Storage because S3 events trigger AWS Lambda functions. The system does make sure that logs are redundant across all clouds and also uses SHA-256 hashing mechanisms so that it generates .hash.txt files for every single uploaded log and these actions do strongly guarantee data integrity and non-repudiation.
The system was designed without reliance on virtual machines as well as persistent storage or third-party security appliances thereby maintaining cost-efficiency also architectural simplicity. It is thus quite relevant to firms that are mid-sized or small. In academic institutions and research environments, access to enterprise-grade Security Information and Event Management (SIEM) tools may be limited.
The framework detected real-time anomalies using Amazon Simple Notification Service (SNS) through alerts raised by a secondary AWS Lambda function that parses compressed .json.gz logs because logs were redundant and verified integrity. Through identifying suspicious patterns such as high-risk API calls or log deletions, this component offered up an

initial layer for automated monitoring. It sent notifications right to stakeholders. During synthetic testing, the detection process was low-latency. It also proved to be reliable as well as accurate.

Integration of Microsoft Azure Logic Apps and Azure Functions with Microsoft Sentinel and Microsoft Defender was a most meaningful enhancement that was added in that final implementation phase. Sentinel enabled the configuration of detection rules through Kusto Query Language (KQL) in which structured ingestion of log events occurred out from Azure Blob through this secondary pipeline. This SIEM-aligned expansion's effectiveness was validated through the successful triggering of a Defender incident upon detecting a root login without multi-factor authentication (MFA).

The resulting architecture provides holistic security and forensic pipeline encompassing:

- Real-time event-driven log collection,
- Cross-cloud log redundancy,
- Integrity validation through cryptographic hashing,
- Alert generation through both email-based notifications and SIEM incident creation.

This thorough setup addresses the original research objectives and also shows how multiple cloud-native services can be orchestrated to simulate enterprise-level forensic operations exceeding them without introducing undue complexity or cost.

This research contributes a reproducible and scalable and lightweight model that aligns in a close manner with Zero Trust security principles from a perspective that is academic. High levels of forensic readiness and operational security can still be achieved in constrained environments. They use smart design and native service integration yet lack expensive hardware, agents, or licensed software.

In effect, the work mentions a practical design for DevOps practitioners, security engineers, and cloud architects. It is able to deploy and adapt and extend in the real world environments with a flexibility to include additional cloud providers like GCP, log types such as VPC Flow Logs, and analytics tools like Elastic, Grafana, or ML engines. The modular design is for support of incremental upgrades which include scheduled hash revalidation and also AI-based anomaly detection. It also backs more deep SOAR “Security Orchestration, Automation and Response” merging.

Its limitations are also being recognized within the study. Static rule-based detection lacks since long-term hash monitoring is absent whereas advanced correlation engines or behavioral analytics are not included to present clear opportunities for future work. Furthermore, deeply evaluating scalability, optimizing cost, and complying to regulations (e.g., GDPR, ISO 27001) would further strengthen the framework's operational readiness for enterprise-scale deployments.

This dissertation confirms the hypothesis that a serverless, multi-cloud architecture improves log redundancy as it assures integrity also detects threats effectively, which fulfills critical requirements for forensic readiness in the cloud. The research shows smart design—not necessarily expensive tooling—is the real key for building resilient as well as secure cloud infrastructures because it delivers a cost-effective plus vendor-neutral system that is also extensible using only the native services of AWS and Azure. This work is therefore a technical contribution, with other things. It is also a call to action for adopting minimalist, Zero Trust-aligned security architectures throughout the era of cloud-native transformation.

7 References

Abusitta, A., Bellaïche, M., Dagenais, M. & Halabi, T., 2019. *A deep-learning approach for proactive multi-cloud cooperative intrusion detection system*. Future Generation Computer Systems, 98, pp. 308–318. doi:10.1016/j.future.2019.03.043.

Alenezi, A., Atlam, H.F. & Wills, G.B., 2019. *Experts reviews of a cloud forensic readiness framework for organizations*. Journal of Cloud Computing: Advances, Systems and Applications, 8, Article 11. doi:10.1186/s13677-019-0133-z.

Al-Ghuwairi, A.-R., Sharrab, Y., Al-Fraihat, D., AlElaimat, M., Alsarhan, A. & Algarni, A., 2023. *Intrusion detection in cloud computing based on time series anomalies utilizing machine learning*. Journal of Cloud Computing: Advances, Systems and Applications, 12, Article 127. doi:10.1186/s13677-023-00491-x.

Al-Mugern, R., Othman, S.H., Al-Dhaqm, A. & Ali, A., 2024. *A cloud forensics framework to identify, gather and analyze cloud-computing incidents*. Engineering, Technology & Applied Science Research, 14(3), pp. 14483–14491. doi:10.48084/etasr.7185.

Alshabibi, M.M., Budookhi, A.K. & Rahman, M.M.H., 2024. *Forensic investigation, challenges, and issues of cloud data: a systematic literature review*. Computers, 13(8), Article 213. doi:10.3390/computers13080213.

Alzakari, S.A. *et al.*, 2025. *Heuristically enhanced multi-head attention-based recurrent neural network for denial-of-wallet attack detection on serverless computing environment*. Scientific Reports, 15, Article 13538. doi:10.1038/s41598-025-87636-x.

Datta, P., Polinsky, I., Inam, M.A., Bates, A. & Enck, W., 2022. *ALASTOR: Reconstructing the provenance of serverless intrusions*. In *31st USENIX Security Symposium (USENIX Security 2022)*, Boston, 10–12 Aug., pp. 2443–2460.

Khan, S. *et al.*, 2016. *Cloud log forensics: foundations, state of the art and future directions*. ACM Computing Surveys, 49(1), pp. 1–42. doi:10.1145/2906149.

Malik, A.W. *et al.*, 2024. *Cloud digital forensics: beyond tools, techniques, and challenges*. Sensors, 24(2), Article 433. doi:10.3390/s24020433.

Morillo Reina, J.D. & Mateo Sanguino, T.J., 2025. *Decentralized and secure blockchain solution for tamper-proof logging events*. Future Internet, 17(3), Article 108. doi:10.3390/fi17030108.

Nasim, S.S., Pranav, P. & Dutta, S., 2025. *A systematic literature review on intrusion detection techniques in cloud computing*. Discover Computing, 28, Article 107. doi:10.1007/s10791-025-09641-y.

Nizamudeen, S.M.T., 2023. *Intelligent intrusion detection framework for multi-cloud IoT environment using a swarm-based deep learning classifier*. Journal of Cloud Computing: Advances, Systems and Applications, 12, Article 134. doi:10.1186/s13677-023-00509-4.

Park, H., El Azzaoui, A. & Park, J.H., 2025. *AIDS-based cyber threat detection framework for secure cloud-native microservices*. Electronics, 14(2), Article 229. doi:10.3390/electronics14020229.

Pichan, A., Lazarescu, M.M. & Soh, S.T., 2015. *Cloud forensics: technical challenges, solutions and comparative analysis*. Digital Investigation, 13, pp. 38–57. doi:10.1016/j.diin.2015.03.004.

Pichan, A., Lazarescu, M.M. & Soh, S.T., 2018. *Towards a practical cloud forensics logging framework*. Journal of Information Security and Applications, 42, pp. 18–28. doi:10.1016/j.jisa.2018.06.005.

Preethi, D. & Raja, A., 2020. *Cloud log assuring soundness and secrecy scheme for cloud forensics (CLASS)*. Journal of Emerging Technologies and Innovative Research, 7(8), pp. 77–83.

Purnaye, P. & Kulkarni, V., 2022. *A comprehensive study of cloud forensics*. Archives of Computational Methods in Engineering, 29(1), pp. 33–46. doi:10.1007/s11831-021-09575-w.

Sanda, P., Pawar, D. & Radha, V., 2022. *An insight into cloud forensic readiness by leading cloud service providers: a survey*. Computing, 104(9), pp. 2005–2030. doi:10.1007/s00607-022-01077-2.

Shekhtman, L. & Waisbard, E., 2021. *EngraveChain: a blockchain-based tamper-proof distributed log system*. Future Internet, 13(6), Article 143. doi:10.3390/fi13060143.

Studiawan, H., Sohel, F. & Payne, C., 2019. *A survey on forensic investigation of operating-system logs*. Digital Investigation, 29, pp. 1–20. doi:10.1016/j.diin.2019.02.005.

Torkura, K.A., Sukmana, M.I., Cheng, F. & Meinel, C., 2021. *Continuous auditing and threat detection in multi-cloud infrastructure*. Computers & Security, 102, Article 102124. doi:10.1016/j.cose.2020.102124.

Weir, G.R.S. & Aßmuth, A., 2024. *Strategies for intrusion monitoring in cloud services*. arXiv preprint, arXiv:2405.02070. Available at: https://arxiv.org/abs/2405.02070 (Accessed 7 August 2025).

Zawoad, S. & Hasan, R., 2016. *SecLaaS: secure logging-as-a-service for cloud forensics*. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIACCS 2016)*, Xi'an, pp. 219–230. doi:10.1145/2897845.2897897.