Reinforcing Security in Multi-Cloud Environment through Integrated Secure Data Transfer and Process Mining

MSc Research Project
MSc in CyberSecurity

Mahesh Gavhane
Student ID: 23111984

School of Computing
National College of Ireland

Supervisor: Prof. Khadija Hafeez

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Mahesh Gavhane
Student ID:	23111984
Programme:	MSc in CyberSecurity
Year:	2024
Module:	MSc Research Project
Supervisor:	Prof. Khadija Hafeez
Submission Due Date:	12.12.2024
Project Title:	Reinforcing Security in Multi-Cloud Environment through Integrated Secure Data Transfer and Process Mining
Word Count:	6578
Page Count:	22

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:	Mahesh Gavhane
Date:	27th January 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):	

Reinforcing Security in Multi-Cloud Environment through Integrated Secure Data Transfer and Process Mining

Mahesh Gavhane
Student ID: 23111984

Abstract

Securing the multi-cloud environment has become essential in this world, where cybersecurity is a major concern. This research explores how data security could be improved in multi-cloud environments by integrating secure file transmission, dependable logging, and advanced analytical techniques like process mining. When data is being transferred within cross-cloud platforms, frequent cyberattacks and data breaches in cloud settings jeopardize data confidentiality and integrity throughout transfer operations, highlighting the need for stronger security measures. This study involves log analysis mechanisms that could identify deviations during file transfer. It will use a combination of process mining, visualization techniques, and secure data transfer protocols. This research focuses on automating and securing file transfers while ensuring compliance with security requirements by implementing a strong framework in a multi-cloud setting involving important platforms like AWS and Azure. In addition to addressing today's security issues, this integrated strategy lays the groundwork for security solutions that can react to unexpected spikes in multi-cloud systems.

Keywords— Multi-cloud, Data security, Log Analysis, AWS and Azure Integration, Process Mining, Secure Data Protocols

1 Introduction

With the help of Cloud Computing, users can benefit from their services like application, infrastructure, and storage services. As per the company or user needs, the cloud user can take access and configure the hardware and software as they wish. As there are many advantages of using Cloud Computing, there are some limitations and challenges linked with it as well. The main challenges involved while using Cloud Computing are, security, load balancing, privacy, and cost estimation. As the data of the user and the application is on the Cloud, security is among the important challenges that need to be taken care of. There are procedures and policies to protect data on the cloud premises. This also helps to prevent unauthorized users from accessing the data and to prevent malicious attacks. It also protects against data leakage, data alteration, software vulnerabilities, SQL injection, cross-site scripting, and flooding attacks.Khalil et al. (2014)

Due to various attacks, cloud subscribers and providers report security issues regularly. In the year 2022, around 49% businesses reported that their 40% of data is sensitive. When this number of businesses increased to 75% after a year of this report was published, half of the average data on the cloud was still encrypted, according to the same report. As there is an increase in cyberattacks on the cloud, businesses need to make sure that their file transfers are safe and secure. If the file transfers on the cloud are not secured, it can cause a data breach and this could affect remote work as well as collaboration. Unauthorized access and compliance violations pose significant threats, highlighting the need for proper management of permissions and adherence to regulations.[1]In the year 2018, there were around 650 million cyber-attacks on cloud users. Along with this, in 2019 cloud users also reported that there were Internet of Things (IoT) attacks, distributed denial of service (DDoS) campaigns, targeted ransomware, advanced phishing campaigns, and attacks targeting containers and cloud services.Lata and Singh (2022)Because of the problems related to service availability failure and possibilities of malicious insiders in the single cloud, customers are given less importance when it comes to dealing with the single cloud provider. Because of this, there's increased demand for multi-clouds.AlZain et al. (2012)

The data could be stored over the cloud and it could be accessed irrespective of location, also data within personal systems could also be secured in a cloud environment. This ensures that cloud computing secures the data that is stored in the cloud system. As the data on the cloud might be important to users, hackers could try to access that data, to avoid this one needs a secure system. When the data is stored in a cloud system by any company, there are chances that any unauthorized user access it. Hence, if the company is trying to store any private data, it should make sure that the cloud system is trustworthy. The vulnerabilities could arise because of poor management in securing data across multiple clouds, which can result in exploitation by the attackers. Hence, cloud service providers must keep developing and upgrading effective security to protect data in multi-cloud environments. Process mining has been identified as a promising research direction for constructing behavioral or workflow models from event logs. It enables a deeper understanding of underlying processes of identifying potential security threats. Despite process mining being considered in security for some time, its application in cloud data security remains largely unexplored. In this work, process mining techniques are utilized as part of a broader framework to analyze event logs and enhance the process of file transfer workflows.Zhang et al. (2023)

With an aim to address operational efficiency and security, this research attempts to build a safe, automated, and well-monitored file transfer system and implement it across multi-cloud environments. The approach of the research combines secure file transmission, reliable logging, and analytical tools to ensure data security in a multi-cloud setup. Process mining techniques are one of the most important parts of the analysis stage which will offer practical insights to efficiently identify and reduce security risks.

1.1 Research Background

In the multi-cloud environment setup, as various cloud providers are involved, security becomes a major concern, especially when thought in the direction of data breaches during transmission. This comes with significant risks which are associated with unauthorized access and attacks on the cloud-based data, which occur particularly when data

[1] https://blog.axway.com/learning-center/digital-security/cyberthreats/unsecured-cloud-file-transfers-dangers-costs

is being transferred between clients and also amongst the multi-cloud platforms such as AWS and Azure. To find threats such as those mentioned above and keep a record of the findings makes use of process mining, which involves analyzing data records for security abnormalities. Since this task is done manually, it is more likely to have some errors. El-Gharib and Amyot (2019) Vervaet (2021)

As Cloud Computing provides scalable and adaptable services across the platforms, this has completely changed the way data is processed and stored. Still, it poses security risks, such as ineffective monitoring mechanisms and vulnerabilities during data flow in both scenarios i.e. within and between cloud environments. While going through this process, managing workflows in cross-cloud platforms, preserving end-to-end encryption, and safeguarding file transfers continue to be crucial concerns. To overcome all these constraints, this research intends to use AWS and Azure to develop and deploy safe, automated, well monitored file transfer systems for both within and multicloud environments. In this method, AWS and Azure will be integrated to automate the file transfers and create logs for further analysis when files are being transferred. It also explores the process mining techniques through the PRoM tool, which analyses workflows and log patterns using the Python library and methods.Islam et al. (2021)Zhou et al. (2020)

1.2 Research Question:

How can secure file transfer protocol, combined with process mining through log analysis, improve workflow efficiency and data security across multi-cloud environments?

This research is centered on how the secure data protocols combined with Python libraries could be integrated to enhance security in multi-cloud environments through log analysis. As organizations on a large scale have started adopting multi-cloud solutions, significantly the security challenges for managing the data across cloud platforms such as AWS and Azure have increased. A secure framework is needed to manage and handle this specific issue of vulnerabilities, abnormalities, and deviations while data transfer could lead to data beaches or any kind of unauthorized access. The eavesdropping and tampering of files during transit, secure data transfer methods make sure that data is transferred safely. Overall, this study explores how transfer protocols along with process mining might be used to enhance the detection and mitigation of security risks. Organizations could also adopt a proactive strategy that utilizes visualization techniques combined with process mining to analyze the data and identify patterns of vulnerabilities, offering a better approach.

1.3 Research Objective

This research study aims to explore the direction of security while data transmission in multi-cloud environments and also address the issues and how a framework will help to resolve the same.

- Design and implement secure file transfer system: To develop and configure an automated file transfer workflow which would ensure the security of data while transfers within and also across multi-cloud environments of AWS and Azure, which uses protocols such as SFTP, Azure Logic Apps, and AWS S3.

- Enhance data security during file transfer: To employ end-to-end encryption, and multi-layered monitoring algorithms which eventually safeguard file contents and also prevent unauthorized access during transit.

- Generate logs and process them for analysis: To make log files in JSON format during file transfer and then convert them in CSV format for further analysis by using various libraries and process mining methods.

- Apply process mining for workflow analysis: Will be using tools such as ProM, which would analyze workflows, identify deviations, and also detect irregularities in the path of execution during file transfers.

- Leverage visualization for workflow analysis: To visualize log analysis, also identify patterns that may indicate abnormalities and deviations.

1.4 Outline

This research paper has the following sections; Section 2 reviews Related Work, identifying existing solutions and research gaps. Section 3 details the Research Methodology, describing the approach. Section 4 presents the Design Specification, outlining the system architecture and requirements and system specifications. Section 5 describes the Implementation of the proposed system. Section 6 provides an Evaluation of the proposed solution and its findings. Finally, Section 7 concludes the research in Conclusion and Future Work, summarizing the key findings and suggesting improvements.

2 Related Work

The term 'multi-cloud environment' refers to the use of various cloud environments and services from different cloud providers. Multi-cloud environment architecture benefits in optimizing resource utilization, which enhances the reliability of the system. However, the security challenges when file transfer happens between multi-cloud environments are still a bit difficult to resolve. Eventually, these methods only work for single-cloud environments but would have some drawbacks for multi-cloud environments resulting in latency, and scalability issues along cross-cloud compatibility. Development in this area of secure data transfers of files amongst multi-cloud environments focuses on cryptographic approaches, such as homomorphic encryption and advanced key management schemes enhancing data privacy.

2.1 Protocol-Oriented vs. Real-World Approaches to Multi-Cloud Security

This category involves two research papers, majorly focusing on the issue of multi-cloud security including various approaches and results. The first paper,AlZain et al. (2012) mentions protocols such as HAIL and DepSky which enhance the data integrity, and confidentiality in multiple cloud environments. Tools such as Proofs of Retrievability (PORs) and Proofs of Data Possession (PDP) were also examined to ensure the authenticity of data. It came to a conclusion that 67% of research focused on single-cloud security in 2009 and 33% on multicloud also mentioning the limitations of the tools used as above, as DepSky lacks support to IAAS and there were confidentiality issues with HAIL. The second paperKanungo (2023) is basically a case study with reports such as the Capitol One and Equifax breaches identifying risks such as attack surface expansion (79%) and complexity in security controls (68%). This comes up with the opinion that cloud-native tools and zero security frameworks are one of the solutions to handle complexities. As both the researches address integrity, confidentiality, and risk mitigation of multi-cloud, while AlZain et al. (2012) focuses on the cryptographic protocols, Kanungo (2023) focuses on real-world case studies. Although they both aim to improve security, their approaches are different; the first is protocol-oriented, while the second is more concerned with thematic analysis and practical application.

With emphasis on security issues and privacy issues, the other two research papers mainly concentrate on cloud computing, mainly multi-cloud and general cloud computing architectures. Technologies for encryption, role-based access control, and privacy enhancement, such as homomorphic encryption are suggested as one of the ways to reduce security vulnerabilities in multi-cloud setups. Mohammad (n.d.)basically addresses issues such as regulatory compliance virtualization, vendor lock-in, interoperability and data breach. Liu (2012) makes the understanding for the architecture of cloud computing, its front-end and back-end architecture, and concept of virtualization, and the differentiation between hybrid, private, and public clouds more clear. Security vulnerabilities in cloud environments such as system transparency, data integrity and access management are also discussed in the above research paper. Even though major security issues are covered in both researches, Mohammad (n.d.)focuses on issues mainly related to multi-cloud and also gives suggestions related to various encryption and access control strategies that are designed for this kind of setting, relying more on compliance frameworks such as GDPR and HIPAA for the multi-cloud settings. These two research papers in comparison suggest that secure access control and data encryption are essential but also mention limitations which include the complexity behind encrypting data on a large scale and challenges that are faced in cross-jurisdictional handling.

2.2 Process Mining Applications for Workflow Efficiency

The three research papers reviewed under the category throw light on the technique of process mining, which is used to create process models and extract data from event logs in order to analyze and enhance the efficiency of workflows. The first research paper, Nicolas and Roger (2022)studies how to use real-time analytics and machine learning are used to enhance services in the IT sector by increasing productivity by 20%, and also automating all the processes of identifying delays. Whereas the second research Vervaet (2021) makes use of heuristic and fuzzy models which results in anomaly detection with acquired accuracy of 85%, also points out the difficulties faced when dealing with unstructured data. The third research, Executions (n.d.) gives views on tools such as α-algorithm for conformance verification, through ‘token game’ which is effective in detecting deviation but has some constraints related to data. Perspectives of process mining such as process, organizational, and case dimensions are also highlighted. This research also involves the use of event logs and Petri nets for modeling, but to identify deviations, particularly the α-algorithm is mentioned. One of the noticeable differences that lie in all three studies is that Nicolas and Roger (2022) gives importance to service productivity, but Vervaet (2021) and Executions (n.d.) major focus on compliance and anomaly detection. Also, this research has faced problems when it comes to handling partial logs and scaling procedures that are complex.

2.3 Data Security Enhancement Techniques in Multi-Cloud

The research papers studied give insight on many different ways that would subsequently improve multi-cloud setting security. In the first paper Ansar et al. (2018), SDSMC, is an ASP.NET implementation, which secures data within different cloud environments, using file slicing and AES encryption. As mentioned this speeds up the encryption, it also has limitations related to high cost and complexity. In the second research Ristic (2013), as mentioned, the researcher uses DepSky, which uses MATLAB to build ByzantineIn the quorum protocols and (k,n) approach to minimize the storage. Without using the server-side code, it ensures data confidentiality, which rescues security risks, with the limitation of empirical testing. Even though both the frameworks mentioned give confidence on data security strengthening, but also point out weaknesses that out be studied further. Here, Ansar et al. (2018) uses tools such as NetBeans, MySQL, and AWS, and addresses the topic of cloud data migration which emphasizes organized strategies such as rehosting and re-platforming, and preparing to enable risk-free and seamless data transfer. This study also mentioned all the crucial security features that tackle dangers such as data loss and intervention. The second study Ristic (2013) uses Triple DES encryption, MD5 hashing, and file locking feature for security to investigate Secure File Transfer Protocols (SFTP) for security during file transfers in between distant systems. Although, both of these papers mainly emphasize data protection, Ristic (2013) mainly discusses securing file transfers on individual remote systems while Ansar et al. (2018) focuses on large-scale cloud migration. The high cost of SFTP implementation and all the platform requirements in cloud migrations are the two major drawbacks.

2.4 Comparative Analysis of Cloud Integration and Automation Solutions

As per the other three research papers reviewed, the main topics covered are cloud integration, automation, and migration using different tools and methods. As per the first research Skrifvars (2022), it proceeds in the direction on utilizing Logic Apps to integrate IFS ERP solutions into Microsoft Azure. This achieves a processing speed of 15 seconds, but it even identifies the limitations of Logic Apps ability how handle complicated data manipulations. The second research Varalakshmi et al. (2023) Mulesoft's Anypoint Platform is discussed, which integrates all the applications across the hybrid cloud environments which utilize API-led connection, eventually increasing agility but also pose some implementation cost challenges. The third paper Wilhelms (2024) gives a vast description of how to create Azure Sandbox Manager which supports moreover 20 resource groups in production, automates environment management, utilizing low code or no code tools such as Power Apps and Logic Apps. Through this review, the major limitations include the complex configurations in Azure Cloud, and high costs along with MuleSoft, but also gives advantages such as improved workflow automation and efficiency in operations.

2.5 Research Niche

The earlier solutions used for secure data transfer and monitoring log analysis for detecting unusual deviations in cross-cloud environments point out various limitations and gaps, which our proposed workflow manages in a detailed way. Existing systems frequently show deficiencies in custom solutions while transferring the files and incomplete analysis of the logs and alert mechanism, complex configuration, and efficiencies. However, this proves beneficial in single cloud platforms but they face difficulties when it involves multi-cloud discussions. Multi-cloud integration offers different challenges, for instance, the tools that are designed for specific platforms can cause compatibility issues and disrupt scalability. The proposed solution successfully and efficiently tackles these gaps by including a secure file transfer protocol, which allows the log analysis, monitoring, and implementation of advanced log analytics by using process mining methods. When data files are transferred, data is encrypted which would be made visible using algorithms (e.g. AES-CTR), While this process is going on, one can confirm that the integrity check is done as a hash-based message authentication code is present. Also, a real-time alert email notification is integrated. By using Azure Logic App for the automation of workflow which is related to file transfer, error handling, and notification management. Also, the integration of Azure Function (UploadToS3) with AWS S3 client allows smooth multi-cloud interaction, the strong functionality of this function also ensures workflow integrity and reliability which deals with platform-specific constraints and provides hybrid compatibility. This solution provides an integrated and valuable improvement for secure, automated along efficient multi-cloud workflow.

3 Research Methodology

Addressing secure file transfer protocol, visualizing techniques, and process mining for identifying the deviations from the expected flow and changes in the file transfer pattern, a planned and systematic approach is utilized to implement this research project. This research paper's implementation starts with setting up the multi-cloud environment by using cloud platforms Azure and AWS. After which the automated Logic app workflow and advanced process mining tools and methods are integrated. In the proposed system, for transferring the file securely, the Logic App has been implemented, which mainly depends on an Azure storage account to store and encrypt the data at rest. This encryption is used to protect the overall workflows and the organization's regulation commitment. Because of this reason, Logic App is chosen for the research implementation. The starting phase of this research implementation involved the setup of the Azure platform with a storage account for setting the SFTP user profile to enable secure file transfer. After the creation of the user profile, the WinSCP SFTP tool was used as a secure file transfer protocol to test the file drop in the servers. after setting up the Azure platform for SFTP, the automated workflow i.e. Logic App has been created with all the security aspects to transfer the file from:

- Within Azure: From the SFTP server to designated storage folders.
- Across Clouds: From Azure to AWS S3 using Azure Functions and the AWS S3 client.

Each step of the file transfer process was perfectly logged to generate the logs. Those logs were fetched in the format of JSON through Azure log analytics workspace and converted into the required structured JSON using schema and parsed the structured JSON into CSV files for further analysis. At this phase, another logic app has been created to automate the entire conversion process and to maintain the regularity in the log data. To make sure the security aspect during the transfer. End-to-end encryption was implemented and during the whole process, data was kept encrypted and decrypted before going to the destination folder path. Alert email notifications were set up to immediately notify for both success and failure notifications which ensure strong security and integrity.

This research also integrates the process mining methods using PROM tool, which helps in analyzing event logs generated by PROM tool in order to identify deviations in file transfer patterns and workflow. Python libraries have been implemented in Jupyter Notebook for detecting and identifying the patterns of unusual behavior. During the whole implementation process, complete documentation of configuration steps, error logs, and setup of the system are carefully maintained to make sure ability to reproduce the same flow. Snapshots of the whole Logic App workflow, file transfer processes and the outcomes were successfully captured to support when used later on. [2] [3]

4 Design Specification

This section depicts the architecture diagram of the proposed solution, highlighting the overall workflow developed for intra-cloud and multi-cloud activities.

4.1 Architecture

The architecture diagram mentioned in Figure 1, demonstrates a secure and automated file transfer that is capable of handling both i.e.intra cloud (Azure to Azure) and across-the-cloud (Azure to AWS) workflow. The workflow starts with user login using the WINSCP SFTP tool, which is based on the SSH authentication method. Upon successful authentication, the user drops the file in the ‘mahesserver1’ incoming folder, the WINSCP transfers the file to an azure hosted sftp server. The processed file moves to the ‘maheshserver2’ outgoing folder (The destination sftp server for the Azure storage account). The primary Logic app will trigger the whole automated workflow. This solution uses an Azure Logic app called ‘x23111984_sftp’ to manage the file transfers with several checks for file validation using dynamic expression. The Logic app enables parallel workflow to handle both flows, one is purely dedicated to transfer within Azure, and another mainly focuses on cross-cloud transfer (AWS). Parallelly cross-cloud workflow executes the Azure Function App - UplaodtoS3 and if the data is valid, it transfers the same file from the incoming folder, decodes the content using dynamic expression, and transfers them securely to the AWS S3 bucket.

[2] https://mikestephenson.me/2022/06/29/logic-apps-with-azure-storage-sftp/

[3] https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-securing-a-logic-app?tabs=azure-portal

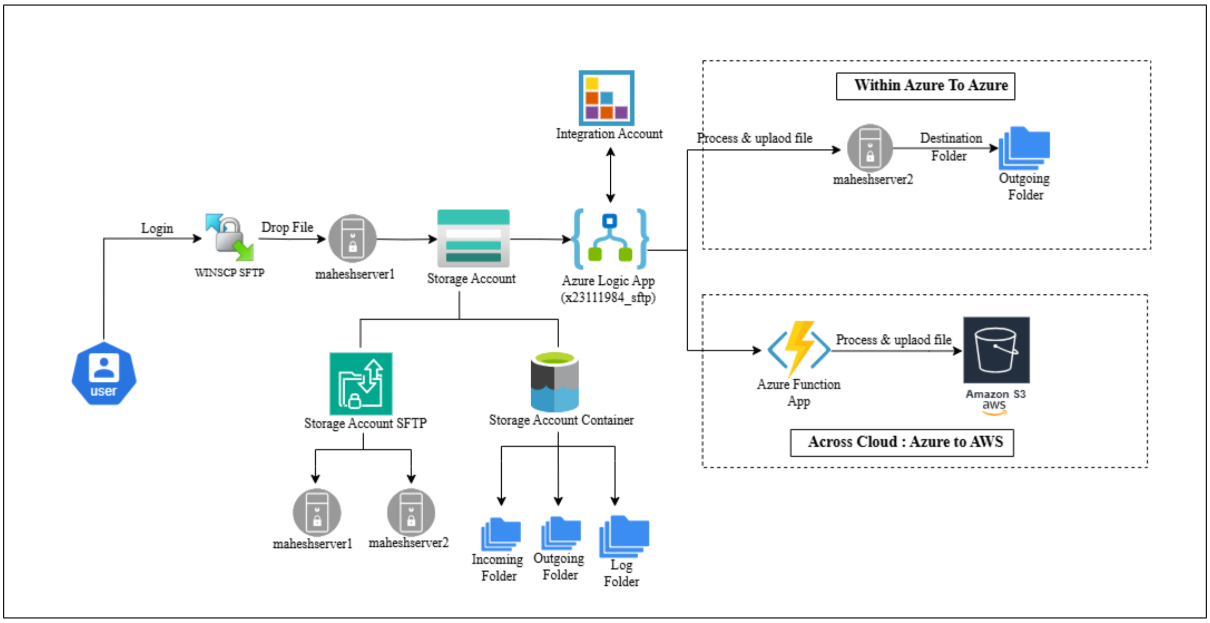


Figure 1: Architecture Diagram: Multi-Cloud Secure File Transfer

The architecture diagram mentioned in Figure 2, is an extended part of the initial workflow which highlights the log capture and analysis process. which increases the secure file transfer workflow (Azure to Azure & Azure to AWS). This approach makes sure that complete monitoring, adherence to regulation, and complex analysis of operational workflow. The architecture diagram of capturing logs starts with the Azure Log Analytics workspace under Resource Group, through which the logs generated by the secure file transfer workflow are gathered in realtime. This log gives full operational data including file transfer events, errors, and processed workflow details. An Azure Logic App called ‘x23111984_logcapture’ successfully manages the log capturing flow by gathering the logs from the Log Analytics Workspace and structuring them into structured files. The logs that have been captured are saved in two structure formats: Logs.json which is in raw format and logs.csv which is structured. Both files are present in the log folder of the Azure storage account container for the following analysis. The structured log i.e. CSV file goes through analysis via two approaches for useful analysis. The logs.csv file at first imported into Jupyter Notebook for pre-processing and applying Python libraries.
Python Libraries are used for recognizing and visualizing the pattern and pointing out the potential security threats inside the workflows. PRoM tool has been used to analyze the logs using process mining methods such as alpha mining. This stage presents a clear visualization of the workflow execution paths, points out any deviation, and optimizes the process to improve both accuracy and efficiency. Collectively, these tools help in advanced analytics and instant monitoring, which ensure a complete understanding of the operations health of the system.

This flow combines properly with the overall workflow by offering a real-time alert mechanism, log management, and audit trails that are fully ready for compliance. Primary workflow mainly Azure to Azure and Azure to AWS are having their logs captured and are undergoing continuous monitoring, making sure that they are fully configured and structured for integration with other services and tools. The information acquired from the analysis is used to make the workflow perfect, which increases the system’s accuracy and performance. Additionally, the organized logs help in thorough reporting and

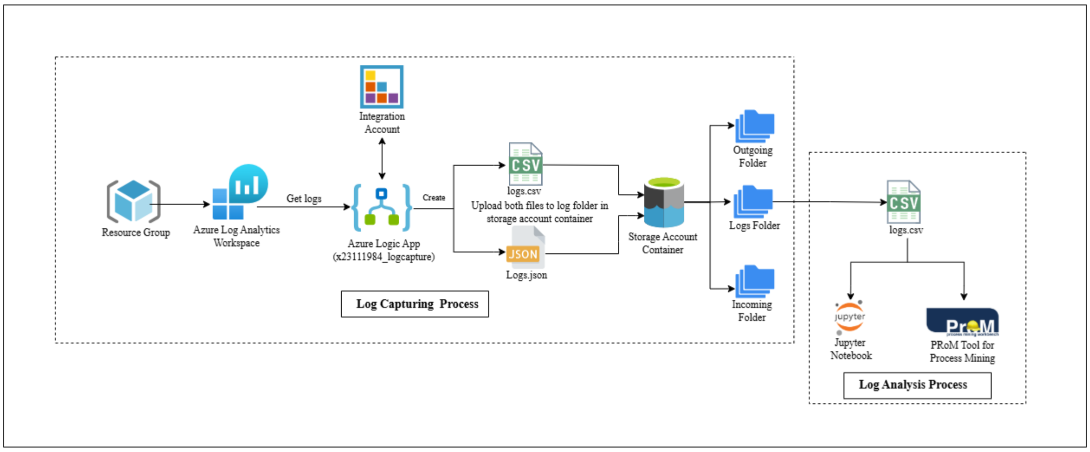


Figure 2: Log Management and Analysis Workflow Architecture

adherence to regulations, with formats like CSV and JSON giving compatibility for external audits. This combined workflow, making use of tools such as Azure monitoring, Jupyter Notebook and PROM, provides an important layer of data analysis to the secure file transfer process. It strengthens the security, improves the performance and ensures the accuracy of operations, building itself a crucial part of the overall architecture for handling and evaluating data across cloud settings.

4.2 System Specification

The implementation is all about ensuring security by using a combination of tools and technologies across different platforms.

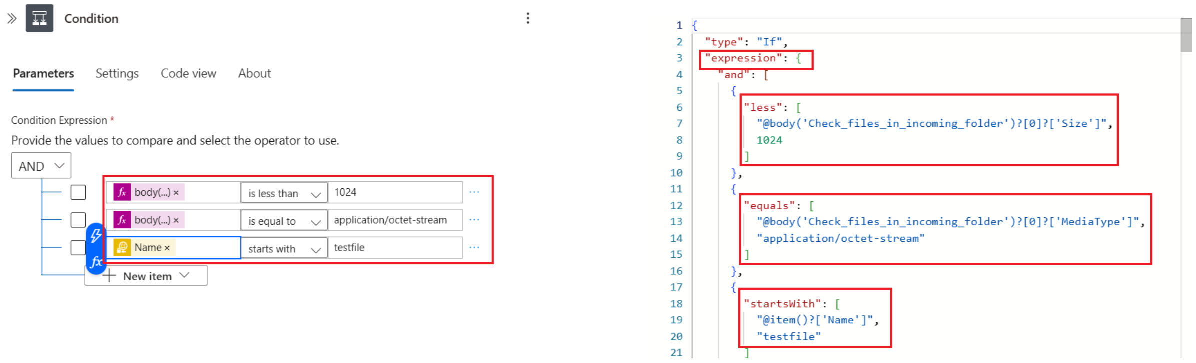
- Azure Technologies: The system uses Azure Resource Group, Azure Integration Account, Azure Logic Apps, Azure Functions, Log Analytics workspace, Azure Managed Identities, KQL and Storage Accounts to manage and analyze data flows seamlessly across services.
- AWS Integration: Used S3 bucket service from AWS for monitoring and data storage which ensures cross-cloud transfer of data is secure.
- Additional tools such as WinSCP, PRoM, and Jupyter Notebook have been used along with Programming Languages including JavaScript, DotNet, and Python libraries.

5 Implementation

The execution of the proposed approach focuses on the design and the implementation of secure and automated file transfer workflow, together with real-time log capturing and advanced log analytics through the integration of process mining. The proposed approach has been split into two main logic app workflows, first one is a logic app for secure file transfer called ‘x23111984_sftp’ and another one is logic app for log capture called ‘x23111984_logcapture’. Each step is systematically designed, making use of tools like Jupyter Notebook and PRoM tool to carry out the outputs. The complete details of the overall implementation is provided below.

5.1 Logic App: Secure File Transfer

This logic app ‘x23111984_sftp’ is implemented to automate the secure file transfer within the Azure cloud and across multi-cloud platforms. The main logic app workflow is triggered every 2 minutes by using a Recurrence Trigger action, which watches the maheshserver1 incoming folder of the Azure Storage Account. The flow of this logic app starts with the action Terminate Flow Variable, which is set to false initially. This flow processes all the files in the incoming folder one by one using For Each action and checking the conditions based on the file parameters like size of the file, media type, and filename using dynamic expression, as shown in Figure 3a and Figure 3b



Condition Expression *

Provide the values to compare and select the operator to use.

AND

body(...)	is less than	1024
body(...)	is equal to	application/octet-stream
Name	starts with	testfile

```
{
  "type": "If",
  "expression": {
    "and": [
      {
        "less": [
          "@body('Check_files_in_incoming_folder')?[0]?['Size']",
          1024
        ]
      },
      {
        "equals": [
          "@body('Check_files_in_incoming_folder')?[0]?['MediaType']",
          "application/octet-stream"
        ]
      },
      {
        "startsWith": [
          "@item()?['Name']",
          "testfile"
        ]
```

(a) Condition Action in Logic App

(b) Dynamic expressions

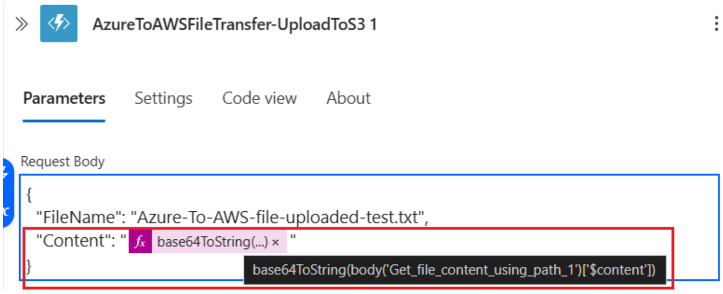
Figure 3: Pre-defined conditions verification by Dynamic expression

Files that match the specific condition for example filename start with ‘testfile’ and only proceed to the next step where the Get File Content action gathers all the file content from the maheshserver1 incoming folder and if the conditions match the processed file uploaded to maheshserver2 outgoing folder using the create file action, and during the file transfer the content is encoded for security purposes. Simultaneously, a parallel action triggered the Azure function - AzureToAWSFileTransfer-UploadToS3 which helps to enable the transfer of files across the cloud platform i.e AWS. The Azure function UploadToS3 is a very important function for enabling secure file transfer to an AWS S3 bucket, which acts as an important part of the cross-cloud file transfer system that connects Azure to AWS. This function is triggered by an HTTP POST request and has been designed for uploading the file directly to the AWS S3 bucket. The UploadToS3 function checks the AWS credentials i.e. ‘accesskey’ and ‘secretkey’ with the AWS region to ensure the secure authentication of the request. Once the request is received, the same function interprets the incoming data to get the filename and the content of the file from the JSON-formatted request body. After the validation of the data, it uses the AmazonS3Client to start and carry out the PutObjectRequest. This request helps to upload the file in S3 bucket based on the filename mentioned and content type for best compatibility.

After the successful upload, the function logged and confirmed the data with an HTTP OK response, indicating to the requester that the operation has been successful. But if the JSON formatted request body is incorrect or missing important details then the function will throw an error with HTTP BadRequest response. However, any failures noticed throughout the upload process, which involve issues related to AWS clients or

S3 services, are addressed effectively. Where all the error messages are logged and it responds with HTTP InternalserverError. These overall methods make sure to maintain the system's integrity and security, whenever potential data transmission errors occur.

The processed file content is encrypted and undergoes decoding via Base64String prior to its upload to an AWS S3 bucket with an assigned file name 'Azure-To-AWS-file-Uploaded-test.txt' as shown in Figure 4. Once the file has been successfully uploaded, a notification email has been sent to confirm the operation.



```
{
  "FileName": "Azure-To-AWS-file-uploaded-test.txt",
  "Content": " base64ToString(...) × "
}
base64ToString(body('Get_file_content_using_path_1')['$content'])
```

Figure 4: Dynamic Expression for Decoding

If the condition is not met, the action Terminate flow variable is set to True, and an alert email is sent by the system with the subject line : ALERT: File not found in the specific folder. The logic app workflow terminates with an error message: Termination due to security validation failure: error message not allowed. This complete process makes sure the secure and automated file transfer validates the file integrity and gives a real-time alert for any error that occurs.

5.2 Logic App: Log Capture

The logic app called 'x23111984_logcapture' is important for logging and monitoring activities in the Azure environment. This logic app triggered in every 2 min and get the logs from azure log analytics workspace using api. log analytics and Kusto Query Language (KQL). This query gives the details about the logic app run IDs, status, and action name from AzureDiagnostics and setup in chronological order. The workflow is shown below in Figure 5

The processed data is in JSON format, which is parsed into specific JSON format using schema to make sure that JSON is in proper structure and format as per the standard. After parsing the JSON content, the logic app triggers the action JavaScript which uses the JavaScript 'JsonToCSV' function which transforms the JSON format into a more common CSV file format. Then, the action called Create File through the logic app to create the two new files, one is in JSON format and the other in CSV format, and follows the pattern for naming convention 'log-yyyy-MM-dd-HH-mm-ss.csv', which makes sure the exact time of its creation is recorded for uniqueness and authenticity.
At the end, the created JSON and CSV files have been successfully uploaded to a destination folder within the Azure storage account named as logs folder inside the mahesh-server2 which makes sure both format are kept secure for logging and further analysis. This structure not only allows the log capture but also makes sure that the files remain accessible for further monitoring, audits, and analysis tasks.

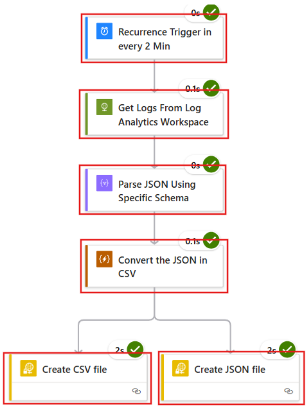
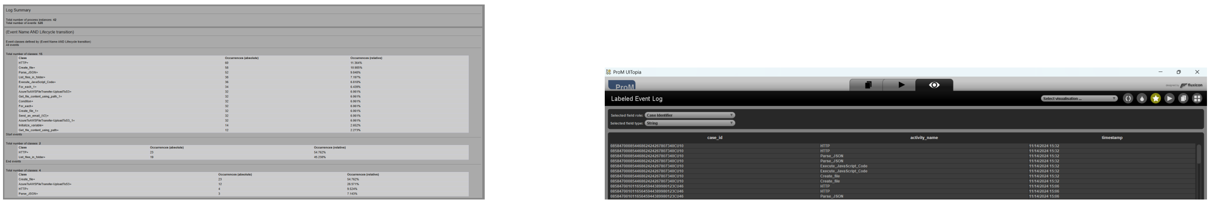

0s
Recurrence Trigger in every 2 Min
0.1s
Get Logs From Log Analytics Workspace
0s
Parse JSON Using Specific Schema
0.1s
Convert the JSON in CSV
2s
Create CSV file
2s
Create JSON file

Figure 5: Log Capture Workflow

5.3 Visualizing Log Analysis with PROM and Jupyter Tool

This part of the section sets out the way to increase data security and efficiency by refining log analysis and incorporating Visualization techniques. Once the logs are fetched and saved in the Azure container, they are available for the further analysis process. This process starts with downloading the stored logs from the Azure container, which are afterward put together for analysis. Those logs are carefully cleaned and transformed by Python libraries which ensure precise analysis. Irregularities and unwanted patterns within the logs are identified by visualizing techniques. This analysis is carried out by Jupyter Notebook. The outcome from this stage provides a detailed graph for visualization which highlights the pattern in the data and offers useful insights for security and operational concerns. After visualization, the PROM tool has been used for process mining on the event logs generated by it. This could be seen in Figure 6b This involves methods like Direct follows and Petri Net Flow Diagram to create a visual representation from event logs. This tool helps in determining the deviations and inaccuracies within the workflow which boosts the efficiency of the overall process.



Log Summary

Total number of process instances: 42
Total number of events: 528

(Event Name AND Lifecycle transition)

Event classes defined by (Event Name AND Lifecycle transition)
All events

Total number of classes: 15

Class	Occurrences (absolute)	Occurrences (relative)
HTTP+	60	11.364%
Create_file+	58	10.985%
Parse_JSON+	52	9.848%
List_files_in_folder+	38	7.197%
Execute_JavaScript_Code+	36	6.818%
For_each_1+	34	6.439%
AzureToAWSFileTransfer-UploadToS3+	32	6.061%
Get_file_content_using_path_1+	32	6.061%
Condition+	32	6.061%
For_each+	32	6.061%
Create_file_1+	32	6.061%
Send_an_email_(V2)+	32	6.061%
AzureToAWSFileTransfer-UploadToS3_1+	32	6.061%
Initialize_variable+	14	2.652%
Get_file_content_using_path+	12	2.273%

Start events

Total number of classes: 2

Class	Occurrences (absolute)	Occurrences (relative)
HTTP+	23	54.762%
List_files_in_folder+	19	45.238%

End events

Total number of classes: 4

Class	Occurrences (absolute)	Occurrences (relative)
Create_file+	23	54.762%
AzureToAWSFileTransfer-UploadToS3+	12	28.571%
HTTP+	4	9.524%
Parse_JSON+	3	7.143%

(a) Log Summary by PROM Tool

Labeled Event Log

Selected field role: Case Identifier
Selected field type: String

case_id	activity_name	timestamp
08584700085446862424267807340CU10	HTTP	11/14/2024 15:32
08584700085446862424267807340CU10	HTTP	11/14/2024 15:32
08584700085446862424267807340CU10	Parse_JSON	11/14/2024 15:32
08584700085446862424267807340CU10	Parse_JSON	11/14/2024 15:32
08584700085446862424267807340CU10	Execute_JavaScript_Code	11/14/2024 15:32
08584700085446862424267807340CU10	Execute_JavaScript_Code	11/14/2024 15:32
08584700085446862424267807340CU10	Create_file	11/14/2024 15:32
08584700085446862424267807340CU10	Create_file	11/14/2024 15:32
08584700101165645944389980123CU46	HTTP	11/14/2024 15:06
08584700101165645944389980123CU46	HTTP	11/14/2024 15:06
08584700101165645944389980123CU46	Parse_JSON	11/14/2024 15:06

(b) Labled Event Log Structure

Figure 6: Visualizing flow using PROM

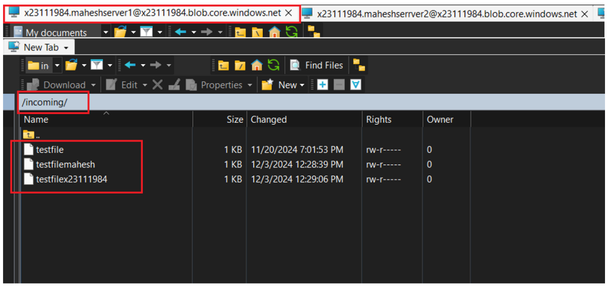
6 Evaluation

6.1 Functional Testing

6.1.1 Azure to Azure File Transfer

To evaluate the functional testing, the following steps have been executed based on the pre-defined conditions when primary Logic App (LA) has been triggered. At each step, the screenshots were taken to justify that the workflow operates as designed.

- **Drop File in Incoming Folder Using WINSCP:** File uploaded to the incoming folder for further processing.

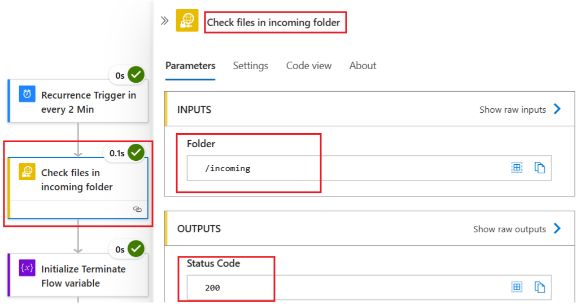

x23111984.maheshserver1@x23111984.blob.core.windows.net
x23111984.maheshserver2@x23111984.blob.core.windows.net
My documents
New Tab
in
Find Files
Download
Edit
Properties
New

/incoming/

Name	Size	Changed	Rights	Owner
..				
testfile	1 KB	11/20/2024 7:01:53 PM	rw-r-----	0
testfilemahesh	1 KB	12/3/2024 12:28:39 PM	rw-r-----	0
testfilex23111984	1 KB	12/3/2024 12:29:06 PM	rw-r-----	0

Figure 7: WINSCP Incoming Directory

- **Incoming Folder verification from Logic App:** Tracking of existing files, using logic app for realtime file verification and checks.

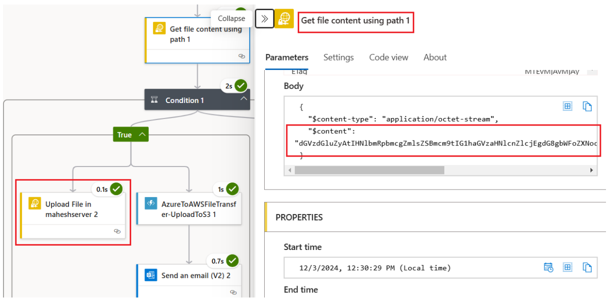

Check files in incoming folder
Parameters
Settings
Code view
About
0s
Recurrence Trigger in every 2 Min
0.1s
Check files in incoming folder
0s
Initialize Terminate Flow variable
INPUTS
Show raw inputs
OUTPUTS
Show raw outputs

Folder: /incoming

Status Code: 200

Figure 8: Check Files from Incoming Folder in LA

- **Verification of File Encryption During Processing:** Check if the file content is encrypted during the process, which makes sure that security protocols are in place during transmission.


Collapse
Get file content using path 1
2s
Condition 1
True
0.1s
Upload File in maheshserver 2
1s
AzureToAWSFileTransfer-UploadToS3 1
0.7s
Send an email (V2) 2
Get file content using path 1
Parameters
Settings
Code view
About
ETag
MTEvMjAvMjAy
PROPERTIES

Body

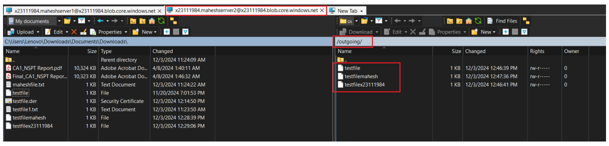
```
{
  "$content-type": "application/octet-stream",
  "$content":
"dGVzdGluZyAtIHNlbmRpbmcgZmlsZSBmcm9tIG1haGVzaHNlcnZlcjEgdG8gbWFoZXNoc
}
```

Start time: 12/3/2024, 12:30:29 PM (Local time)

End time

Figure 9: Verify Encrypted Content

- **Check Destination Outgoing Folder using WINSCP and Azure Portal:** Verify that files were sent and stored successfully in maheshserver2 storage account.


x23111984.maheshserver1@x23111984.blob.core.windows.net
x23111984.maheshserver2@x23111984.blob.core.windows.net
New Tab
My documents
Upload
Edit
Properties
New
Download
Find Files

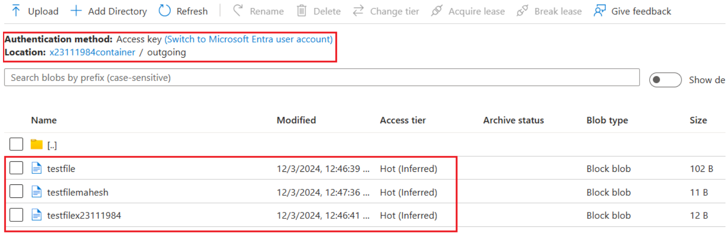
C:\Users\Lenovo\Downloads\Documents\Downloads\

Name	Size	Type	Changed
..		Parent directory	12/3/2024 11:24:09 AM
CA1_NSPT Report.pdf	10,324 KB	Adobe Acrobat Do...	4/8/2024 1:40:11 AM
Final_CA1_NSPT Repor...	10,323 KB	Adobe Acrobat Do...	4/8/2024 1:46:32 AM
maheshfile.txt	1 KB	Text Document	12/3/2024 11:24:22 AM
testfile	1 KB	File	11/20/2024 7:01:53 PM
testfile.der	1 KB	Security Certificate	12/3/2024 12:14:50 PM
testfile1.txt	1 KB	Text Document	12/3/2024 11:23:50 AM
testfilemahesh	1 KB	File	12/3/2024 12:28:39 PM
testfilex23111984	1 KB	File	12/3/2024 12:29:06 PM

/outgoing/

Name	Size	Changed	Rights	Owner
..				
testfile	1 KB	12/3/2024 12:46:39 PM	rw-r-----	0
testfilemahesh	1 KB	12/3/2024 12:47:36 PM	rw-r-----	0
testfilex23111984	1 KB	12/3/2024 12:46:41 PM	rw-r-----	0

Figure 10: WINSCP Outgoing Directory

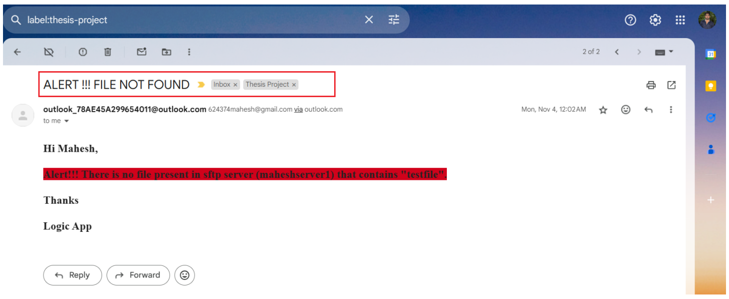


Authentication method: Access key (Switch to Microsoft Entra user account)
Location: x23111984container / outgoing

	Name	Modified	Access tier	Archive status	Blob type	Size
□	[..]					
□	testfile	12/3/2024, 12:46:39 ...	Hot (Inferred)		Block blob	102 B
□	testfilemahesh	12/3/2024, 12:47:36 ...	Hot (Inferred)		Block blob	11 B
□	testfilex23111984	12/3/2024, 12:46:41 ...	Hot (Inferred)		Block blob	12 B

Figure 11: Outgoing Directory in Storage Account

- **Check Escalation Email Notification on Condition Failure:** Verify the Logic App error response by mail notification, which shows that logic app is responsive based on its status.



ALERT !!! FILE NOT FOUND

outlook_78AE45A299654011@outlook.com 624374mahesh@gmail.com via outlook.com
to me
Mon, Nov 4, 12:02 AM

Hi Mahesh,

Alert!!! There is no file present in sftp server (maheshserver1) that contains "testfile".

Thanks

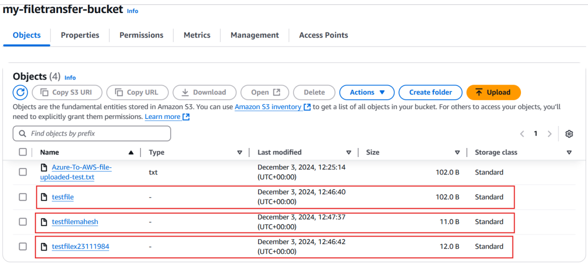
Logic App

Figure 12: Email Notification on Condition Failure

6.1.2 Azure to AWS File Transfer

To evaluate the functional testing, the primary Logic App's parallel action has been executed, i.e Azure Function action. At each step, the screenshots were taken to justify that the workflow operates as designed.

- Successfully uploaded in AWS S3 Bucket: Verifying multi-cloud transmission and handling of files, which indicates successful upload of files to an AWS S3 Bucket.

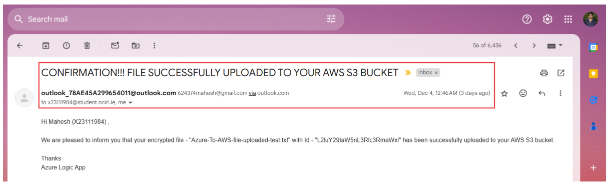


Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	Azure-To-AWS-file-uploaded-test.txt	txt	December 3, 2024, 12:25:14 (UTC+00:00)	102.0 B	Standard
□	testfile	-	December 3, 2024, 12:46:40 (UTC+00:00)	102.0 B	Standard
□	testfilemahesh	-	December 3, 2024, 12:47:37 (UTC+00:00)	11.0 B	Standard
□	testfilex23111984	-	December 3, 2024, 12:46:42 (UTC+00:00)	12.0 B	Standard

Figure 13: File Uploaded in AWS S3 Bucket

- Email confirmation on successful S3 upload: Sent confirmation email to verify the successful file processing and the upload to S3, which improves workflow transparency and monitoring.



CONFIRMATION!!! FILE SUCCESSFULLY UPLOADED TO YOUR AWS S3 BUCKET

outlook_78AE45A299654011@outlook.com 624374mahesh@gmail.com via outlook.com
to x23111984@student.ncirl.ie, me
Wed, Dec 4, 12:46 AM (3 days ago)

Hi Mahesh (X23111984) ,

We are pleased to inform you that your encrypted file - "Azure-To-AWS-file-uploaded-test.txt" with Id - "L2luY29taW5nL3Rlc3RmaWxl" has been successfully uploaded to your AWS S3 bucket.

Thanks
Azure Logic App

Figure 14: Email Confirmation on Successful S3 Upload

6.1.3 Visualization using PROM Tool and Jupyter Notebook

PRoM Tools

- **Petri Net Flow Method:** The below Figure 15 shows the source node as the beginning and the sink node as the end of the process, which helps in identifying the redundancies and inefficiency in the overall process. Conditional nodes help in decision-making that automatically direct the flow based on specific criteria which not only makes sure that various scenarios are handled but helps in maintaining the accuracy and authenticity of the process output. This graph helps with easy scalability.

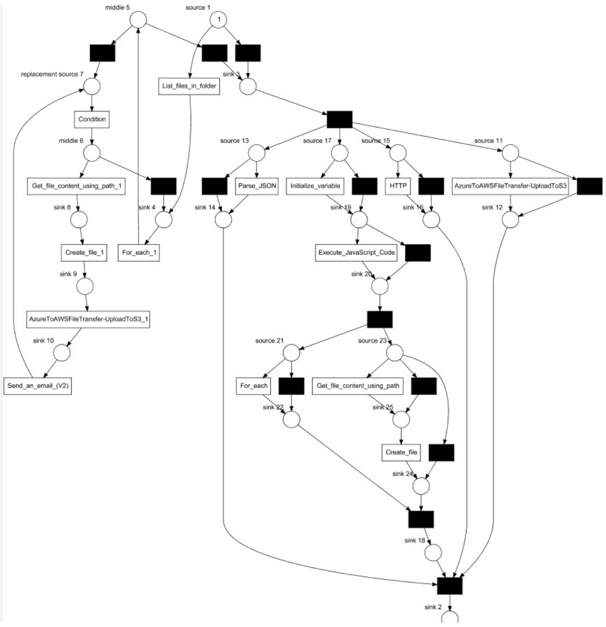


Figure 15: Process Mining – Petri Net Method

- **Direct Follow Method:** The below graph 16 has been generated by the Direct follows (minimum self-distance graph) process mining method for visualizing overall workflow. This graph helps in identifying any additional steps executed and bottlenecks in the workflow while maintaining the quality of the overall design. The below screenshot shows that the workflow operates as designed.

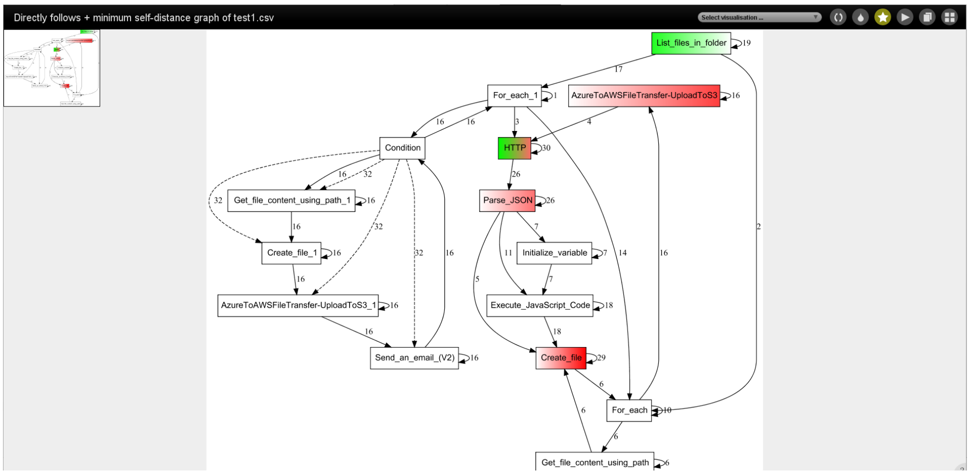

Directly follows + minimum self-distance graph of test1.csv
Select visualisation ...
List_files_in_folder 19
17
For_each_1 1
AzureToAWSFileTransfer-UploadToS3 16
16
16
3
4
Condition
HTTP 30
16
32
26
32
Get_file_content_using_path_1 16
Parse_JSON 26
16
32
7
2
Create_file_1 16
32
16
11
Initialize_variable 7
14
16
16
5
7
AzureToAWSFileTransfer-UploadToS3_1 16
Execute_JavaScript_Code 18
16
18
Send_an_email_(V2) 16
Create_file 29
6
6
For_each 10
6
Get_file_content_using_path 6

Figure 16: Process Mining – Direct Follow Method

Jupyter Notebook: In the below Graph 17: time series decomposition is shown, where the residual axis shows deviation in the patterns or unexpected spikes which represent issues like security breaches, system errors, and failure in workflow. The regular analysis of this graph could help in analyzing overall workflow performance.

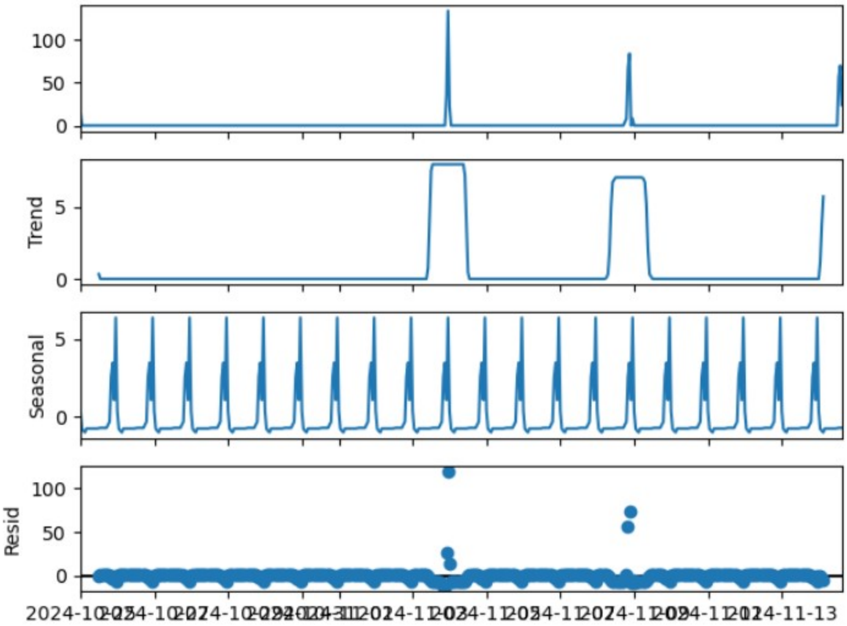


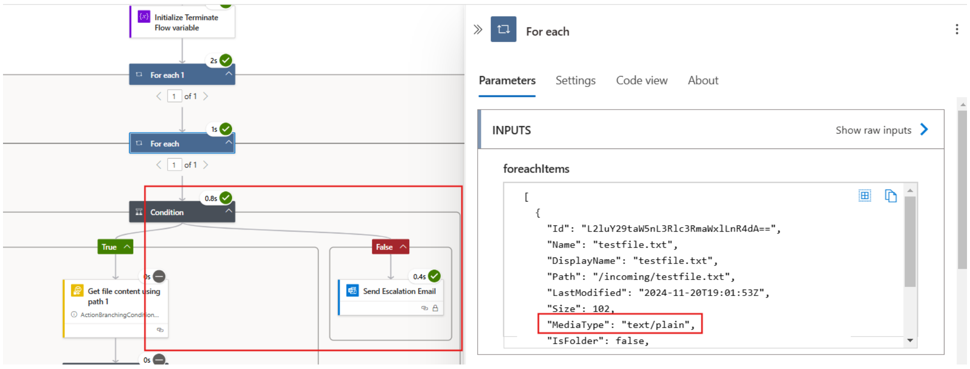
Figure 17: Time Series Decomposition Graph

6.2 Security Testing

6.2.1 Testing and validating file integrity in Cross-Platform

For security testing, the following steps are executed.

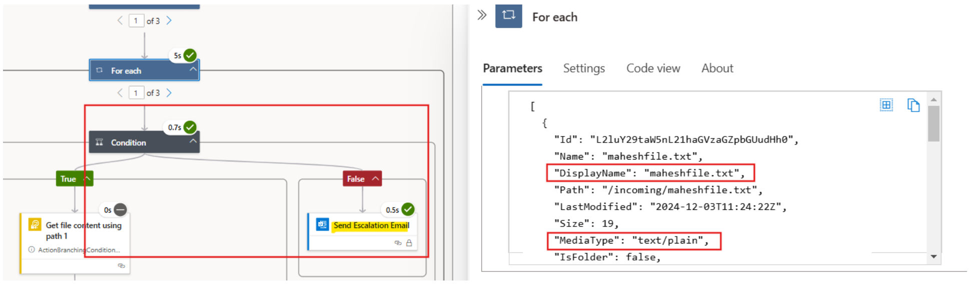
- **Testing Media Type Discrepancies:** Condition failed because of media type, expected media type is “application/octat-stream.” which is not matching. Logic app flow terminated, and sent escalation email.



```
[
  {
    "Id": "L2luY29taW5nL3Rlc3RmaWxlLnR4dA==",
    "Name": "testfile.txt",
    "DisplayName": "testfile.txt",
    "Path": "/incoming/testfile.txt",
    "LastModified": "2024-11-20T19:01:53Z",
    "Size": 102,
    "MediaType": "text/plain",
    "IsFolder": false,
```

Figure 18: Media Type Mismatch

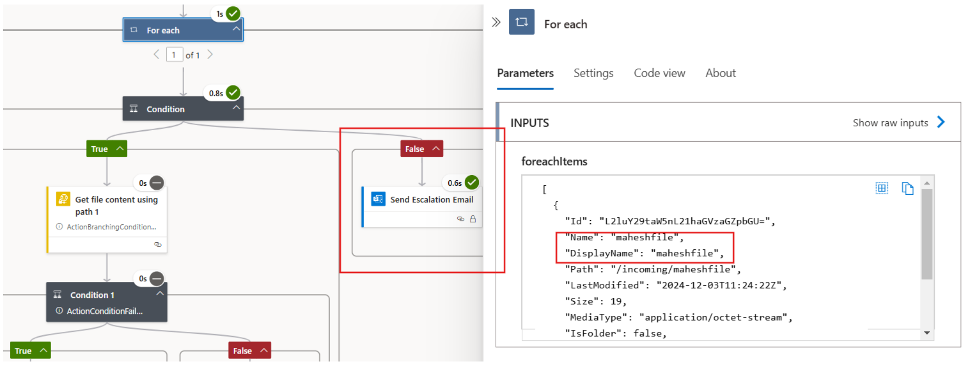
- **Testing Media Type and Filename Mismatch:** Condition failed, the expected media type is ‘application/octet-stream’ and the filename starts with ‘testfile’ which does not match. Logic app flow terminated, and sent escalation email, Figure 19



```
[
  {
    "Id": "L2luY29taW5nL21haGVzaGZpbGUudHh0",
    "Name": "maheshfile.txt",
    "DisplayName": "maheshfile.txt",
    "Path": "/incoming/maheshfile.txt",
    "LastModified": "2024-12-03T11:24:22Z",
    "Size": 19,
    "MediaType": "text/plain",
    "IsFolder": false,
```

Figure 19: Display Name and Media Type Mismatch

- **Testing Filename Mismatch:** Expected filename should start with “testfile” which does not match. Logic app flow terminated, and sent escalation email, Figure 20



```
[
  {
    "Id": "L2luY29taW5nL21haGVzaGZpbGU=",
    "Name": "maheshfile",
    "DisplayName": "maheshfile",
    "Path": "/incoming/maheshfile",
    "LastModified": "2024-12-03T11:24:22Z",
    "Size": 19,
    "MediaType": "application/octet-stream",
    "IsFolder": false,
```

Figure 20: Filename Mismatch

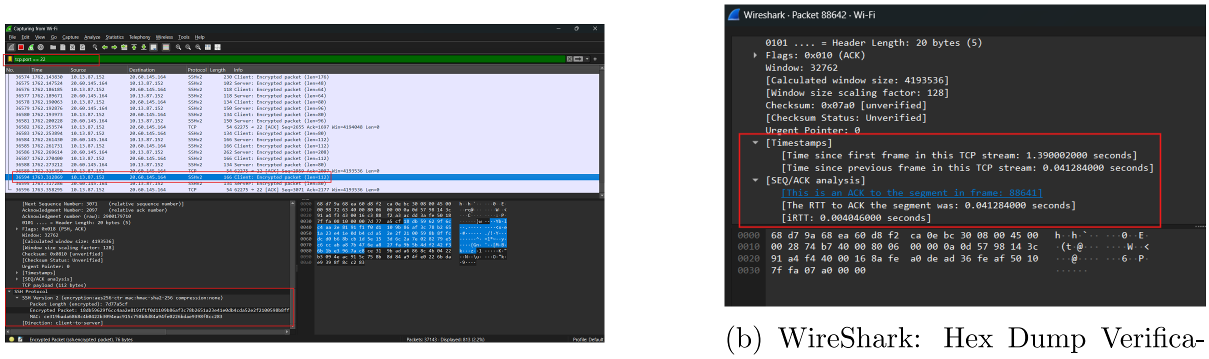
The above scenario-based testing is performed it is working as expected, and the workflow operates as designed.

6.2.2 Testing Cross-Platform File Transfer Data Security

To test the data security during the file transfer in multi-cloud, the Wireshark tool has been used to evaluate and validate the data transfer security.

Check encrypted traffic where WireShark captures the packet during the transmission to a hosted SFTP, that shows the packet is encrypted. After the verification of encryption details, it shows that individual packets are encrypted with encryption algorithms aes256-ctr. This is good from a security perspective as if someone captures these packets, they cannot understand and reconstruct the original file.
Integrity verification: The payload of the packets confirms that integrity and security have been implemented with the use of hash-based message authentication code (HMAC), which ensures the data remains unchanged during the transmission which maintains the integrity of the overall workflow. 21a
Hex Dump Verification: WireShark also shows the Hex dump 21b which confirms that the data is encrypted and holds its security.

This approach addresses that the outcome of the above component verifies that security protocols during the transmission are in place and workflow operates as designed.


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port == 22

```
No.   Time        Source         Destination    Protocol Length Info
36574 1762.143830 10.13.87.152   20.60.145.164  SSHv2    230 Client: Encrypted packet (len=176)
36575 1762.147524 20.60.145.164  10.13.87.152   SSHv2    102 Server: Encrypted packet (len=48)
36576 1762.186185 10.13.87.152   20.60.145.164  SSHv2    118 Client: Encrypted packet (len=64)
36577 1762.189671 20.60.145.164  10.13.87.152   SSHv2    118 Server: Encrypted packet (len=64)
36578 1762.190063 10.13.87.152   20.60.145.164  SSHv2    134 Client: Encrypted packet (len=80)
36579 1762.192876 20.60.145.164  10.13.87.152   SSHv2    150 Server: Encrypted packet (len=96)
36580 1762.193973 10.13.87.152   20.60.145.164  SSHv2    134 Client: Encrypted packet (len=80)
36581 1762.200228 20.60.145.164  10.13.87.152   SSHv2    150 Server: Encrypted packet (len=96)
36582 1762.253574 10.13.87.152   20.60.145.164  TCP       54 62275 → 22 [ACK] Seq=2655 Ack=1697 Win=4194048 Len=0
36583 1762.253894 10.13.87.152   20.60.145.164  SSHv2    134 Client: Encrypted packet (len=80)
36584 1762.261430 20.60.145.164  10.13.87.152   SSHv2    166 Server: Encrypted packet (len=112)
36585 1762.261731 10.13.87.152   20.60.145.164  SSHv2    166 Client: Encrypted packet (len=112)
36586 1762.269614 20.60.145.164  10.13.87.152   SSHv2    262 Server: Encrypted packet (len=208)
36587 1762.270400 10.13.87.152   20.60.145.164  SSHv2    166 Client: Encrypted packet (len=112)
36588 1762.273212 20.60.145.164  10.13.87.152   SSHv2    134 Server: Encrypted packet (len=80)
36589 1762.316450 10.13.87.152   20.60.145.164  TCP       54 62275 → 22 [ACK] Seq=2959 Ack=2097 Win=4193536 Len=0
36594 1763.312869 10.13.87.152   20.60.145.164  SSHv2    166 Client: Encrypted packet (len=112)
36595 1763.317286 20.60.145.164  10.13.87.152   SSHv2    134 Server: Encrypted packet (len=80)
36596 1763.358295 10.13.87.152   20.60.145.164  TCP       54 62275 → 22 [ACK] Seq=3071 Ack=2177 Win=4193536 Len=0
```

```
    [Next Sequence Number: 3071    (relative sequence number)]
    Acknowledgment Number: 2097    (relative ack number)
    Acknowledgment number (raw): 2900179710
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x018 (PSH, ACK)
    Window: 32762
    [Calculated window size: 4193536]
    [Window size scaling factor: 128]
    Checksum: 0x0810 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▸ [Timestamps]
  ▸ [SEQ/ACK analysis]
    TCP payload (112 bytes)
▾ SSH Protocol
  ▾ SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
      Packet Length (encrypted): 7d77a5cf
      Encrypted Packet: 18db59629f6cc4aa2e8191f1f0d1109b86af3c78b2651a23e41e0db4cda52e2f2100598b8ff
      MAC: ce319bada6868c4b0422b3094eac915c758b8d84a94fe0226bdae9398f8cc283
    [Direction: client-to-server]
```

```
0000  68 d7 9a 68 ea 60 d8 f2  ca 0e bc 30 08 00 45 00   h··h`··· ···0··E·
0010  00 98 72 63 40 00 80 06  00 00 0a 0d 57 98 14 3c   ··rc@··· ····W··<
0020  91 a4 f3 43 00 16 c3 88  f2 a3 ac dd 3a fe 50 18   ···C···· ····:·P·
0030  7f fa 08 10 00 00 7d 77  a5 cf 18 db 59 62 9f 6c   ······}w ····Yb·l
0040  c4 aa 2e 81 91 f1 f0 d1  10 9b 86 af 3c 78 b2 65   ··.····· ····<x·e
0050  1a 23 e4 1e 0d b4 cd a5  2e 2f 21 00 59 8b 8f fc   ·#······ ./!·Y···
0060  dc d0 b6 8b cb 1d 5e 15  3d 6c 2a 7e 02 82 79 e5   ······^· =l*~··y·
0070  c6 cc ab a8 7b 47 6e a8  27 fa 9b 5b 4d f2 42 f3   ····{Gn· '··[M·B·
0080  6b 1b e3 96 7a c8 ce 31  9b ad a6 86 8c 4b 04 22   k···z··1 ·····K·"
0090  b3 09 4e ac 91 5c 75 8b  8d 84 a9 4f e0 22 6b da   ··N··\u· ···O·"k·
00a0  e9 39 8f 8c c2 83                                  ·9····
```

Encrypted Packet (ssh.encrypted_packet), 76 bytes   Packets: 37143 · Displayed: 813 (2.2%)   Profile: Default

(a) WireShark: Encrypted Traffic Analysis

```
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x010 (ACK)
    Window: 32762
    [Calculated window size: 4193536]
    [Window size scaling factor: 128]
    Checksum: 0x07a0 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  ▾ [Timestamps]
      [Time since first frame in this TCP stream: 1.390002000 seconds]
      [Time since previous frame in this TCP stream: 0.041284000 seconds]
  ▾ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 88641]
      [The RTT to ACK the segment was: 0.041284000 seconds]
      [iRTT: 0.004046000 seconds]
```

```
0000  68 d7 9a 68 ea 60 d8 f2  ca 0e bc 30 08 00 45 00   h··h`··· ···0··E·
0010  00 28 74 b7 40 00 80 06  00 00 0a 0d 57 98 14 3c   ·(t·@··· ····W··<
0020  91 a4 f4 40 00 16 8a fe  a0 de ad 36 fe af 50 10   ···@···· ···6··P·
0030  7f fa 07 a0 00 00                                  ······
```

(b) WireShark: Hex Dump Verification

Figure 21: Combined Figure with Subfigure Numbering

6.3 Discussion

The primary Logic App, which handles the overall workflow of the proposed system, only triggered for certain files based on requirements. This ensures the security and authenticity of the process which reduces the risk caused by cyberattacks like Cross-Site Scripting (XSS) and Command Injection by executing the javascript code and command line scripts to manipulate the filename which will affect the overall solution.

For the proposed solution, the integration of secure file transfer protocol makes sure that transmission of data is encrypted, which reduces the risk of manipulation in workflow. This combination not only prevents data but also improves the overall workflow efficiency and operations and includes end-to-end encryption and secure protocols like SFTP which prevent the data in transit and at rest. Also increases the transparency of the flow by integrating process mining tool which provides various details for visualization regarding the workflow. In the proposed solution an upgrade may be needed when integrating with realtime processing approaches like integrating process mining tool with enterprise version and more.

7 Conclusion and Future Work

This research is able to explore the area of developing and implementing secure, automated file transfers utilizing log analysis and also employing process mining to optimize the workflow across multi-cloud platforms such as AWS and Azure. Through the overall

implementation of a secure file transfer protocol with log capturing and process mining methods, the primary research question has been effectively addressed. Also used Python libraries to visualize and identify deviations in the primary workflow, and this whole process is end-to-end encrypted. According to the proposal, an automated file transfer system ensuring security and authenticity is developed. The Azure Logic Apps and Function App ensure efficient and secure file transfer in multi-cloud and single-cloud environments, proving that the demonstrated workflow successfully transfers and handles files securely. Also provided a hybrid cloud environment, which is not commonly available for transferring files through SFTP.
Current results generated from the process mining tool could be made better using the Enterprise Version of tools. This study has only addressed small-scale multi-cloud environments, but large-scale multi-cloud environments may come with their own set of complexities that have not been tested due to limited availability of resources and time. In future, Terraform and Ansible could be implemented for automation. The recurring action of the logic app could be replaced with the functionality where the solution automatically runs when files are detected in a specific folder. Also, the security of files and their payload could be increased by implementing public and private key functionality. Also, as manual files are transferred to Jupyter Notebook or any process mining tools, this could be directly integrated with Logic App which would omit the manual process and save time increasing workflow efficiency.

References

AlZain, M. A., Pardede, E., Soh, B. and Thom, J. A. (2012). Cloud computing security: from single to multi-clouds, *2012 45th Hawaii International Conference on System Sciences*, IEEE, pp. 5490–5499.

Ansar, M., Ashraf, M. W. and Fatima, M. (2018). Data migration in cloud: A systematic review, *Am. Sci. Res. J. Eng. Technol. Sci.(ASRJETS)* **48**(1): 73–89.

El-Gharib, N. M. and Amyot, D. (2019). Process mining for cloud-based applications: A systematic literature review, *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, IEEE, pp. 34–43.

Executions, D. A. P. (n.d.). Process mining and security: Detecting anomalous process executions and checking process conformance.

Islam, M. S., Pourmajidi, W., Zhang, L., Steinbacher, J., Erwin, T. and Miranskyy, A. (2021). Anomaly detection in a large-scale cloud platform, *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, IEEE, pp. 150–159.

Kanungo, S. (2023). Security challenges and solutions in multi-cloud environments, *Stochastic Modelling and Computational Sciences* **3**(2): 139–146.

Khalil, I. M., Khreishah, A. and Azeem, M. (2014). Cloud computing security: A survey, *Computers* **3**(1): 1–35.

Lata, S. and Singh, D. (2022). Intrusion detection system in cloud environment: Literature survey & future research directions, *International Journal of Information Management Data Insights* **2**(2): 100134.

Liu, W. (2012). Research on cloud computing security problem and strategy, *2012 2nd international conference on consumer electronics, communications and networks (CECNet)*, IEEE, pp. 1216–1219.

Mohammad, N. (n.d.). Enhancing security and privacy in multi-cloud environments: A comprehensive study on encryption techniques and access control mechanisms, *International Journal of Computer Engineering and Technology (IJCET)* **12**: 51–63.

Nicolas, N. M. B. and Roger, A. E. (2022). An overview of intrusion detection within an information system: The improvment by process mining., *Netw. Commun. Technol.* **7**(1): 55–60.

Ristic, I. (2013). *Openssl cookbook: A guide to the most frequently used openssl features and commands*, Feisty Duck.

Skrifvars, M. (2022). Cloud migration to azure logic apps: A case study using the cloudstep decision process.

Varalakshmi, K., Dharma Prakash, V. and Noble Lourdhu, D. H. (2023). Multipurpose file transfer and file inquiry, *Journal of Science, Computing and Engineering Research* **6**(4): 90–96.

Vervaet, A. (2021). Monilog: An automated log-based anomaly detection system for cloud computing infrastructures, *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, IEEE, pp. 2739–2743.

Wilhelms, J. (2024). Azure sandbox manager: enhancing efficiency and reducing costs.

Zhang, X., Cui, L., Shen, W., Zeng, J., Du, L., He, H. and Cheng, L. (2023). File processing security detection in multi-cloud environments: a process mining approach, *Journal of Cloud Computing* **12**(1): 100.

Zhou, P., Wang, Y., Li, Z., Wang, X., Tyson, G. and Xie, G. (2020). Logsayer: Log pattern-driven cloud component anomaly diagnosis with machine learning, *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, IEEE, pp. 1–10.