Politecnico di Torino

GRENOBLE INP UGA Ensimag

amadeus

Global Security Operations Center

Politecnico Di Torino - Ensimag

Master of Science in Cybersecurity - Diplome D'ingénieur

A.y. 2024/2025

SECURING THE CLOUD

Design and Implementation of a Security Blueprint for Amadeus Mergers and Acquisitions

Supervisors:

Prof. Marco Mellia
Prof. Sebastien Viardot
Ing. Jeremy Van De Woestyne

Candidate:

Yann Freddy Dongue Dongmo

Abstract

This thesis presents the design and implementation of a scalable and replicable security blueprint aimed at enhancing cloud security operations within Amadeus, specifically in the Global Security Operations Center (GSOC). The initiative responds to the increasing complexity of managing security across multiple cloud environments resulting from mergers and acquisitions, each with distinct infrastructures and configurations.

The project focuses on integrating Cloud-Native Application Protection Platforms (CNAPP) and Security Information and Event Management (SIEM) systems into Amadeus's AWS ecosystem. A comprehensive proof of concept was developed to demonstrate the feasibility of centralized log collection, secure storage, and real-time ingestion into Splunk and Prisma Cloud. The architecture emphasizes confidentiality, integrity, and availability of logs, while also addressing cost optimization and operational scalability.

Several deployment methodologies were evaluated, including manual configuration, scripting, and infrastructure-as-code approaches. CloudFormation, the AWS infrastructure-as-code tool was selected for its robustness, security compliance, and maintainability. The blueprint also includes a decentralized strategy for syslog integration, ensuring secure transmission of system logs from virtual machines to the SIEM platform.

Extensive testing validated the architecture's effectiveness in log collection, policy enforcement, and cross-account visibility. The project further explores the environmental and societal implications of large-scale cloud logging, proposing strategies to mitigate resource consumption and ensure regulatory compliance.

This work contributes to the strategic goal of standardizing cloud security practices across Amadeus, enabling proactive threat detection, streamlined incident response, and alignment with international security standards. It also reflects my technical growth and understanding of cloud security architecture within a real-world enterprise context.

Acknowledgements

To those who gave me everything before I even began,
my parents, Thomas Akafak Zengo and Yvonne Wamba.
Your strength, your sacrifices, your quiet faith
are the foundation upon which this work stands.
Nothing here would exist without you.

To mum Jacky and mum Mado,
my mothers in Italy,
for your warmth, your care, and your open arms
in a land that became home because of you.

To my family, near and far, across Europe and Cameroon,
your love traveled every distance,
steady, silent, yet always present.

To the Sounna family,
for your unwavering presence,
for your support in moments of doubt,
and for the warmth that carried me forward.
You were a constant when everything else was uncertain.

To my mentor, Eng. Yannick Azeugo,
for your guidance, your patience,
and for keeping showing me what it means
to grow with purpose and discipline.

To my friends,
for the laughter, the late nights, and the moments in between,
for believing in me even when I doubted myself,
and for making this journey lighter than it would have been alone.

And to all those who stood beside me
in words, in silence, in belief,
this journey was never mine alone,
but ours.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context and Motivation

Amadeus is a company under Spanish law created in 1987 by four European airlines (Air France, Iberia, Lufthansa and the Scandinavian Airlines System) with the aim of creating a common structure for the computerized distribution of air segments and then bringing together the entire air ticket sales offer, all airlines combined. Amadeus enables airlines and travel agencies to manage and distribute flights globally. It supports Altéa Reservation airlines with full distribution tools and "non-Altéa" airlines with schedules, seat availability, and real-time fares. The Altéa IT suite also handles booking, inventory, and departure control systems.

The Global Security Operations Center (GSOC) at Amadeus is a critical hub dedicated to safeguarding the organization's IT infrastructure, systems, and data. Operating 24/7, the GSoC is responsible for monitoring, detecting, and responding to potential cybersecurity threats across Amadeus's global network. It leverages advanced security technologies, threat intelligence, and real-time analytics to ensure the integrity and availability of Amadeus's systems, which are vital to the airline and travel industries. The GSoC collaborates closely with other departments to implement robust security measures, mitigate risks, and ensure compliance with international security standards, providing a secure environment for Amadeus's clients and partners worldwide.

Given that Amadeus operates in airport IT and travel distribution (2 business units of Amadeus), there are often acquisitions within these business units. These acquisitions usually come with their own specificities and their own infrastructure, and since Amazon Web Services (AWS) is widely used, there is a good chance that it is part of what they rely on. As a member of the GSoC, I joined the Cloud Architecture team to design and implement a proof of concept for a full scalable and repeatable security blueprint that will be used to ensure that each

new cloud environment will be covered quickly by the GSOC. This project is vital to enhance Amadeus's cloud security capabilities by enabling proactive threat detection, automated responses, and ensuring compliance with industry security standards.

1.2 Contribution

The objective of this thesis is to design and implement a scalable and standardized security framework for cloud environments within Amadeus, with a particular focus on supporting the Global Security Operations Center (GSOC) in managing multi-cloud infrastructures.

This work addresses the challenges of securing heterogeneous cloud environments resulting from mergers and acquisitions, where each newly integrated entity may have its own architecture, configurations, and security practices. In this context, ensuring rapid onboarding, centralized visibility, and consistent security policies becomes essential.

The main contribution of this thesis is the development of a security blueprint that enables the efficient integration and monitoring of cloud environments. This blueprint is designed to be both scalable and reproducible, allowing it to be applied across multiple accounts and infrastructures with minimal manual intervention.

More specifically, the contributions of this work are as follows:

- **Design of a cloud security architecture:** A comprehensive architecture is proposed to ensure secure log collection, storage, and analysis across multiple AWS accounts. The architecture emphasizes confidentiality, integrity, and availability of security logs.
- **Integration of CNAPP and SIEM solutions:** The work demonstrates how Cloud-Native Application Protection Platforms (CNAPP), such as Prisma Cloud, can be integrated with Security Information and Event Management (SIEM) systems like Splunk to provide centralized threat detection and compliance monitoring.
- **Implementation of a centralized logging strategy:** A scalable logging pipeline is developed to collect, store, and forward logs from distributed cloud environments. This includes the management of audit logs, flow logs, and system logs, ensuring real-time visibility and analysis.
- **Evaluation of deployment methodologies:** Different deployment approaches are analyzed, including manual configuration, scripting, and infrastructure-as-code solutions. This evaluation highlights their advantages and limitations in terms of scalability, maintainability, and security.

- **Adoption of Infrastructure-as-Code (IaC):** AWS CloudFormation is selected and implemented as the primary deployment strategy, enabling automation, reproducibility, and consistency across cloud environments.

- **Proof of concept and validation:** A complete proof of concept is developed to validate the proposed architecture. Extensive testing demonstrates its effectiveness in log ingestion, policy enforcement, and cross-account visibility.

Overall, this work contributes to improving cloud security operations within Amadeus by providing a structured and scalable approach to monitoring, securing, and managing cloud infrastructures in a multi-cloud context.

1.3 Research Questions

In order to guide the development and evaluation of this work, a set of research questions is defined. These questions focus on the challenges of securing multi-cloud environments and on the effectiveness of the proposed security architecture. They provide a structured framework for both the design of the solution and the interpretation of the results.

Research Question 1
How can a scalable and standardized security architecture be designed to efficiently onboard and secure multiple cloud environments in a multi-cloud context?

As organizations adopt multi-cloud strategies and integrate new infrastructures through mergers and acquisitions, they face significant challenges in maintaining consistent security practices. A standardized architecture is required to ensure rapid onboarding, reduce configuration errors, and enforce uniform security policies across heterogeneous environments.

Research Question 2
How can centralized log collection and monitoring be implemented to ensure real-time threat detection while maintaining scalability and cost efficiency?

Cloud environments generate large volumes of logs from multiple sources, including audit logs, network traffic, and system events. Efficiently collecting, storing, and analyzing these logs is essential for detecting security threats. However, this must be achieved without introducing excessive costs or performance bottlenecks, making scalability a key concern.

Research Question 3
What is the most effective deployment strategy to ensure reproducibility, maintainability, and compliance in cloud security architectures?

Deploying security solutions across multiple cloud environments requires a consistent and repeatable approach. Manual configurations are error-prone and difficult

to maintain, especially at scale. This raises the need to evaluate automated deployment strategies, such as infrastructure-as-code, to ensure reliability, compliance with security standards, and long-term maintainability.

1.4 Thesis Structure

The remainder of this thesis is organized as follows.

Chapter 2 provides the background and context necessary to understand the problem addressed in this work. It introduces cloud computing concepts, multi-cloud architectures, and the main security challenges associated with distributed environments. It also reviews existing solutions and related work in cloud security, including CNAPP and SIEM technologies.

Chapter 3 presents the proposed security architecture and methodology. It details the design of the cloud security framework, including log collection mechanisms, integration of security tools, and the overall system architecture.

Chapter 4 describes the implementation of the proposed solution. It focuses on the deployment strategy using Infrastructure-as-Code, particularly AWS CloudFormation, and explains how the architecture is applied in practice across multiple cloud environments.

Chapter 5 discusses the results and evaluation of the proposed approach. It analyzes the effectiveness of the solution in terms of scalability, security, and operational efficiency, and highlights the main challenges encountered during the project.

Finally, Chapter 6 concludes the thesis by summarizing the main contributions of the work and outlining possible directions for future improvements.

Chapter 2

Background

This chapter presents the fundamental concepts and technologies required to understand the problem addressed in this thesis. As the work focuses on securing cloud-based infrastructures in a multi-cloud context, it is essential to first introduce the key principles of cloud computing and the associated security challenges. The chapter begins with an overview of cloud computing models, including service models and deployment models, highlighting their advantages and limitations. It then explores the architectural characteristics of modern cloud environments, with a particular focus on multi-account and multi-cloud configurations. Subsequently, the chapter discusses the main security challenges in cloud environments, such as misconfigurations, lack of visibility, and the complexity of managing distributed systems. Special attention is given to logging, monitoring, and threat detection, which are central to effective security operations. In addition, this chapter reviews the state of the art cloud security solutions currently adopted in industry and research. This includes an analysis of modern security approaches and platforms, such as Cloud-Native Application Protection Platforms (CNAPP), Security Information and Event Management (SIEM) systems, and other relevant tools used to ensure visibility, compliance, and threat detection in cloud environments. Finally, the chapter introduces Infrastructure-as-Code (IaC) as a key enabler for automation, scalability, and reproducibility in cloud deployments. These concepts provide the foundation for the design and implementation of the proposed solution presented in the following chapters.

2.1 Cloud Computing

Cloud computing refers to the on-demand delivery of computing resources, such as virtual machines, storage, networking, and software services, over the internet following a pay-per-use pricing model. Instead of relying on locally

managed infrastructure, users access remote resources hosted in large-scale data centers operated by cloud service providers. These providers ensure the availability, maintenance, and scalability of the underlying infrastructure, allowing users to focus on their applications and services.

At its core, cloud computing abstracts the complexity of physical hardware by leveraging virtualization technologies. This abstraction enables multiple users to share the same physical resources while maintaining isolation and security. As a result, resources can be dynamically allocated and released based on demand, providing a high degree of elasticity and efficient resource utilization.

One of the key advantages of cloud computing lies in its ability to provide scalability and flexibility. Organizations can rapidly provision resources to accommodate workload fluctuations without the need for significant upfront investments in hardware. This contrasts with traditional on-premises infrastructures, where capacity planning must anticipate peak demand, often leading to over-provisioning or underutilization of resources. In cloud environments, resources can be scaled horizontally or vertically, enabling systems to adapt efficiently to changing requirements.

Cloud computing also introduces a shift from capital expenditure (CapEx) to operational expenditure (OpEx). Instead of purchasing and maintaining physical infrastructure, organizations pay only for the resources they consume. This economic model reduces financial barriers to entry and allows businesses to innovate more rapidly by experimenting with new technologies and services without substantial initial costs.

In addition to its economic and operational benefits, cloud computing plays a central role in modern digital ecosystems. It supports a wide range of applications, including enterprise systems, big data processing, artificial intelligence, and high-performance computing. Furthermore, the global availability of cloud services enables distributed systems and facilitates collaboration across geographically dispersed teams.

Cloud computing has therefore become a fundamental enabler of digital transformation across industries. Organizations of all sizes, from small startups to large enterprises such as Amadeus, rely on cloud technologies to improve agility, accelerate development cycles, and deliver scalable and resilient services. This widespread adoption, however, also introduces new challenges, particularly in terms of security, governance, and system complexity, which are addressed in the following sections.

2.1.1 Cloud Deployment Models

Cloud computing environments can be deployed using different models, depending on the level of control, ownership, and sharing of resources. These deployment models define how cloud infrastructures are organized and accessed, and they play

a critical role in determining the trade-offs between scalability, security, cost, and operational complexity.

Selecting an appropriate deployment model is a key decision for organizations, as it directly impacts how data is managed, how applications are deployed, and how security policies are enforced. Factors such as regulatory requirements, workload sensitivity, performance constraints, and budget considerations must be taken into account when choosing a cloud strategy.

Four main cloud deployment models are commonly identified: public cloud, private cloud, hybrid cloud, and community cloud.

Public Cloud

The public cloud is owned and operated by third-party cloud service providers, such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform. In this model, all computing resources, including servers, storage systems, and networking components, are managed by the provider and delivered over the internet.

Public cloud environments rely on a multi-tenant architecture, where multiple customers share the same physical infrastructure while remaining logically isolated. This approach allows providers to achieve economies of scale, resulting in reduced costs and highly elastic resource provisioning.

One of the main advantages of the public cloud is its scalability. Organizations can quickly provision or release resources based on demand, enabling efficient handling of variable workloads. Additionally, the pay-per-use pricing model allows companies to optimize costs by paying only for the resources they consume.

However, the public cloud model also introduces certain challenges. Since resources are shared, organizations must rely on the provider's security mechanisms and ensure proper configuration of their services. Misconfigurations, lack of visibility, and compliance constraints are among the most common risks associated with public cloud environments.

Private Cloud

A private cloud consists of computing resources dedicated exclusively to a single organization. It can be deployed on-premises within the organization's own data center or hosted externally by a third-party provider. In both cases, the infrastructure is not shared with other tenants, providing a higher degree of isolation.

Private cloud environments offer enhanced control over data, infrastructure, and security policies. This makes them particularly suitable for organizations handling sensitive information or operating under strict regulatory frameworks, such as those in the healthcare, financial, or governmental sectors.

In addition, private clouds allow for greater customization of infrastructure and services, enabling organizations to tailor their environments to specific operational requirements. However, this level of control comes at the cost of increased complexity and higher operational expenses, as the organization is responsible for managing and maintaining the infrastructure.

Hybrid Cloud

The hybrid cloud model combines public and private cloud environments, enabling data and applications to be shared between them. This integration allows organizations to take advantage of the scalability and cost efficiency of the public cloud while maintaining sensitive workloads in a more controlled private environment.

Hybrid cloud architectures support use cases such as workload migration, data replication, and disaster recovery. For example, critical data can be stored in a private cloud, while computationally intensive tasks are offloaded to the public cloud. This approach allows organizations to optimize performance and resource utilization.

One of the key benefits of hybrid cloud is its flexibility. It provides multiple deployment options and allows organizations to adapt their infrastructure to changing business needs. However, managing hybrid environments can be complex, as it requires ensuring consistent security policies, identity management, and data governance across multiple platforms.

Community Cloud

A community cloud is a shared infrastructure used by a group of organizations that have similar requirements, such as regulatory constraints, security policies, or operational objectives. The infrastructure may be jointly owned and managed by the participating organizations or provided by a third-party service provider.

This model enables organizations to share costs while maintaining a level of control and customization that is not typically available in public cloud environments. It is particularly relevant in sectors where collaboration and compliance are essential, such as healthcare, research institutions, or government agencies.

Although community clouds offer advantages in terms of collaboration and cost sharing, they may also introduce challenges related to governance, resource allocation, and coordination among participating entities.

Overall, each deployment model presents specific advantages and limitations. In practice, many modern organizations adopt hybrid or multi-cloud strategies, combining different deployment models to meet their technical, operational, and security requirements. As shown in the table below, each deployment model presents different trade-offs.

Table 2.1: Types of Cloud Computing

Cloud Type	Definition	Advantages	Disadvantages
Public Cloud	Cloud services offered over the internet and shared across multiple organizations	Cost-effective, no maintenance, high reliability	Lower security, limited customization
Private Cloud	Cloud infrastructure dedicated to a single organization, either on-premises or hosted by a provider	Enhanced security and privacy, customizable	Higher cost, limited scalability
Hybrid Cloud	Combination of public and private clouds with integration between them	Flexibility, optimized resource usage	Complex management
Community Cloud	Shared infrastructure among organizations with common requirements	Collaboration, shared cost	Less common, less flexible than private cloud

2.1.2 Cloud Service Models

Cloud computing services can be categorized into different models depending on the level of abstraction and the responsibilities assigned to the cloud provider and the user. These models define how computing resources are delivered and managed, and they play a critical role in shaping the architecture, scalability, and security of cloud-based systems.

The most widely recognized cloud service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Serverless Computing, and Software as a Service (SaaS). These models are often described as forming a layered stack, where each level builds upon the capabilities of the previous one while abstracting additional aspects of infrastructure management.

Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) represents the most fundamental layer of cloud computing services. In this model, cloud providers offer virtualized computing resources such as virtual machines, storage, and networking components over

the internet. Users retain control over the operating systems, applications, and configurations, while the provider manages the underlying physical infrastructure.

IaaS provides a high degree of flexibility and scalability, allowing organizations to provision and manage resources dynamically according to their needs. It is particularly suitable for workloads requiring full control over the environment, such as custom applications, legacy systems, or complex architectures. However, this level of control also implies greater responsibility for system configuration, maintenance, and security management.

Platform as a Service (PaaS)

Platform as a Service (PaaS) provides a managed environment for developing, testing, deploying, and maintaining applications. In this model, the cloud provider abstracts the underlying infrastructure, including servers, storage, networking, and runtime environments, allowing developers to focus on application logic rather than system management.

PaaS solutions offer pre-configured development frameworks, tools, and services that accelerate the application development lifecycle. This leads to increased productivity and reduced time-to-market. Typical examples include services such as Microsoft Azure App Service and AWS Elastic Beanstalk. However, the abstraction of infrastructure may limit customization and reduce control over the execution environment.

Serverless Computing

Serverless computing, often considered an extension of PaaS, further abstracts infrastructure management by eliminating the need to provision or manage servers entirely. In this model, applications are executed in response to specific events, and resources are automatically allocated by the cloud provider as needed.

Serverless architectures are inherently scalable and follow a fine-grained, event-driven execution model. This enables efficient resource utilization, as computing resources are consumed only when functions are executed. As a result, serverless computing can significantly reduce operational overhead and costs.

Despite these advantages, serverless models introduce new challenges, such as increased dependency on provider-specific services, potential latency issues, and complexity in monitoring and debugging distributed functions.

Software as a Service (SaaS)

Software as a Service (SaaS) represents the highest level of abstraction in the cloud service model stack. In this model, fully functional applications are delivered over

the internet, and users access them through web interfaces without managing any underlying infrastructure or platform components.

The cloud provider is responsible for maintaining the application, including updates, security patches, and availability. This model offers ease of use, rapid deployment, and minimal operational overhead. Common examples of SaaS applications include web-based email services, collaboration tools, and cloud security platforms such as Prisma Cloud.

However, SaaS solutions offer limited customization and control compared to lower-level service models. Organizations must also consider data privacy, vendor lock-in, and integration challenges when adopting SaaS-based solutions.

Overall, these service models provide different levels of control, flexibility, and responsibility. The choice of an appropriate model depends on the specific requirements of the organization, including performance, scalability, security, and operational constraints. As illustrated in the figure below, cloud service models can be viewed as a layered stack, where each layer abstracts additional infrastructure complexity.

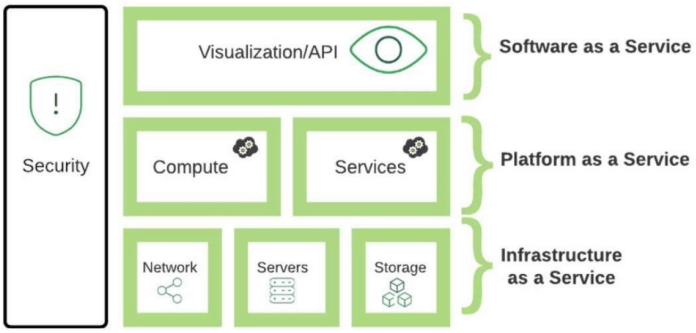

Figure 2.1: Cloud Computing Stack

The following table summarizes the key differences between the main cloud service models.

2.1.3 Shared Responsibility Model

As organizations increasingly migrate their workloads to cloud environments, understanding the distribution of security responsibilities becomes a fundamental requirement. This concept is formalized through the shared responsibility model,

Table 2.2: Types of Cloud Service Models

Cloud Model	**Description**	**Key Features**	**Examples**
IaaS	Fundamental cloud service for renting IT infrastructure such as servers, storage, and networking resources.	Pay-as-you-go, scalable, full control over infrastructure	AWS EC2, Azure Blob Storage
PaaS	Provides a managed environment for developing, testing, and deploying applications without managing underlying infrastructure.	Pre-configured environments, faster development	AWS Elastic Beanstalk, Google App Engine
SaaS	Delivers ready-to-use software applications over the internet on a subscription basis.	No infrastructure management, automatic updates, web access	Gmail, Microsoft 365, Prisma Cloud
Serverless Computing	Focuses on executing applications in response to events without manual infrastructure management.	Auto-scaling, event-driven, pay-per-execution	AWS Lambda, Azure Functions

which defines how security duties are divided between the cloud service provider and the customer.

In traditional on-premises infrastructures, organizations are responsible for securing the entire technology stack, including physical infrastructure, networking, operating systems, and applications. However, with the adoption of cloud computing, part of these responsibilities is transferred to the cloud provider. The extent of this transfer depends on the service model used, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS).

The shared responsibility model distinguishes between two main domains: *security of the cloud* and *security in the cloud*. Cloud providers are responsible for the security *of* the cloud, which includes the protection of physical data centers, hardware, networking infrastructure, and foundational services. On the other hand, customers are responsible for security *in* the cloud, which includes the protection of their data, applications, configurations, and access management.

The distribution of responsibilities varies depending on the level of abstraction offered by the service model. In an IaaS environment, customers retain significant control over operating systems, network configurations, and deployed applications, and therefore bear a larger share of the security responsibilities. In PaaS, the provider manages additional layers such as runtime environments and middleware, reducing the operational burden on the customer. In SaaS, most components are managed by the provider, while the customer focuses primarily on data protection and identity and access management.

It is important to note that certain responsibilities remain with the customer regardless of the service model. In particular, organizations are always responsible for protecting their data, managing user identities, and enforcing access control policies. These aspects are critical, as they are often the primary targets of cyberattacks.

Misunderstanding or misapplying the shared responsibility model is a common source of security incidents in cloud environments. For instance, improperly configured storage services, weak identity and access management policies, or insufficient monitoring can lead to data breaches, even when the cloud provider's infrastructure is secure.

Furthermore, in multi-cloud environments, the application of the shared responsibility model becomes more complex. Different cloud providers may define responsibilities differently, requiring organizations to adopt a consistent and centralized approach to security management across platforms.

The shared responsibility model therefore represents a fundamental concept for understanding cloud security. It highlights the importance of clearly defining roles and responsibilities in order to ensure effective protection of cloud-based systems and data. As illustrated in the figure below, the distribution of responsibilities decreases for the customer as the level of abstraction increases from IaaS to SaaS.

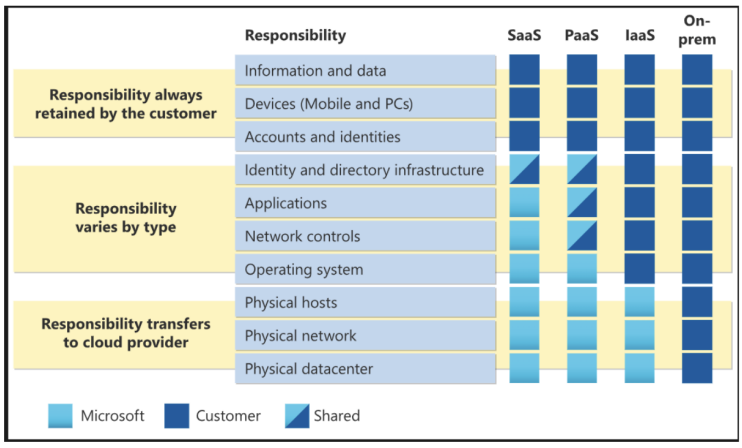

Figure 2.2: Shared Responsability Model

2.2 Multi-Cloud and Multi-Account Architectures

The rapid adoption of cloud computing has led organizations to increasingly rely on complex distributed infrastructures. In this context, multi-cloud and multi-account architectures have emerged as key strategies to address requirements related to scalability, resilience, and organizational flexibility. A multi-cloud architecture refers to the use of services from multiple cloud providers, while a multi-account architecture involves the use of multiple isolated environments within a single cloud platform.

These architectures are particularly prevalent in large enterprises, where different business units, applications, or acquired entities operate independently. While they provide significant operational and strategic benefits, they also introduce substantial challenges, particularly in terms of security, governance, and system management.

2.2.1 Multi-Cloud Architectures

A multi-cloud architecture enables organizations to distribute workloads across multiple cloud providers, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform. This approach allows organizations to leverage the specific capabilities of each provider, selecting services that best fit their technical and business requirements.

One of the primary motivations for adopting a multi-cloud strategy is to avoid vendor lock-in. By reducing dependency on a single provider, organizations gain greater flexibility and can negotiate costs more effectively. Additionally, multi-cloud

architectures enhance system resilience by distributing workloads across different infrastructures, reducing the impact of outages affecting a specific provider.

From a performance perspective, multi-cloud environments enable workload optimization by placing applications closer to users or by leveraging specialized services offered by different providers. This can improve latency, availability, and overall system efficiency.

However, these benefits come at the cost of increased complexity. Each cloud provider has its own service models, APIs, security mechanisms, and configuration paradigms. As a result, organizations must manage heterogeneous environments that require different expertise and operational practices.

2.2.2 Multi-Account Architectures

Within a single cloud provider, organizations commonly adopt a multi-account strategy to structure their environments. In AWS, for example, this involves creating multiple accounts under a single organization to isolate workloads, teams, and environments.

Multi-account architectures provide strong logical isolation between different environments, such as development, testing, and production. This isolation reduces the risk of accidental changes or security incidents affecting critical systems. It also supports the principle of least privilege by allowing access control policies to be enforced at the account level.

In addition, multi-account structures improve governance and cost management. Organizations can track resource usage and allocate budgets per account, enabling better financial control and accountability across business units.

In the context of mergers and acquisitions, multi-account architectures become even more relevant. Each acquired entity may bring its own cloud environment, with specific configurations, tools, and security practices. Integrating these environments into a unified structure is a complex task that requires careful planning and standardization.

2.2.3 Multi-Cloud Security Architecture

Securing multi-cloud and multi-account environments requires a unified and scalable approach capable of addressing the challenges of distributed systems. Modern security architectures aim to provide centralized visibility, consistent policy enforcement, and real-time threat detection across all environments.

A key component of such architectures is the use of centralized control planes, often implemented through SaaS-based platforms. These control planes enable security teams to define and enforce policies across multiple cloud providers and

accounts from a single interface. This reduces operational complexity and ensures consistency in security practices.

Another critical aspect is the collection and correlation of security data across environments. Logs generated by different services, accounts, and providers must be aggregated and analyzed to detect anomalies and potential threats. This requires the integration of advanced monitoring and analytics tools, such as SIEM platforms.

Furthermore, comprehensive security requires visibility into different types of traffic, including ingress (incoming traffic), egress (outgoing traffic), and east-west traffic (internal communication between resources). Monitoring these flows is essential to detect attacks such as data exfiltration, unauthorized access, and lateral movement within the infrastructure.

Automation also plays a central role in multi-cloud security architectures. By leveraging Infrastructure-as-Code (IaC), organizations can deploy and maintain consistent security configurations across environments, reducing the risk of human error and ensuring scalability.

2.2.4 Security Challenges and Weaknesses

Despite their advantages, multi-cloud and multi-account architectures introduce several critical security challenges and inherent weaknesses.

Fragmented visibility and monitoring: One of the most significant challenges is the fragmentation of visibility across multiple environments. Each cloud provider offers its own monitoring tools and logging mechanisms, which are not always compatible or easily integrated. As a result, security teams may lack a unified view of system activity, making it difficult to detect threats in a timely manner.

Inconsistent security policies and configurations: Different environments may be configured using different standards, tools, and practices. This inconsistency leads to gaps in security controls, increasing the risk of vulnerabilities. For example, access control policies may vary between accounts, and logging configurations may not be uniformly enforced.

High risk of misconfigurations: Misconfigurations are one of the leading causes of security incidents in cloud environments. In multi-cloud architectures, the diversity of interfaces and services increases the likelihood of errors. Common issues include publicly exposed storage buckets, overly permissive access policies, and improperly configured network rules.

Complex identity and access management (IAM): Managing identities and permissions across multiple accounts and providers is particularly challenging. Organizations must ensure that users have appropriate access rights while avoiding excessive privileges. Without proper governance, identity sprawl and privilege escalation can occur, increasing the attack surface.

Expanded attack surface: The use of multiple cloud environments significantly increases the number of potential entry points for attackers. Each account, service, and API represents a potential vulnerability that must be secured. This expanded attack surface makes it more difficult to implement comprehensive protection strategies.

Tool fragmentation and lack of integration: Organizations often deploy different security tools for each cloud provider. This fragmentation complicates integration and reduces the effectiveness of security operations, as data must be collected, normalized, and correlated across multiple systems.

Data governance and compliance challenges: Ensuring compliance with regulatory requirements becomes more complex in distributed environments. Data may be stored across different regions and providers, each subject to different legal frameworks. This makes it difficult to enforce consistent data protection and governance policies.

Operational complexity and scalability issues: Managing multi-cloud environments requires advanced operational capabilities. Monitoring, deployment, and incident response processes must be adapted to handle distributed systems at scale. Without automation and standardization, these processes can become inefficient and error-prone.

2.2.5 Relevance to Cloud Security Operations

In the context of a Global Security Operations Center (GSOC), these challenges highlight the critical need for centralized, scalable, and automated security solutions. Security teams must be able to monitor multiple environments simultaneously, correlate events from different sources, and respond to threats in real time.

This requires the implementation of unified security platforms capable of aggregating logs, enforcing policies, and providing consistent visibility across all environments. Technologies such as Security Information and Event Management (SIEM) systems and Cloud-Native Application Protection Platforms (CNAPP) play a key role in achieving these objectives.

The increasing complexity of multi-cloud and multi-account architectures therefore necessitates the development of standardized and scalable security frameworks. Addressing these challenges is essential to ensure effective protection of cloud environments and constitutes the core objective of this thesis. Given the complexity and security challenges introduced by multi-cloud and multi-account architectures, organizations rely on various security solutions to ensure visibility, compliance, and threat detection. The following section presents an overview of the state of the art cloud security solutions currently adopted in industry.

2.3 State of the Art Cloud Security Solutions

The increasing complexity of multi-cloud and multi-account environments has led to the development of advanced cloud security solutions aimed at providing centralized visibility, threat detection, and compliance management. These solutions are designed to address the challenges associated with distributed infrastructures, heterogeneous configurations, and large-scale data processing.

Modern cloud security approaches can be broadly categorized into three main types: Cloud-Native Application Protection Platforms (CNAPP), Security Information and Event Management (SIEM) systems, and cloud-native security services provided by cloud providers.

2.3.1 Cloud-Native Application Protection Platforms (CNAPP)

Cloud-Native Application Protection Platforms (CNAPP) represent a comprehensive approach to securing cloud environments by integrating multiple security capabilities into a unified platform. Rather than relying on separate tools, CNAPP consolidates several components to provide end-to-end visibility and protection across the entire cloud lifecycle. A CNAPP solution typically includes the following key components:

- **Cloud Security Posture Management (CSPM):** CSPM focuses on identifying and remediating misconfigurations in cloud environments. It continuously evaluates cloud resources against security best practices and compliance standards. CSPM tools detect issues such as publicly exposed storage, overly permissive identity policies, or missing encryption. This component is essential because misconfigurations are one of the primary causes of cloud security breaches. It analyzes cloud resources to identify issues such as publicly exposed storage buckets, overly permissive access controls, and other configuration weaknesses. Although CSPM is a fundamental component of CNAPP, CNAPP solutions extend beyond infrastructure security by also protecting workloads, applications, and CI/CD pipelines. CSPM primarily focuses on the configuration of cloud infrastructure services (e.g., virtual machines, storage, and networking), whereas CNAPP incorporates workload protection, threat detection, and runtime security, providing a more comprehensive approach to cloud security.

- **Data Security Posture Management (DSPM):** DSPM focuses specifically on ensuring the proper protection of data. This involves identifying and locating sensitive data, monitoring access, and assessing the security posture of any application or repository where this data resides. Within a CNAPP solution, DSPM tools analyze cloud data stores to classify sensitive information

and evaluate their security posture. This enables the implementation of appropriate encryption mechanisms, access controls, and compliance with data protection regulations. DSPM therefore plays a critical role in reducing data exposure risks, which are among the most significant threats in cloud environments.

- **Kubernetes Security Posture Management (KSPM):** KSPM focuses on securing Kubernetes environments by continuously assessing the security posture of clusters, nodes, and containerized workloads. It identifies misconfigurations, insecure settings, and policy violations within Kubernetes components. As Kubernetes has become a standard platform for orchestrating containerized applications in cloud-native environments, KSPM plays a critical role in securing microservices-based architectures. It ensures that configurations align with best practices and helps mitigate risks associated with container orchestration systems.

- **Cloud Workload Protection (CWP):** CWP provides runtime protection for workloads deployed in the cloud, including virtual machines, containers, and serverless functions. It monitors system activity, detects malicious behavior, and enforces security policies at the workload level. It protects against threats such as malware, privilege escalation, and unauthorized access. By continuously discovering assets, identifying risks, and detecting threats, CWPP solutions enable organizations to monitor their multi-cloud environments and identify deviations from established security policies.

- **Cloud Infrastructure Entitlement Management (CIEM):** CIEM addresses identity and access management challenges in cloud environments. It analyzes permissions assigned to users, roles, and services, and identifies excessive privileges or risky access patterns. It helps enforce the principle of least privilege and prevents identity-based attacks.

- **Cloud Detection and Response (CDR):** CDR focuses on detecting and responding to security threats within cloud environments. It continuously monitors cloud activity to identify suspicious behaviors, unauthorized access attempts, malware, and insider threats through real-time analysis and behavioral monitoring. CDR solutions often rely on machine learning and advanced analytics to detect anomalies. Within a CNAPP framework, CDR can identify indicators of compromise such as unusual authentication patterns, suspicious API activity, and privilege escalation attempts, enabling rapid threat detection and response.

2.3.2 Security Information and Event Management (SIEM)

Security Information and Event Management (SIEM) systems are a core component of modern security operations. They are designed to collect, aggregate, and analyze security data from multiple sources in order to detect, investigate, and respond to security incidents. A SIEM system typically consists of several key components:

- **Data Collection and Ingestion:** SIEM platforms collect logs and events from various sources, including cloud services, applications, network devices, and operating systems. These logs include authentication events, API calls, network traffic, and system activity, ensuring comprehensive visibility.

- **Log Normalization and Parsing:** Collected data is normalized into a common format to enable consistent analysis across sources. Parsing extracts relevant fields from raw logs, allowing structured queries and correlation.

- **Correlation Engine:** The correlation engine analyzes events and identifies relationships between them using predefined rules or patterns. It detects suspicious behaviors such as failed login attempts, privilege escalation, or lateral movement.

- **Threat Detection and Analytics:** SIEM systems use advanced analytics, including statistical models and machine learning, to identify anomalies and detect both known and unknown threats.

- **Alerting and Incident Management:** When suspicious activity is detected, alerts are generated and prioritized based on severity. These alerts are integrated into incident response workflows to enable timely investigation and remediation.

- **Data Storage and Retention:** SIEM platforms store large volumes of log data for long periods, enabling forensic analysis, compliance reporting, and historical investigations. Efficient storage management is critical in large-scale environments.

- **Dashboards and Visualization:** SIEM systems provide real-time dashboards that allow security analysts to monitor system activity, identify trends, and assess threats quickly.

- **Compliance and Reporting:** SIEM platforms generate reports that support compliance with regulatory requirements and internal policies by providing detailed audit trails.

In contrast to CNAPP, which focuses on proactive risk identification and prevention, SIEM provides a reactive approach centered on detection and response. The combination of both solutions enables organizations to achieve comprehensive cloud security coverage, addressing both preventive and detective aspects of security operations.

2.3.3 Cloud-Native Security Services

In addition to third-party security platforms such as CNAPP and SIEM, cloud providers offer native security services designed to protect their environments. These services are tightly integrated within the cloud ecosystem and provide built-in capabilities for threat detection, compliance monitoring, and security management.

These services leverage the provider's internal visibility into infrastructure and services, enabling them to analyze events, configurations, and workloads with high accuracy and minimal deployment effort.

The main categories of cloud-native security services include:

- **Threat Detection Services:** Services such as AWS GuardDuty and Google Cloud Event Threat Detection continuously monitor logs, network traffic, and account activity to identify threats such as unauthorized access, anomalous behavior, or data exfiltration.
- **Security Posture and Compliance Management:** Tools such as AWS Security Hub and Google Security Command Center aggregate security findings and evaluate configurations against best practices and compliance standards.
- **Vulnerability Assessment:** Services such as AWS Inspector automatically scan workloads and configurations to detect vulnerabilities, outdated software, and insecure settings.
- **Identity and Access Management (IAM):** IAM services enable fine-grained access control and enforce the principle of least privilege to prevent unauthorized access.
- **Logging and Monitoring:** Services such as AWS CloudTrail and Google Cloud Logging collect detailed records of API calls, user activity, and system events, which are essential for auditing and integration with SIEM systems.
- **Automation and Remediation:** Cloud-native tools support automated responses to security findings, such as isolating compromised resources or enforcing corrective configurations.

The following table provides a comparative overview of commonly used cloud security tools, highlighting their characteristics, strengths, and limitations.

Table 2.3: Cloud Security Tools

Tool	**Type**	**Strengths**	**Limitations**	**Best for**
Prisma Cloud (Palo Alto Networks)	CNAPP (Cloud-Native Application Protection Platform)	Multi-cloud support, deep visibility into workloads, strong compliance and vulnerability management	Complex UI, high cost	Enterprises with multi-cloud environments needing deep security posture management
Splunk	SIEM (Security Information and Event Management)	Powerful data analytics, custom dashboards, scalable for large environments, integration with many data sources	Requires skilled personnel	Organizations needing advanced log analysis and threat detection
AWS Security Hub	CSPM (Cloud Security Posture Management)	Native integration with AWS services, pay-as-you-go, easy setup	Limited to AWS, basic customization	AWS organizations
Google Security Command Center	CSPM	Native integration with GCP, vulnerability scanning	Limited to GCP	GCP-focused businesses needing centralized security management

2.3.4 Advantages and Limitations

Cloud-native security services offer several advantages. They are easy to deploy, require minimal configuration, and provide deep integration within the cloud provider's ecosystem. Moreover, their direct access to native data sources enhances detection accuracy and enables efficient monitoring.

However, these services also present important limitations. They are restricted to a single cloud provider and therefore do not provide unified visibility across multi-cloud environments. As a result, organizations must deploy multiple native solutions, leading to fragmentation and increased operational complexity.

From a broader perspective, each category of cloud security solutions addresses specific aspects of the problem. CNAPP platforms excel at identifying misconfigurations and vulnerabilities and provide a comprehensive view of the security posture. SIEM systems are highly effective for log aggregation, correlation, and incident detection, but depend heavily on proper data ingestion and configuration. Cloud-native tools offer strong integration but remain limited to their respective ecosystems.

This diversity of tools leads to several challenges, including tool sprawl, integration complexity, and inconsistent enforcement of security policies. Furthermore, the absence of a unified security architecture makes it difficult to achieve centralized visibility and coordinated threat detection.

Consequently, organizations must adopt a combination of these solutions to achieve comprehensive protection. This highlights the need for integrated and scalable security architectures capable of unifying visibility, policy enforcement, and threat detection across multi-cloud environments.

2.3.5 Relevance to Multi-Cloud Security

In multi-cloud environments, the need for centralized visibility and consistent security enforcement becomes critical. Cloud-native tools provide strong local visibility within each environment, while CNAPP and SIEM solutions enable global visibility and cross-environment correlation.

CNAPP platforms provide proactive detection of vulnerabilities and misconfigurations, while SIEM systems enable reactive monitoring and incident response through log analysis. Together, they form a comprehensive security framework combining preventive and detective capabilities.

However, integrating these solutions in large-scale environments remains complex, requiring careful design of architectures, data pipelines, and governance models. This challenge highlights the need for scalable and standardized approaches to cloud security, which is the focus of this thesis.

2.4 Cloud Security Challenges

Despite the availability of advanced cloud security solutions, several challenges remain in ensuring effective and scalable protection of multi-cloud environments. These challenges stem from the increasing complexity of distributed infrastructures, the heterogeneity of cloud services, and the limitations of existing security approaches. As organizations adopt multi-cloud strategies, they must manage a growing number of resources, configurations, and data flows, which significantly increases the difficulty of maintaining a consistent and secure environment.

- **Lack of centralized visibility:** In multi-cloud environments, resources and workloads are distributed across multiple providers and accounts, each exposing different interfaces, logging mechanisms, and monitoring tools. This fragmentation makes it difficult to obtain a unified and comprehensive view of system activity. Security events are often scattered across platforms, requiring complex aggregation and correlation processes. As a result, security teams may face delays in detecting incidents, and important signals may be overlooked due to incomplete visibility.

- **Configuration complexity and misconfigurations:** Cloud environments involve a large number of configurable components, including virtual networks, storage services, identity policies, and security groups. The diversity of services and configuration options across providers increases the likelihood of errors. Misconfigurations, such as publicly exposed storage buckets, overly permissive access controls, or improperly configured network rules, remain one of the leading causes of cloud security breaches. The dynamic nature of cloud environments further complicates this issue, as configurations may change frequently and require continuous monitoring.

- **Identity and access management challenges:** Identity and Access Management (IAM) is a critical aspect of cloud security, yet it becomes significantly more complex in multi-cloud environments. Each provider implements its own IAM model, with different concepts, policies, and permission structures. Managing identities, roles, and access rights across multiple platforms increases the risk of inconsistencies and excessive privileges. Ensuring the principle of least privilege while maintaining operational efficiency is particularly challenging, and failures in IAM management can lead to unauthorized access, privilege escalation, or insider threats.

- **Log management and scalability:** Cloud environments generate massive volumes of logs from various sources, including infrastructure components, applications, APIs, and user activities. These logs are essential for monitoring,

auditing, and threat detection. However, collecting, storing, and processing this data at scale presents significant technical challenges. Differences in log formats and structures across providers complicate normalization and correlation. Without efficient log management pipelines and scalable analytics capabilities, critical security events may go undetected or be identified too late.

- **Tool fragmentation and integration issues:** Organizations typically rely on a combination of security tools, including CNAPP platforms, SIEM systems, and cloud-native services. While each tool addresses specific aspects of security, their integration into a cohesive architecture is complex. Differences in data models, APIs, and operational workflows create integration challenges and may result in inconsistent security policies. This fragmentation increases operational overhead and reduces the overall effectiveness of security operations.

- **Compliance and governance challenges:** Ensuring compliance with regulatory requirements and internal policies becomes increasingly difficult in multi-cloud environments. Data may be distributed across different regions and providers, each subject to distinct legal and regulatory frameworks. Organizations must ensure consistent enforcement of security controls, data protection policies, and audit mechanisms across all environments. This requires continuous monitoring and alignment with compliance standards, which can be challenging to achieve at scale.

- **Operational complexity and scalability:** Managing multi-cloud environments requires advanced expertise and coordination across multiple teams and technologies. Security operations must handle monitoring, incident response, policy enforcement, and system updates across heterogeneous infrastructures. As the number of cloud accounts and services grows, manual processes become inefficient and error-prone. The lack of automation and standardization limits scalability and increases the risk of operational failures.

- **Onboarding and integration of new environments:** One of the most critical challenges in evolving cloud infrastructures is the onboarding of new environments into existing security frameworks. In large organizations, particularly those undergoing mergers and acquisitions, new cloud accounts and platforms must be integrated rapidly while maintaining consistent security controls. This process often involves configuring logging pipelines, access permissions, and monitoring tools. When performed manually, onboarding becomes time-consuming, error-prone, and difficult to scale, leading to delays in achieving full security coverage.

- **Heterogeneity of cloud ecosystems:** Multi-cloud environments are inherently heterogeneous, as each provider offers distinct services, architectures, and operational models. This diversity complicates the standardization of security practices and requires security teams to develop expertise across multiple platforms. The lack of uniformity in APIs, configuration models, and security mechanisms increases the difficulty of implementing centralized security strategies.

These challenges highlight the limitations of current cloud security approaches and emphasize the need for centralized, scalable, and automated solutions. In particular, organizations require security architectures capable of providing unified visibility, standardized onboarding processes, consistent policy enforcement, and efficient threat detection across all cloud environments.

Addressing these challenges constitutes the main objective of this thesis, which aims to design and implement a scalable and automated cloud security framework adapted to multi-cloud infrastructures.

2.5 Amadeus Cloud Security Context and Challenges

Amadeus operates in a highly dynamic and large-scale technological environment, where its business services rely on a hybrid and increasingly multi-cloud infrastructure. Historically, the company managed its operations through on-premises data centers, handling a high volume of transactions with strict availability and performance requirements. Over time, in order to improve scalability, resilience, and global service delivery, Amadeus progressively migrated part of its infrastructure to cloud platforms, primarily Microsoft Azure.

As part of this transformation, centralized security solutions such as Cloud-Native Application Protection Platforms (CNAPP) and Security Information and Event Management (SIEM) systems were introduced. These tools enabled the Global Security Operations Center (GSOC) to monitor cloud environments, detect security threats, and enforce compliance policies. Initially, these solutions were deployed in a relatively homogeneous environment, covering Azure-based tenants and on-premises infrastructures.

However, even within this initial scope, the integration of cloud environments into the security ecosystem was not fully automated. The onboarding of Azure tenants and on-premises systems into CNAPP and SIEM platforms was largely performed manually. This process required significant effort from security teams, involving configuration of logging pipelines, access permissions, and data ingestion mechanisms. While manageable at a smaller scale, this manual approach introduced limitations in terms of scalability, consistency, and operational efficiency.

The situation became significantly more complex with the expansion of Amadeus through mergers and acquisitions. New business units introduced additional cloud environments based on different providers, including Amazon Web Services (AWS) and Google Cloud Platform (GCP). Each of these environments operates with distinct architectures, service models, identity management systems, and logging mechanisms.

This transition from a relatively homogeneous infrastructure to a heterogeneous multi-cloud environment exposed several critical challenges. First, the lack of standardization across cloud providers made it difficult to apply consistent security policies. Each environment required specific configurations, increasing the risk of misconfigurations and security gaps.

Second, the existing CNAPP and SIEM solutions, originally designed for Azure and on-premises environments, were not fully adapted to support multi-cloud integration at scale. Onboarding new environments into these platforms required significant manual effort, including the configuration of connectors, normalization of log formats, and adaptation of detection rules.

Third, the absence of automation in the onboarding process created bottlenecks in the integration of new environments. As the number of cloud accounts increased, manual processes became inefficient, error-prone, and difficult to maintain. This directly impacted the ability of the GSOC to ensure comprehensive and timely security coverage.

Furthermore, the diversity of logging formats and APIs across cloud providers complicated the aggregation and correlation of security data. Without a unified and standardized data pipeline, it became challenging to achieve centralized visibility and perform effective threat detection across environments.

In addition, identity and access management became increasingly complex. Each cloud provider implements its own IAM model, requiring security teams to manage permissions across multiple systems. This increased the risk of inconsistent access control policies and potential privilege escalation.

These challenges highlight a fundamental limitation of the existing security architecture: while CNAPP and SIEM solutions were already in place, their integration was not designed to scale in a rapidly evolving multi-cloud context. The reliance on manual processes and the lack of standardized onboarding mechanisms prevented the GSOC from achieving a fully centralized and automated security posture.

Consequently, there is a critical need for a scalable, automated, and standardized approach to onboarding cloud environments into security platforms. Such an approach must ensure consistent configuration, efficient data integration, and unified visibility across all cloud providers.

Addressing these challenges constitutes the core objective of this thesis, which aims to design and implement a cloud security architecture capable of supporting

automated onboarding, centralized monitoring, and effective threat detection in a multi-cloud environment.

Chapter 3

Proposed Security Architecture and Methodology

This chapter presents the proposed security architecture and methodology designed to address the challenges identified in multi-cloud environments. As discussed in the previous chapters, the increasing complexity of distributed cloud infrastructures, combined with the limitations of existing security solutions, necessitates a scalable, automated, and centralized approach to cloud security.

The objective of this work is to design a security framework capable of ensuring consistent visibility, efficient threat detection, and standardized onboarding across heterogeneous cloud environments. In particular, the proposed solution aims to integrate cloud platforms into a unified security architecture, with a primary focus on Amazon Web Services (AWS).

While the broader context of this work involves a multi-cloud environment including Microsoft Azure and Google Cloud Platform (GCP), the implementation and architectural design presented in this chapter are specifically developed for AWS environments. This choice is motivated by the fact that cloud security architectures are inherently tied to provider-specific services, APIs, and configurations. As a result, designing a fully generic solution across all cloud providers would introduce additional complexity beyond the scope of this work.

Nevertheless, the proposed architecture is designed with extensibility in mind. The integration of additional cloud providers, such as GCP, is considered as part of future work. The principles, methodologies, and automation mechanisms presented in this chapter can be adapted to other cloud platforms with appropriate modifications.

This chapter first introduces the overall architecture of the proposed solution,

highlighting its key components and design principles. It then describes the mechanisms used for log collection and data ingestion, which are essential for enabling centralized monitoring and analysis. Particular attention is given to the integration of security tools, including Cloud-Native Application Protection Platforms (CNAPP) and Security Information and Event Management (SIEM) systems, in order to ensure comprehensive coverage across cloud environments.

In addition, this chapter presents the methodology adopted to automate the onboarding of new cloud environments into the security ecosystem. This includes the use of Infrastructure-as-Code (IaC) techniques to standardize configurations, reduce manual intervention, and improve scalability.

Finally, the chapter discusses how the proposed architecture addresses the key challenges identified earlier, including the lack of centralized visibility, tool fragmentation, and operational complexity. The proposed solution therefore provides a structured and scalable approach to cloud security, enabling efficient monitoring, consistent policy enforcement, and improved incident response capabilities. Extending the proposed architecture to fully support additional cloud providers represents an important direction for future research and development.

3.1 Proposed Architecture Overview

The proposed security architecture is designed to address the challenges associated with securing multi-cloud environments, with a particular focus on the integration and protection of Amazon Web Services (AWS) tenants within a centralized security framework. As highlighted in previous chapters, the increasing complexity of distributed cloud infrastructures, combined with the limitations of existing security solutions, requires a scalable, automated, and standardized approach to cloud security.

The primary objective of the proposed architecture is to provide centralized visibility, consistent policy enforcement, and efficient threat detection across all onboarded cloud environments. To achieve this, the architecture integrates multiple components, including cloud-native logging services, Infrastructure-as-Code (IaC) mechanisms, and centralized security platforms such as Security Information and Event Management (SIEM) systems and Cloud-Native Application Protection Platforms (CNAPP).

At a high level, the architecture is structured around three main layers: the cloud environment layer, the data collection and processing layer, and the security operations layer.

- **Cloud Environment Layer:** This layer consists of AWS tenants representing different business units or environments, such as development, testing, and production. Each account contains various cloud resources, including compute

instances, storage services, networking components, and identity management configurations. These environments generate a continuous stream of operational and security-related events, such as API calls, user activities, and system logs.

- **Data Collection and Processing Layer:** The second layer is responsible for collecting, aggregating, and processing data generated by the cloud environments. AWS-native services such as CloudTrail for APIs logs and VPC Flow Logs are used to capture detailed information about system activity. These logs are then centralized and prepared for further analysis. This layer also includes mechanisms for normalizing and structuring data to ensure compatibility with downstream security platforms.
- **Security Operations Layer:** The final layer integrates centralized security tools, including SIEM and CNAPP platforms. The SIEM system ingests logs from all onboarded environments, enabling real-time monitoring, correlation of events, and detection of potential security incidents. In parallel, the CNAPP platform provides visibility into the overall security posture by identifying misconfigurations, vulnerabilities, and compliance issues across cloud environments.

A key feature of the proposed architecture is the automation of the onboarding process for new cloud environments. Using Infrastructure-as-Code (IaC) techniques, the architecture enables standardized deployment of logging configurations, access controls, and data integration pipelines. This ensures that newly created AWS tenants can be rapidly and consistently integrated into the centralized security ecosystem without manual intervention.

The architecture also emphasizes modularity and extensibility. While the current implementation focuses on AWS environments, the design principles allow for the integration of additional cloud providers in the future. This is achieved by abstracting core functionalities such as log collection, data normalization, and security integration, enabling the architecture to adapt to heterogeneous cloud ecosystems.

From an operational perspective, the proposed architecture supports the Global Security Operations Center (GSOC) by providing a unified view of security events across all environments. Security analysts can monitor activity, investigate incidents, and enforce policies from a centralized interface, improving response times and overall security posture.

Finally, the architecture is designed to address the key challenges identified in the previous chapter, including the lack of centralized visibility, tool fragmentation, and operational complexity. By combining automation, standardization, and integration, the proposed solution provides a scalable and efficient framework for securing cloud environments in a multi-cloud context.

A high-level representation of the architecture is illustrated in the figure below, which highlights the interactions between cloud environments, data collection mechanisms, and security platforms.

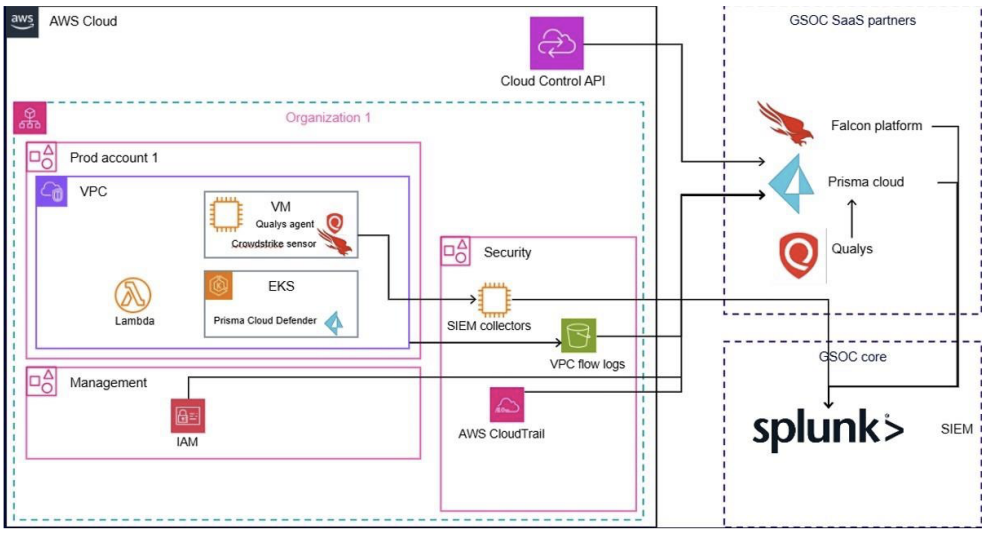

Figure 3.1: High Level Design

3.2 Architecture Description

3.2.1 AWS Environment and Workload Components

The foundation of the proposed architecture is the AWS cloud environment, which hosts the workloads and services that must be monitored, protected, and integrated into the centralized security ecosystem. Each AWS tenant represents a logical environment associated with a specific business unit, project, or operational context, such as production, development, or testing environments. This separation allows organizations to enforce isolation, apply different security policies, and manage resources independently.

Within each AWS account, resources are deployed inside Virtual Private Clouds (VPCs), which provide network-level isolation and segmentation. A VPC defines the network boundaries of the environment, including IP address ranges, subnets, routing configurations, and connectivity with external networks. This segmentation is critical for limiting lateral movement in case of a security breach and for enforcing network-level security controls.

The architecture supports multiple types of workloads, each with distinct security considerations:

- **Virtual Machines (EC2 Instances):** Amazon EC2 instances are used to host traditional applications and services. These instances represent a critical attack surface, as they run operating systems and application stacks that may be vulnerable to exploits. To mitigate these risks, security agents such as Qualys and CrowdStrike Falcon are deployed on each instance. These agents provide continuous vulnerability scanning, endpoint detection and response (EDR), and real-time monitoring of system activity. They enable the detection of malicious behaviors such as unauthorized access, privilege escalation, or execution of suspicious processes.
- **Containerized Workloads (EKS):** Modern applications are often deployed using container orchestration platforms such as Amazon Elastic Kubernetes Service (EKS). Containers introduce additional layers of complexity due to their dynamic and ephemeral nature. To address these challenges, the architecture integrates CNAPP agents such as Prisma Cloud Defender, which provide runtime protection, vulnerability management, and compliance monitoring for containerized environments. These tools monitor container behavior, detect anomalies, and ensure that configurations adhere to security best practices.
- **Serverless Components (Lambda):** Serverless computing services, such as AWS Lambda, are used to execute event-driven functions without managing underlying infrastructure. While serverless architectures reduce operational overhead, they introduce new security challenges, including limited visibility and the need to secure function execution and permissions. These components generate logs that must be captured and analyzed to detect abnormal execution patterns or unauthorized invocations.
- **Identity and Access Management (IAM):** IAM is a fundamental component of cloud security, responsible for controlling access to resources. It defines users, roles, and permissions, and governs how identities interact with cloud services. In complex environments, IAM configurations can become difficult to manage, increasing the risk of excessive privileges or misconfigurations. The architecture ensures that IAM activities are monitored and logged, enabling the detection of suspicious actions such as unauthorized role assumptions or privilege escalation attempts.

Overall, the AWS environment layer represents the primary source of security-relevant data. Each component generates logs, metrics, and events that are essential for monitoring, analysis, and threat detection. Ensuring comprehensive coverage of these components is therefore a critical requirement for the effectiveness of the overall security architecture.

3.2.2 Centralized Log Collection Architecture

Effective security monitoring relies on the collection of comprehensive and high-quality data from cloud environments. In the proposed architecture, log collection is a critical component that enables visibility into system activities, user behavior, and network communications.

In order to ensure scalability and centralization, the architecture introduces a dedicated **security account** within each AWS tenant structure. This account acts as a centralized logging and aggregation layer, where logs from all other AWS accounts are collected before being forwarded to external security platforms such as the SIEM.

This approach follows AWS best practices for multi-account architectures, where security and logging functions are isolated in a dedicated account. By centralizing logs within a security account, the architecture improves visibility, simplifies management, and enhances security by restricting direct access to log data.

AWS provides several native logging services that are leveraged to capture different types of events:

- **AWS CloudTrail:** CloudTrail records all API calls made within an AWS account, including actions performed by users, services, and applications. These logs provide detailed information about who performed an action, when it occurred, and which resources were affected. In the proposed architecture, CloudTrail is configured to send logs from all AWS accounts to a centralized storage location within the security account. This enables unified auditing and simplifies the detection of suspicious activities such as unauthorized access or abnormal API usage.

- **VPC Flow Logs:** VPC Flow Logs capture information about network traffic within the VPC, including accepted and rejected connections. These logs provide visibility into communication patterns between resources and can be used to detect anomalies such as unexpected outbound traffic or unauthorized access attempts. Similar to CloudTrail logs, VPC Flow Logs are aggregated in the security account to enable centralized analysis.

- **Application and System Logs:** In addition to infrastructure-level logs, applications and operating systems generate their own logs. These logs contain valuable information about application behavior, errors, and user interactions. In the proposed architecture, these logs are directly forwarded to the SIEM, ensuring that application-level events are also included in the overall security monitoring process.

The use of a dedicated security account introduces several advantages. First, it provides a single point of aggregation for all logs, simplifying data management

and improving visibility across accounts. Second, it enhances security by isolating log storage from operational environments, reducing the risk of tampering or unauthorized access. Finally, it enables consistent configuration of logging policies across all accounts.

However, the diversity and volume of logs generated by cloud environments still present significant challenges. Logs must be collected in a consistent manner, stored securely, and processed efficiently. The architecture addresses these challenges by centralizing log collection within the security account and preparing data for integration with downstream security platforms.

Furthermore, log normalization is an essential step in this process. Since different services generate logs in different formats, data must be structured and standardized to enable effective analysis and correlation. This ensures that systems such as SIEM platforms can process and interpret the data correctly.

By establishing a robust and centralized log collection mechanism based on a dedicated security account, the architecture ensures that all relevant security events are captured and made available for analysis, forming the foundation for threat detection and incident response.

3.2.3 SIEM Integration and Log Ingestion

Once logs are collected from the different components of the AWS environment, they must be centralized, processed, and analyzed in order to enable effective security monitoring and incident detection. This function is fulfilled by the Security Information and Event Management (SIEM) platform, which represents a core component of the proposed architecture.

In this work, the SIEM platform is implemented using Splunk, which serves as the central system for log ingestion, storage, correlation, and analysis. Splunk enables the aggregation of data from multiple sources, providing a unified view of security events across all onboarded cloud environments.

- **Log Ingestion Mechanisms:** Logs generated by AWS services, such as CloudTrail and VPC Flow Logs, are forwarded to the SIEM through dedicated ingestion pipelines. These pipelines may rely on collectors, forwarding agents, or API-based integrations that securely transmit data from cloud environments to the SIEM platform. Ensuring reliable and secure data transfer is critical to avoid data loss and maintain the integrity of security monitoring processes.

- **Data Normalization and Structuring:** Before analysis, logs must be parsed and normalized into a consistent format. This process involves extracting relevant fields, such as timestamps, source IP addresses, user identities, and event types. Normalization enables the SIEM to process heterogeneous data

sources in a unified way, facilitating efficient querying and correlation of events across different environments.

- **Event Correlation and Detection:** One of the key capabilities of the SIEM is its ability to correlate events from multiple sources in order to identify complex attack patterns. By applying predefined rules, behavioral analytics, and detection models, the SIEM can detect suspicious activities such as repeated failed login attempts, abnormal API usage, or lateral movement within the infrastructure.

- **Alerting and Incident Response:** When a potential security incident is detected, the SIEM generates alerts that are forwarded to the Global Security Operations Center (GSOC). These alerts are prioritized based on severity and contextual information, allowing security analysts to focus on the most critical threats. The SIEM also supports incident investigation by providing detailed logs and historical data for forensic analysis.

- **Scalability and Data Management:** Given the large volume of logs generated in cloud environments, the SIEM must be able to scale efficiently. Splunk provides mechanisms for distributed data storage and processing, ensuring that large datasets can be handled without performance degradation. Efficient data retention and indexing strategies are also required to balance storage costs and analytical capabilities.

From an architectural perspective, the SIEM acts as the central hub of the security monitoring system. It aggregates data from all cloud environments, correlates events, and provides actionable insights to security teams. This centralized approach addresses one of the main challenges identified earlier, namely the lack of unified visibility in multi-cloud environments.

However, the effectiveness of the SIEM is highly dependent on the quality and completeness of the data it receives. This highlights the importance of robust log collection mechanisms and standardized onboarding processes, which ensure that all relevant data sources are properly integrated into the system.

3.2.4 CNAPP Integration and Security Posture Management

In addition to reactive monitoring provided by the SIEM, the proposed architecture integrates a Cloud-Native Application Protection Platform (CNAPP) to ensure proactive security across cloud environments. CNAPP solutions focus on identifying misconfigurations, vulnerabilities, and compliance issues before they can be exploited, complementing the detection capabilities of the SIEM.

In this architecture, the CNAPP functionality is implemented using Prisma Cloud, which provides comprehensive visibility into the security posture of AWS environments. Prisma Cloud operates by continuously analyzing cloud configurations, workloads, and runtime activities to detect potential security risks.

- **Workload Protection and Runtime Security:** Prisma Cloud Defender agents are deployed on workloads, including virtual machines (EC2 instances) and containerized environments (EKS). These agents monitor system activity in real time, detecting threats such as malware execution, privilege escalation, and unauthorized access attempts. In containerized environments, the Defender also provides visibility into container behavior, ensuring that workloads comply with security policies during execution.

- **Cloud Security Posture Management (CSPM):** A key component of CNAPP is CSPM, which continuously evaluates cloud configurations against security best practices and compliance standards. Prisma Cloud analyzes AWS resources such as storage buckets, IAM policies, and network configurations to identify misconfigurations. For example, it can detect publicly exposed storage, overly permissive access controls, or missing encryption settings. By providing continuous assessment, CSPM helps reduce the risk of security breaches caused by configuration errors.

- **Vulnerability Management:** CNAPP also includes vulnerability scanning capabilities that identify weaknesses in operating systems, container images, and application dependencies. These vulnerabilities are prioritized based on their severity and potential impact, allowing security teams to focus on the most critical risks. This proactive approach reduces the attack surface of cloud environments.

- **Compliance Monitoring:** The CNAPP platform continuously assesses cloud environments against regulatory and organizational standards, such as CIS benchmarks or internal security policies. This ensures that configurations remain compliant and helps organizations prepare for audits.

- **Integration with the Overall Architecture:** Within the proposed architecture, CNAPP operates in parallel with the SIEM. While the SIEM focuses on log-based detection and incident response, CNAPP provides continuous assessment of the security posture. Findings generated by Prisma Cloud can be integrated into the SIEM to enhance correlation and provide additional context for security events.

This complementary relationship between CNAPP and SIEM is essential for

achieving comprehensive cloud security. CNAPP enables proactive risk identification and prevention, while SIEM provides reactive monitoring and incident response capabilities.

However, similar to SIEM, the effectiveness of CNAPP depends on the proper onboarding and configuration of cloud environments. Ensuring that all AWS accounts are correctly integrated into the CNAPP platform is therefore a critical requirement. This reinforces the need for automated onboarding mechanisms, which are addressed in the following sections.

3.2.5 Automated Onboarding Methodology

One of the main limitations identified in the existing security architecture is the reliance on manual processes for onboarding cloud environments into security platforms. As discussed in previous chapters, the integration of Amadeus cloud tenants into CNAPP and SIEM systems was initially performed manually, requiring significant effort from security teams. This approach does not scale in a multi-cloud context, particularly in environments characterized by rapid growth, such as those resulting from mergers and acquisitions.

To address this limitation, the proposed architecture introduces an automated onboarding methodology based on Infrastructure-as-Code (IaC) principles. The objective is to standardize and automate the integration of new AWS tenants into the centralized security ecosystem, ensuring consistency, scalability, and efficiency.

- **Infrastructure-as-Code Approach:** The onboarding process is implemented using Infrastructure-as-Code tools. This approach allows the definition of infrastructure configurations, logging mechanisms, and security integrations as code templates. These templates can be versioned, reused, and deployed automatically across multiple environments, ensuring consistent configurations.
- **Standardized Onboarding Templates:** Predefined templates are used to deploy all required components for security integration. These templates include the configuration of logging services (such as CloudTrail and VPC Flow Logs), the setup of permissions and roles required for data access, and the integration with external security platforms such as SIEM and CNAPP. By using standardized templates, the architecture ensures that all onboarded tenants follow the same security baseline.
- **Automated Deployment Process:** When a new AWS tenant is created or needs to be integrated, the onboarding process is triggered automatically. The Infrastructure as a code templates are deployed within the cloud tenant, configuring all necessary components without manual intervention. This significantly reduces the time required to onboard new environments and eliminates the risk of human errors associated with manual configurations.

- **Integration with Security Platforms:** As part of the onboarding process, the architecture ensures that logs and security data are automatically forwarded to the SIEM platform and that the account is registered within the CNAPP solution. This includes configuring log forwarding pipelines, enabling required APIs, and granting appropriate permissions for data collection and analysis.

- **Consistency and Compliance Enforcement:** Automation ensures that all environments are configured according to predefined security standards. This reduces inconsistencies across accounts and ensures compliance with organizational policies. Any deviation from the expected configuration can be detected and corrected more easily.

- **Scalability and Maintainability:** The use of IaC enables the architecture to scale efficiently as the number of cloud accounts increases. Updates to security configurations can be applied centrally by modifying the templates and redeploying them across environments. This simplifies maintenance and ensures that all accounts remain aligned with the latest security requirements.

- **Reduction of Operational Complexity:** By automating the onboarding process, the architecture significantly reduces the operational burden on security teams. Tasks that previously required manual configuration are now handled automatically, allowing security analysts to focus on higher-value activities such as threat analysis and incident response.

Overall, the automated onboarding methodology represents a key contribution of this work. It transforms a previously manual and error-prone process into a scalable, standardized, and efficient mechanism, enabling organizations to integrate new cloud environments into their security ecosystem rapidly and reliably.

This approach directly addresses the challenges of tool fragmentation, lack of visibility, and operational complexity identified in previous chapters, and provides a foundation for achieving centralized and effective cloud security in multi-cloud environments.

3.2.6 Data Flow and Security Event Processing

The effectiveness of the proposed security architecture relies on the continuous and structured flow of data across its different components. This section describes how security-relevant data is generated, collected, processed, and analyzed to enable centralized monitoring, threat detection, and incident response.

The overall data flow follows a multi-stage process, beginning within AWS environments and culminating in centralized analysis by the Global Security Operations Center (GSOC).

- **Step 1: Data Generation within AWS Environments** Security-relevant data is continuously generated by various components within AWS tenants. These include infrastructure services, applications, and user activities. Events such as API calls, authentication attempts, network communications, and system-level operations are captured by services such as AWS CloudTrail, VPC Flow Logs, and application logging mechanisms. In addition, workload-level telemetry is generated by security agents such as CrowdStrike Falcon, Qualys, and Prisma Cloud Defender.

- **Step 2: Log Collection and Aggregation in the Security Account** All logs generated across AWS accounts are forwarded to a centralized security account. This account acts as an aggregation layer, where logs are collected, stored, and organized. Centralizing logs within a dedicated account ensures consistency, improves visibility, and enhances security by isolating log storage from operational environments. It also enables uniform enforcement of retention and access policies.

- **Step 3: Data Forwarding to the SIEM Platform** Once aggregated, logs are transmitted to the SIEM platform (Splunk) through secure ingestion pipelines. These pipelines may rely on API integrations, forwarding agents, or cloud-native connectors. The SIEM ingests large volumes of structured and unstructured data from multiple sources, preparing it for analysis.

- **Step 4: Data Normalization and Correlation in the SIEM** Within the SIEM, logs undergo parsing and normalization to extract relevant fields such as timestamps, source identifiers, user identities, and event types. This normalization enables consistent analysis across heterogeneous data sources. Correlation engines and detection rules are then applied to identify suspicious patterns, such as abnormal user behavior, repeated failed authentication attempts, or lateral movement across resources.

- **Step 5: Parallel Security Analysis by CNAPP** In parallel to SIEM processing, the CNAPP platform (Prisma Cloud) continuously analyzes cloud environments to provide a proactive view of the security posture. It evaluates configurations, workloads, and runtime activities in order to identify misconfigurations, vulnerabilities, and compliance issues.

  Prisma Cloud is integrated into the architecture using a hybrid approach that combines direct interaction with cloud services and access to centralized data sources. This enables the platform to enhance its visibility across cloud environments and to analyze both real-time and historical information.

  This approach complements the SIEM integration, where both platforms leverage centralized data while providing different types of analysis. While the

SIEM focuses on log aggregation, event correlation, and incident detection, Prisma Cloud provides contextual insights related to configurations, workloads, and compliance.

Together, these complementary capabilities enable a more comprehensive security strategy, combining reactive detection mechanisms with proactive risk identification. The detailed implementation of this integration is described in the following chapter.

- **Step 6: Alert Generation and Enrichment** When anomalies or policy violations are detected, alerts are generated by both the SIEM and CNAPP platforms. These alerts are enriched with contextual information, such as affected resources, severity levels, and potential impact. This enrichment facilitates faster and more accurate incident investigation.
- **Step 7: Centralized Monitoring and Incident Response by GSOC** All alerts and security insights are centralized within the Global Security Operations Center (GSOC). Security analysts use dashboards and monitoring tools to gain real-time visibility into the system. They investigate alerts, correlate information from multiple sources, and initiate response actions when necessary.
- **End-to-End Security Visibility and Feedback Loop:** The architecture establishes a continuous feedback loop between detection, analysis, and response. Insights gained from incidents can be used to refine detection rules, improve configurations, and strengthen security policies. This iterative process enhances the overall resilience of the system.

By ensuring a structured and automated flow of data from cloud environments to centralized security platforms, the proposed architecture addresses key challenges such as lack of visibility, delayed detection, and fragmented security operations. It enables organizations to achieve comprehensive, scalable, and efficient cloud security monitoring in a multi-cloud context.

3.3 Discussion and Limitations

The proposed security architecture provides a structured and scalable approach to addressing the challenges of securing multi-cloud environments. By combining centralized log collection, SIEM-based detection, and CNAPP-based posture management, the solution enables both reactive and proactive security capabilities. In particular, the introduction of an automated onboarding methodology significantly improves consistency, reduces manual effort, and enhances the scalability of security operations.

One of the main strengths of the proposed approach lies in its ability to provide centralized visibility across distributed cloud environments. By aggregating logs within a dedicated security account and integrating them with security platforms, the architecture addresses one of the key challenges identified in previous chapters, namely the fragmentation of monitoring and analysis capabilities.

Furthermore, the complementary use of SIEM and CNAPP platforms allows for a more comprehensive security strategy. While the SIEM focuses on event correlation and incident detection, CNAPP enables continuous assessment of configurations, workloads, and compliance. This dual approach improves overall security coverage and supports more effective threat detection and prevention.

However, the proposed architecture also presents several limitations.

- **Provider-specific design:** The current implementation focuses on AWS environments, as the architecture relies on provider-specific services, APIs, and configurations. While the design principles are extensible, adapting the solution to other cloud providers, such as Google Cloud Platform (GCP), requires additional development and integration efforts.
- **Dependence on proper configuration:** The effectiveness of the architecture depends on the correct configuration of logging mechanisms, data pipelines, and security integrations. Misconfigurations or incomplete onboarding of cloud environments may lead to gaps in visibility and reduced detection capabilities.
- **Scalability of data processing:** As cloud environments grow, the volume of generated logs increases significantly. Managing, storing, and processing large amounts of data efficiently remains a challenge, particularly for SIEM platforms, which may incur high operational and financial costs.
- **Integration complexity:** Although the architecture aims to unify multiple security tools, integrating heterogeneous platforms such as SIEM and CNAPP remains complex. Differences in data formats, APIs, and operational models require careful configuration and ongoing maintenance.
- **Limited coverage of multi-cloud environments:** While the architecture is designed with extensibility in mind, the current implementation does not fully address all aspects of multi-cloud integration. In particular, environments based on other cloud providers are not yet fully integrated into the automated onboarding process.

Despite these limitations, the proposed architecture provides a solid foundation for scalable and centralized cloud security. It demonstrates how automation, standardization, and integration can significantly improve the effectiveness of security operations in complex cloud environments.

Chapter 4

Implementation

4.1 Implementation Design Decisions

4.1.1 Log Storage Design and Implementation

As illustrated in Figure 6.1, all logs are centrally stored within the security account before being ingested by GSOC tools such as the SIEM and CNAPP platforms. This centralized storage layer is implemented using Amazon S3, which provides a scalable, durable, and secure object storage solution adapted to large-scale log management.

Storage Strategy

To optimize storage costs while maintaining high availability and performance, the proposed architecture leverages the **S3 Intelligent-Tiering** storage class. This mechanism automatically moves objects between different access tiers based on usage patterns, without requiring manual lifecycle configuration.

This approach is particularly suitable for log storage, as access patterns are unpredictable. Logs may need to be accessed at any time for forensic analysis, incident investigation, or compliance purposes. By dynamically optimizing storage costs while preserving millisecond access latency, S3 Intelligent-Tiering provides an efficient balance between cost and performance.

Retention and Lifecycle Management

To control storage growth and ensure compliance with operational requirements, a log retention policy of **90 days** is defined. Logs are initially stored in the standard access tier and automatically transitioned to lower-cost tiers through Intelligent-Tiering.

This strategy avoids complex manual lifecycle configurations while ensuring that logs remain accessible when needed. It also supports cost optimization without compromising operational efficiency.

Security and Confidentiality

Ensuring the confidentiality and integrity of log data is a critical requirement for the proposed architecture.

- **Encryption at Rest:** Logs stored in S3 are encrypted using server-side encryption (SSE-S3) based on AES-256 encryption. Each object is encrypted with a unique key, which is managed and regularly rotated by AWS, ensuring strong protection against unauthorized access.
- **Encryption in Transit:** All log transfers between AWS services and the S3 bucket are secured using TLS. Additionally, strict bucket policies enforce that all PUT and GET operations must use secure transport (HTTPS), preventing unencrypted data access.
- **Access Control:** Access to the log storage is restricted to authorized services and security tools such as the SIEM and CNAPP platforms. Authentication mechanisms and IAM roles ensure that only trusted entities can access log data.

Integrity and Tamper Protection

Maintaining the integrity of logs is essential for reliable forensic analysis and compliance.

- **CloudTrail Integrity Validation:** CloudTrail provides built-in log file integrity validation through cryptographic hashing and digital signatures. This enables verification that logs have not been modified after delivery.
- **VPC Flow Logs Integrity:** VPC Flow Logs include checksums for detecting transmission errors. However, these mechanisms do not fully protect against intentional tampering. Additional integrity validation mechanisms can therefore be considered as part of future improvements.

Availability and Durability

Amazon S3 provides built-in high availability and durability guarantees. Data is automatically replicated across multiple availability zones within a region, ensuring resilience against infrastructure failures.

S3 is designed to deliver up to **99.99% availability** and **99.999999999% durability** (11 nines), making it highly suitable for critical log storage. This ensures that logs remain accessible even in the event of partial system failures.

Discussion

The proposed log storage architecture provides a secure, scalable, and cost-efficient solution for managing large volumes of security data. By combining centralized storage, automated tiering, strong encryption, and high durability, it ensures that logs are protected, accessible, and compliant with security requirements.

However, the approach also introduces challenges related to long-term storage costs and integrity guarantees for certain log types. These aspects will be further analyzed and improved in the implementation and evaluation phases.

4.1.2 API Call Logging Strategy and Design

In cloud environments, monitoring API activity is a fundamental requirement for ensuring security, governance, and operational visibility. In AWS, this functionality is provided by **AWS CloudTrail**, a native service that records API calls performed by users, services, and applications across the entire infrastructure.

CloudTrail logs provide detailed information about each API request, including the identity of the caller, the time of the request, the source IP address, the action performed, and the affected resources. These logs constitute a primary data source for detecting unauthorized activities, investigating incidents, and ensuring compliance with security policies.

Control Plane vs Data Plane

A key concept in understanding CloudTrail logging is the distinction between the **control plane** and the **data plane**, as they represent two fundamentally different types of operations within cloud environments.

- **Control Plane (Management Events):** The control plane refers to operations that manage cloud resources and configurations. These include API calls such as creating or deleting resources, modifying configurations, assigning permissions, and managing identities. Examples include creating IAM roles, launching EC2 instances, modifying security groups, or updating network configurations. These actions directly impact the structure and security posture of the environment.
- **Data Plane (Data Events):** The data plane refers to operations performed on the resources themselves. These include actions such as reading or writing

data to storage services (e.g., S3 object access), querying databases, or interacting with application-level services. Data plane events are typically more granular and occur at a much higher frequency than control plane events.

This distinction is critical because it directly impacts both the **security relevance** and the **volume of generated logs**. While control plane events provide high-value information related to configuration and access changes, data plane events generate significantly larger volumes of logs and require more advanced filtering and processing mechanisms.

Logging Strategy and Design Choices

Given these characteristics, the proposed architecture adopts a phased logging strategy, prioritizing **management events (control plane)** as the primary source of security monitoring.

This decision is motivated by several factors:

- **High Security Value:** Control plane events capture administrative actions that can directly impact the security of the environment. Unauthorized modifications to IAM policies, network configurations, or resource provisioning can lead to severe security breaches.

- **Governance and Compliance:** These events are essential for auditing and demonstrating compliance with security standards, as they provide a complete trace of configuration changes and administrative activities.

- **Reduced Data Volume:** Compared to data plane events, management events generate a lower volume of logs, making them easier to process, store, and analyze within SIEM platforms.

- **Detection of Critical Threats:** Many advanced attack techniques, such as privilege escalation, persistence mechanisms, or lateral movement, involve control plane operations that can be detected through management event analysis.

While data plane events provide valuable insights into application-level activity, their high volume introduces challenges in terms of storage, processing cost, and signal-to-noise ratio. Therefore, their integration is considered as a secondary phase, to be implemented selectively based on use cases such as sensitive data access monitoring.

CloudTrail Event Types

CloudTrail supports multiple categories of events, each serving different monitoring purposes:

- **Management Events:** These correspond to control plane operations and represent the core focus of the proposed logging strategy.
- **Data Events:** These provide visibility into resource-level operations, such as S3 object access or Lambda execution. Due to their high volume, they must be carefully scoped and selectively enabled.
- **Insights Events:** These events are generated by CloudTrail Insights and highlight unusual API activity patterns, such as sudden spikes in usage or abnormal behavior.
- **Network Events:** These provide limited visibility into network-level activities and are less commonly used compared to other event types.

In this work, the architecture focuses primarily on management events, with the possibility of extending coverage to data events in future iterations.

Organizational CloudTrail Architecture

To ensure consistent and centralized logging across all AWS environments, the architecture leverages an **organization-wide CloudTrail configuration**.

Within the AWS Organization, a dedicated **security account** is designated as a **delegated administrator**. This account is responsible for configuring and managing CloudTrail at the organization level, enabling centralized control without relying on the management account for operational tasks.

The architecture follows several key design principles:

- **Delegated Administration Model:** The security account is granted delegated administrator permissions, allowing it to configure CloudTrail across all accounts. This improves operational flexibility while maintaining centralized governance.
- **Organization Trail:** A single organization trail is created, automatically enabling logging for all existing and newly created accounts. This ensures consistent security coverage across the entire organization.
- **Multi-Region Logging:** The trail is configured as a multi-region trail to capture API activity across all AWS regions, preventing visibility gaps in geographically distributed environments.

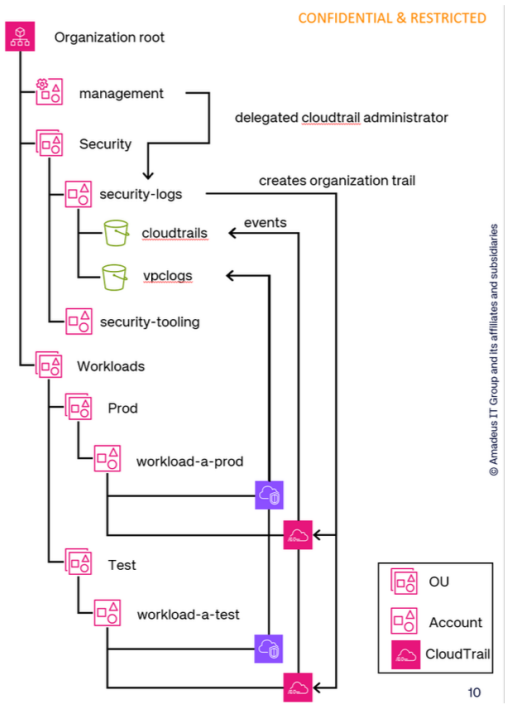

Figure 4.1: Organization-wide CloudTrail architecture

- **Centralized Log Storage:** All logs are aggregated into a centralized S3 bucket located in the security account. This design simplifies data access, improves visibility, and facilitates integration with external security platforms such as SIEM and CNAPP.
- **Automatic Account Onboarding:** New accounts added to the organization are automatically included in the logging configuration, ensuring continuous coverage without manual intervention.

Security and Governance Considerations

Deploying CloudTrail at the organizational level introduces several important security and governance considerations:

- **Separation of Responsibilities:** The use of a dedicated security account ensures that logging and monitoring capabilities are isolated from operational workloads, reducing the risk of tampering or accidental misconfiguration.
- **Trail Ownership and Persistence:** The organization trail is owned by the management account, ensuring that logging remains active even if delegated

administrator permissions are modified or revoked.

- **Centralized Control and Standardization:** By enforcing a single logging configuration across all accounts, the architecture reduces inconsistencies and ensures compliance with organizational policies.
- **Scalability:** The design must handle increasing volumes of API activity as the number of accounts and workloads grows, requiring efficient storage and processing mechanisms.

Access Control and Permissions

Proper configuration of access permissions is critical to ensure secure and reliable log collection.

The delegated administrator account must be granted permissions to:

- Configure CloudTrail at the organization level
- Interact with AWS Organizations to enable service integration
- Manage S3 storage configurations, including encryption and access policies

These permissions allow CloudTrail to automatically enable logging across all accounts, securely deliver logs to centralized storage, and maintain consistent configurations over time.

Discussion

The proposed API logging strategy provides a scalable and security-focused approach to monitoring cloud environments. By prioritizing control plane events and leveraging an organization-wide CloudTrail configuration, the architecture ensures high-value visibility into critical activities while maintaining manageable data volumes.

This approach establishes a strong foundation for threat detection, governance, and compliance. However, it also highlights trade-offs between visibility and scalability, particularly when considering the integration of high-volume data plane events. Future work may focus on selective data event logging and advanced analytics to further enhance detection capabilities.

4.1.3 VPC Flow Logs Design

VPC Flow Logs are a native AWS feature that enables the collection of network traffic metadata within a Virtual Private Cloud (VPC). They provide visibility into both inbound (ingress) and outbound (egress) traffic at the level of network

interfaces, subnets, or entire VPCs. Unlike packet capture tools, flow logs do not record payload data, but instead capture metadata about network communications, making them lightweight and scalable for large cloud environments.

Each flow log record represents a summary of traffic observed during a defined aggregation interval. These records include key attributes such as source and destination IP addresses, ports, protocol, number of packets, number of bytes, and the action taken (ACCEPT or REJECT). This information is essential for monitoring network behavior, detecting anomalies, and supporting forensic investigations.

Flow Logs Architecture and Centralization

In the proposed architecture, VPC Flow Logs are configured at the VPC level within each AWS account, as AWS does not currently support enabling flow logs at the organizational level. Consequently, flow logs must be activated individually for each VPC across all accounts.

As illustrated in Figure 4.2, logs generated within each workload account are forwarded to a centralized storage location located in the security account. This design ensures centralized visibility across all environments while maintaining separation between production workloads and logging infrastructure. This centralized

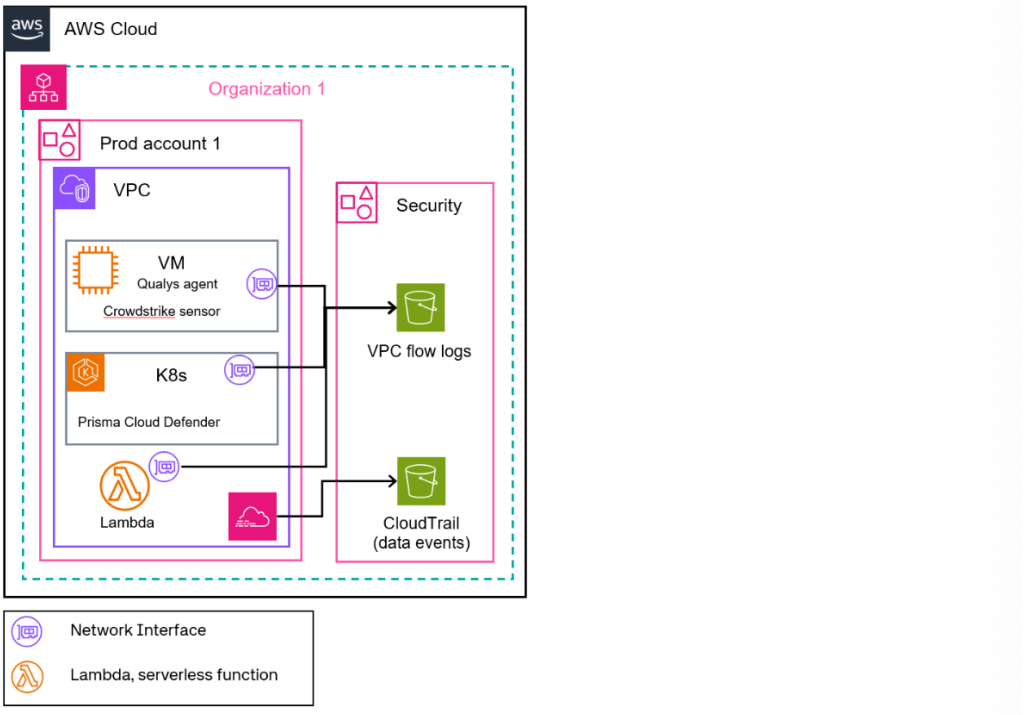

Figure 4.2: Centralized VPC Flow Logs architecture

approach simplifies log management, improves monitoring capabilities, and enables integration with external security platforms such as SIEM and CNAPP solutions.

Traffic Visibility and Coverage

Flow logs capture traffic associated with Elastic Network Interfaces (ENIs), which are attached to various resources such as EC2 instances, load balancers, and container workloads. This enables visibility into communication patterns between components within the cloud environment.

However, certain limitations must be considered. For example, serverless services such as AWS Lambda may not always generate flow logs if they operate outside of a customer-managed VPC. In such cases, network visibility must be complemented by other logging mechanisms, such as CloudTrail or application logs.

Capture Window and Aggregation

Each flow log record is generated based on a defined aggregation interval, known as the capture window. AWS supports two main configurations:

- **10-minute aggregation (default):** This configuration generates fewer records and reduces storage costs, but provides lower temporal resolution. It is suitable for general monitoring and trend analysis.
- **1-minute aggregation:** This configuration provides higher granularity and more detailed visibility into network activity, which is beneficial for detecting short-lived anomalies or attacks. However, it significantly increases log volume and storage requirements.

In this project, the default 10-minute aggregation interval is selected as a trade-off between visibility and cost efficiency.

Log Format and Storage

Flow log records are aggregated and delivered periodically to a storage destination, typically Amazon S3. Logs are compressed using Gzip format to reduce storage consumption and can be stored in different formats:

- **Plain text format:** widely supported and compatible with most security tools.
- **Parquet format:** optimized for analytics and querying, but not always supported by external platforms.

In this architecture, the text format is selected to ensure compatibility with downstream tools such as the SIEM and CNAPP platforms.

Limitations of VPC Flow Logs

Despite their usefulness, flow logs have several inherent limitations that must be considered when designing a security monitoring strategy:

- **No payload visibility:** Flow logs do not capture packet contents, limiting their ability to detect application-layer attacks.
- **Incomplete traffic coverage:** Certain types of traffic are not captured, including:
  - Traffic between instances using the same network interface
  - Traffic to the instance metadata service (169.254.169.254)
  - DHCP and ARP traffic
  - Traffic through some VPC endpoints
- **NAT Gateway visibility limitations:** When traffic passes through a NAT Gateway, the source or destination IP addresses may be altered. Without additional configuration, flow logs may display the NAT IP instead of the original client IP.

Example: NAT Gateway Traffic Interpretation

The behavior of flow logs in NAT scenarios is illustrated in Figure 4.3. When client IP preservation is not enabled, the flow logs may show the NAT Gateway IP instead of the original source or destination.

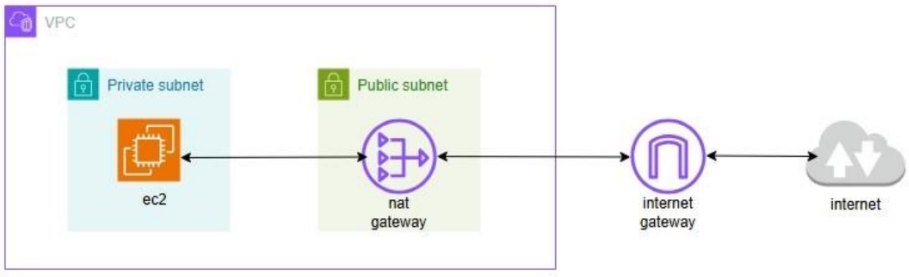

Figure 4.3: Traffic flow through NAT Gateway

Understanding these behaviors is critical for accurate analysis of network activity and avoiding misinterpretation of security events.

Discussion

VPC Flow Logs provide essential visibility into network-level activity within AWS environments. When combined with centralized storage and integrated with SIEM and CNAPP platforms, they form a critical component of the overall security monitoring strategy. However, their limitations require complementary logging mechanisms to achieve comprehensive security coverage.

4.1.4 Research on Syslog Integration

In addition to cloud-native logging mechanisms such as AWS CloudTrail and VPC Flow Logs, it is essential to collect logs generated at the operating system and application levels within compute resources. These logs provide detailed insights into system behavior, application execution, user activity, and potential security incidents that are not visible through infrastructure-level monitoring alone.

To achieve this, the Syslog protocol is commonly used as a standardized mechanism for transmitting log messages from distributed systems to centralized logging platforms. Syslog is widely adopted due to its simplicity, compatibility with multiple systems, and ability to support both TCP and UDP transport mechanisms.

Within the context of this project, Syslog serves as a key component for ingesting workload-level logs into the SIEM platform (Splunk). These logs complement cloud-native logs by providing fine-grained visibility into events occurring inside virtual machines, such as authentication attempts, process execution, system errors, and application-specific logs.

Role of Syslog in the Security Architecture

Syslog plays a critical role in achieving comprehensive observability across the cloud environment. While CloudTrail focuses on API-level activity and VPC Flow Logs provide network-level visibility, Syslog enables monitoring at the host and application levels.

This multi-layered logging strategy is essential for effective threat detection, as many security incidents originate within workloads and may not be reflected in infrastructure logs alone. For example, malicious activities such as privilege escalation, unauthorized process execution, or application-layer attacks can only be detected through system-level logs.

By integrating Syslog into the SIEM platform, the architecture ensures that security analysts have access to a complete and correlated view of events across all layers of the system.

Initial Architecture: Centralized Syslog Collector

The initial design considered for Syslog integration was based on deploying a centralized Syslog collector within the security account of each AWS tenant. This architecture is illustrated in Figure 4.4.

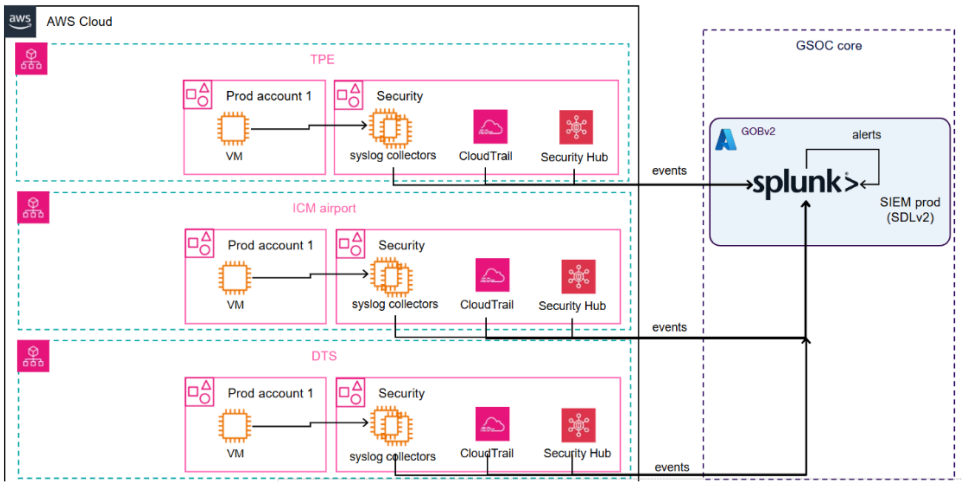

Figure 4.4: Initial Syslog architecture with centralized collector

In this model, virtual machines deployed across multiple workload accounts would forward their Syslog messages to a centralized collector hosted within the security account. The collector would act as an intermediary layer responsible for aggregating, buffering, and forwarding logs to the SIEM platform.

This approach aligns with traditional on-premise logging architectures, where centralized log collectors are commonly used to manage and control log ingestion pipelines.

Advantages of the Centralized Collector Approach

The centralized collector architecture offers several technical benefits:

- **Aggregation and normalization:** Logs from multiple sources can be aggregated in a single location, allowing preprocessing, filtering, and normalization before ingestion into the SIEM.
- **Buffering and reliability:** The collector can temporarily store logs in case of network disruptions or SIEM unavailability, reducing the risk of data loss.
- **Controlled ingestion pipeline:** The architecture enables better control over log flow, including rate limiting, transformation, and enrichment of log data.

- **Reduced direct exposure of SIEM endpoints:** Only the collector communicates with the SIEM, limiting the number of direct connections and potentially improving security.

Despite these advantages, further analysis revealed several critical limitations.

Limitations and Challenges

The centralized Syslog collector approach introduces significant challenges, particularly in a cloud-native, multi-account environment:

- **Operational complexity:** The collector infrastructure must be deployed, configured, monitored, and maintained. This includes managing virtual machines, applying security patches, and ensuring high availability.
- **Scalability constraints:** As the number of workloads increases, the volume of Syslog messages grows significantly. To handle this load, additional components such as load balancers and auto-scaling groups are required, increasing architectural complexity.
- **Network dependencies:** Each workload VPC must establish secure connectivity with the centralized collector. This requires consistent network configuration across all accounts, which is difficult to guarantee in environments shaped by mergers and acquisitions.
- **Single point of failure risk:** Although mitigations can be implemented, the collector remains a critical component whose failure could disrupt log collection.
- **Long-term maintainability:** The architecture introduces dependencies on infrastructure that must be maintained beyond the duration of the project. This raises concerns regarding ownership, operational costs, and sustainability.

These limitations make the centralized collector approach less suitable for a scalable and dynamic cloud environment.

Alternative Architecture: Direct Syslog Forwarding

To overcome these challenges, an alternative approach was considered, based on direct Syslog forwarding from workloads to the SIEM platform. This architecture is illustrated in Figure 4.5.

In this model, each virtual machine is responsible for sending its Syslog messages directly to the SIEM through a secure and private connection.

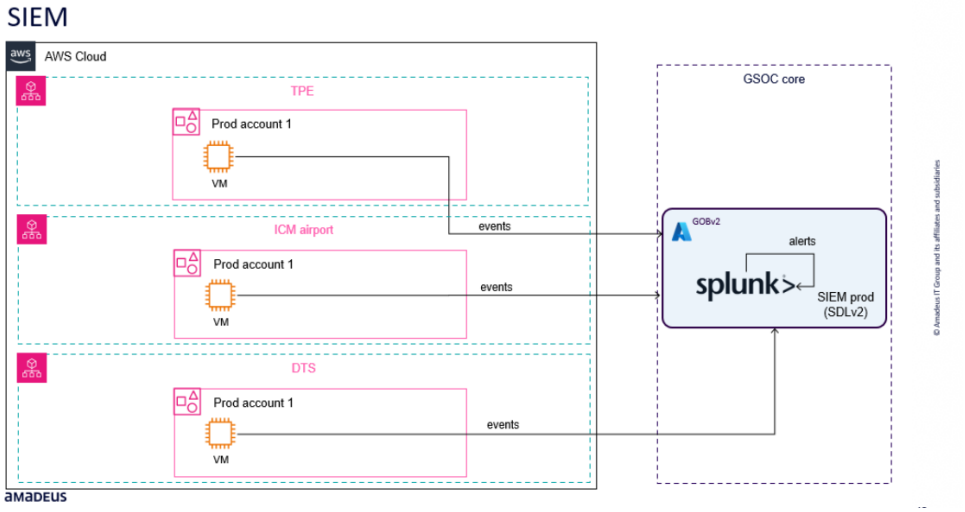

Figure 4.5: Direct Syslog forwarding to SIEM platform

Advantages of Direct Forwarding

This approach offers several advantages aligned with cloud-native design principles:

- **Reduced architectural complexity:** Eliminates the need for intermediate components such as collectors, load balancers, and scaling mechanisms.
- **Improved scalability:** Log forwarding is distributed across all workloads, preventing bottlenecks and enabling horizontal scalability.
- **Lower operational overhead:** No additional infrastructure needs to be deployed or maintained, reducing long-term operational burden.
- **Faster deployment:** Integration can be achieved by configuring Syslog agents on each workload, simplifying onboarding.

Limitations of Direct Forwarding

However, this approach also introduces certain trade-offs:

- **Dependence on network connectivity:** Each workload must maintain a reliable connection to the SIEM platform.
- **Limited buffering capability:** In case of connectivity issues, logs may be lost unless local buffering mechanisms are implemented.
- **Increased number of connections:** The SIEM platform must handle connections from multiple sources, which may require careful scaling and configuration.

Design Decision and Architectural Alignment

After evaluating both approaches, the centralized Syslog collector architecture was not retained due to its complexity, scalability limitations, and operational overhead.

The direct forwarding model was selected as it better aligns with the key objectives of this project:

- **Scalability:** supports dynamic and growing environments
- **Simplicity:** reduces architectural and operational complexity
- **Automation compatibility:** easier to integrate within Infrastructure-as-Code onboarding processes
- **Sustainability:** minimizes long-term maintenance requirements

This decision reflects a broader design philosophy adopted throughout this work: prioritizing lightweight, scalable, and cloud-native approaches over traditional, infrastructure-heavy solutions.

Discussion

Syslog integration is a critical component for achieving full visibility into workload-level activities in cloud environments. It complements cloud-native logging mechanisms by providing detailed insights into system and application behavior.

The analysis conducted in this section highlights the importance of evaluating not only the technical capabilities of a solution but also its scalability, operational impact, and long-term sustainability.

By selecting a direct forwarding approach, the proposed architecture ensures efficient, scalable, and maintainable integration of Syslog data into the centralized security ecosystem.

4.2 Deployment Techniques and Implementation Approaches

4.2.1 Overview of Deployment Approaches

After defining the architectural design and key implementation decisions, the next step consists in determining how the proposed security components can be effectively deployed across AWS environments. The deployment strategy plays a critical role in ensuring that the architecture is not only functional, but also scalable, reproducible, and maintainable over time.

In a multi-account cloud environment, particularly in large organizations, manual configuration quickly becomes impractical due to the number of resources involved and the need for consistency across tenants. Therefore, several deployment approaches were evaluated in this work, ranging from manual configuration methods to fully automated Infrastructure-as-Code (IaC) solutions.

The objective of this evaluation is to identify the most suitable approach for deploying and managing the security account and its associated components, including centralized log storage, logging configurations, and access control mechanisms. Each approach is analyzed based on the following criteria:

- **Scalability:** Ability to support deployment across multiple accounts and environments.
- **Repeatability:** Capability to reproduce the same configuration reliably.
- **Maintainability:** Ease of updating and managing configurations over time.
- **Security:** Compliance with best practices and reduction of human error.
- **Operational overhead:** Effort required to deploy and maintain the solution.

The following subsections present the different approaches that were considered, along with their advantages and limitations.

4.2.2 Manual Deployment using AWS Management Console

The most straightforward approach to deploying cloud resources consists of using the AWS Management Console. This graphical interface allows users to create and configure services interactively, without requiring programming knowledge.

In this approach, the security account and its associated resources, such as S3 buckets for log storage, lifecycle policies, and access controls, are configured manually through the console interface. For example, creating a log storage bucket involves selecting the appropriate region, defining access policies, enabling encryption, and configuring lifecycle rules directly through the user interface.

This method is particularly useful for initial experimentation, testing, or for users who are not familiar with automation tools. It provides an intuitive way to understand AWS services and their configurations.

However, despite its simplicity, this approach presents several significant limitations in the context of a multi-account security architecture. First, manual configuration is highly error-prone, as it relies on human interaction for each step. Small inconsistencies in configuration can lead to security gaps or operational issues. Second, it lacks repeatability, making it difficult to reproduce the same

setup across multiple environments. Third, it does not provide version control or rollback capabilities, which are essential for managing infrastructure changes in a controlled manner.

As a result, while the AWS Management Console is useful for prototyping and learning purposes, it is not suitable for deploying production-grade security architectures that require consistency and scalability.

4.2.3 Semi-Automated Deployment using AWS CloudShell

An alternative approach is to use AWS CloudShell, which provides a browser-based command-line interface for interacting with AWS services using the AWS CLI. This method enables partial automation through scripting, allowing users to execute predefined commands to create and configure resources.

In this project, CloudShell was used to explore the possibility of automating the creation of log storage buckets and lifecycle configurations using shell scripts. By defining a sequence of AWS CLI commands, it is possible to deploy resources more quickly than through the console, while reducing manual effort.

This approach offers several advantages compared to manual configuration. It improves deployment speed and allows basic automation of repetitive tasks. Additionally, scripts can be reused and shared, providing a degree of standardization across environments.

However, CloudShell-based deployment still presents important limitations. Although scripts automate execution, they do not provide native support for state management, meaning that the system does not track the current state of deployed resources. As a result, it becomes difficult to detect configuration drift or manage updates safely. Furthermore, rollback mechanisms are not inherently supported, increasing the risk of inconsistencies in case of errors.

In large-scale environments, these limitations become critical, as maintaining consistency across multiple accounts requires more robust and controlled deployment mechanisms. Therefore, while CloudShell improves upon manual deployment, it does not fully meet the requirements of scalability and maintainability.

4.2.4 Infrastructure-as-Code using AWS CloudFormation

Infrastructure-as-Code (IaC) represents a more advanced and robust approach to cloud deployment. AWS CloudFormation enables the definition of infrastructure resources using declarative templates written in YAML or JSON. These templates describe the desired state of the infrastructure, and AWS automatically provisions and manages the resources accordingly.

In this approach, all components of the security architecture, including S3 buckets, access policies, encryption settings, and lifecycle configurations, are defined

within CloudFormation templates. Once defined, these templates can be deployed consistently across multiple environments, ensuring that all configurations adhere to the same standards.

CloudFormation provides several key advantages. First, it ensures repeatability, as the same template can be used to deploy identical infrastructures. Second, it supports version control, allowing changes to be tracked and managed over time. Third, it includes built-in rollback mechanisms, which automatically revert changes in case of deployment failures. This significantly reduces the risk of inconsistent or partially configured environments.

Additionally, CloudFormation is a managed AWS service, meaning that it integrates seamlessly with other AWS components and follows AWS security best practices. This makes it particularly suitable for deploying security-critical infrastructure.

Given these advantages, CloudFormation was selected as the primary deployment method for this project. It provides the necessary level of automation, reliability, and scalability required for implementing the proposed security architecture.

4.2.5 Infrastructure-as-Code using Terraform

Terraform is another widely used Infrastructure-as-Code tool that allows the definition and deployment of infrastructure across multiple cloud providers. It uses a declarative language (HCL) to describe resources and supports advanced features such as state management and modular configurations.

Terraform was evaluated as a potential solution due to its flexibility and multi-cloud capabilities. In theory, it would allow the extension of the architecture to other cloud providers beyond AWS, aligning with the long-term objective of supporting multi-cloud environments.

However, several practical constraints limited its applicability in this project. First, the use of Terraform requires access to a properly configured execution environment, including the installation of the Terraform CLI and management of state files. In the context of this work, organizational constraints such as proxy restrictions and limited permissions prevented the use of local or managed Terraform environments.

Additionally, the chosen deployment approach involved running Terraform from a virtual machine, which introduces additional operational overhead. Maintaining this instance requires ongoing management, including security patching, access control, and monitoring. This contradicts the objective of minimizing operational complexity.

As a result, despite its strengths, Terraform was not selected as the primary deployment tool in this project.

4.2.6 Comparison and Final Decision

Based on the evaluation of the different deployment approaches, a comparative analysis was conducted to determine the most suitable solution for implementing the proposed architecture.

Manual deployment using the AWS Management Console was found to be simple but unsuitable for large-scale environments due to its lack of automation and susceptibility to human error. CloudShell-based deployment improves efficiency through scripting but lacks essential features such as state management and rollback capabilities.

Infrastructure-as-Code solutions provide a more robust and scalable approach. Among them, AWS CloudFormation offers strong integration with AWS services, built-in security mechanisms, and full support for lifecycle management of resources. Terraform, while powerful and flexible, was not compatible with the operational constraints of the project.

Consequently, AWS CloudFormation was selected as the preferred deployment approach. It provides the best balance between automation, security, scalability, and maintainability, making it well-suited for implementing and managing the proposed security architecture in AWS environments.

The following table summarizes the comparative evaluation of the different deployment approaches based on the criteria defined previously.

Table 4.1: Comparison of Deployment Techniques

Approach	Advantages	Limitations	Decision
AWS Management Console	User-friendly interface, no coding required, suitable for initial testing and exploration	Prone to human errors, not scalable, lacks repeatability, no version control or rollback mechanisms	Not selected
AWS CloudShell (CLI Scripts)	Faster than manual configuration, enables basic automation, reusable scripts	No state management, no rollback mechanisms, limited scalability for large environments	Not selected
AWS CloudFormation	Infrastructure-as-Code support, repeatable deployments, built-in rollback, version control, strong integration with AWS services	Requires initial learning effort, templates can become complex	Selected
Terraform	Multi-cloud support, strong state management, modular and scalable	Requires additional infrastructure, operational overhead, constrained by organizational limitations	Not selected

4.2.7 Secure Storage Configuration and Bucket Policies

In order to ensure the confidentiality, integrity, and controlled accessibility of audit logs, a comprehensive security configuration was implemented at the storage layer. Amazon S3 was selected as the centralized storage service for all logs, including AWS CloudTrail events, VPC Flow Logs, and additional security-relevant data sources. Although Amazon S3 provides high durability and scalability, its default configuration allows broad access within the account, which is not appropriate for a security-critical logging architecture.

To address these limitations, a set of fine-grained bucket policies was defined to strictly regulate access to the storage. These policies enforce a least privilege model and explicitly control write (`PutObject`), read (`GetObject`), and listing (`ListBucket`) operations. The configuration relies on a combination of explicit deny and allow statements, enriched with conditional constraints based on AWS context keys such as `aws:SourceOrgID`, `aws:SourceArn`, and `aws:PrincipalArn`.

Restricting Write Access (PutObject)

Controlling access to the storage is a critical requirement in order to preserve the integrity and trustworthiness of collected logs. By default, any IAM principal with sufficient permissions can perform operations in an S3 bucket. In a centralized logging architecture, such behavior introduces significant risks, including unauthorized data injection and potential corruption of audit trails.

To mitigate these risks, an explicit deny policy was implemented to block all `PutObject` and `GetObject` operations that do not originate from authorized AWS services or the expected organizational context. This deny rule applies to all principals and ensures that any request failing to meet the defined conditions is automatically rejected, regardless of other permissions that may be granted elsewhere.

In parallel, a complementary allow policy was defined to grant write access exclusively to AWS log delivery services, such as `cloudtrail.amazonaws.com` and `delivery.logs.amazonaws.com`. These permissions are further constrained through conditional expressions that enforce strict validation of the request origin. In particular:

- The `aws:SourceOrgID` condition ensures that only requests originating from the designated AWS Organization are accepted.
- The `aws:SourceArn` condition restricts access to specific AWS resources, such as predefined CloudTrail trails.
- The requirement for the `s3:x-amz-acl` parameter (e.g., `bucket-owner-full-control`) guarantees proper ownership and control over uploaded objects.

This layered approach ensures that only trusted AWS services can deliver logs to the storage while preventing any unauthorized entity from injecting or modifying data.

```
# Put by default is allowed for all iams within the account
# Deny put for all (within the owner account) execpt the org trail
- Sid: AWSDenyPutObjectFromEveryoneElse
  Effect: Deny
  Principal: '*'
  Action: 's3:PutObject'
  Resource:
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}"
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}/*"
  Condition:
    StringNotEquals:
      aws:SourceArn: !Sub "arn:aws:cloudtrail:eu-west-1:${ManagementAccountId}:trail/amadeus-soc-management-trail"
# Allow put only for the org trail
- Sid: AWSCloudTrailWrite20150319
  Effect: Allow
  Principal:
    Service: cloudtrail.amazonaws.com
  Action: 's3:PutObject'
  Resource:
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}"
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}/*"
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}/AWSLogs/${ManagementAccountId}/*"
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}/AWSLogs/${OrgId}/*"
  Condition:
    StringEquals:
      s3:x-amz-acl: bucket-owner-full-control
      aws:SourceArn: !Sub "arn:aws:cloudtrail:eu-west-1:${ManagementAccountId}:trail/amadeus-soc-management-trail"
```

Figure 4.6: API Calls Bucket Policy

```
# Deny for all execpt for aws services principal of the org
- Sid: AWSDenyPutObjectFromEveryoneElse
  Effect: Deny
  Principal: '*'
  Action:
    - s3:PutObject
    - s3:GetObject # in the console for example we can get the object since the iam has got permissions, need this
  Resource: !Sub "arn:aws:s3:::amadeus-soc-logs-flowlogs-${OrgId}/*"
  Condition:
    StringNotEquals:
      aws:SourceOrgID: !Sub "${OrgId}"

# Allow just for the delivery.logs.amazonaws.com of the allowed account
- Sid: AWSLogsDeliveryWrite
  Effect: Allow
  Principal:
    Service: delivery.logs.amazonaws.com
  Action: 's3:PutObject'
  Resource: !Sub "arn:aws:s3:::amadeus-soc-logs-flowlogs-${OrgId}/*"
  Condition:
    StringEquals:
      aws:SourceOrgID: !Sub "${OrgId}"
      s3:x-amz-acl: bucket-owner-full-control
```

Figure 4.7: Flow Logs Bucket Policy

Controlled Read Access (GetObject)

Access to stored logs must be tightly controlled to prevent unauthorized disclosure of sensitive information. In the proposed architecture, only a limited set of security tools, such as Splunk and Prisma Cloud, require read access to the stored data.

To enforce this requirement, a bucket policy was defined to allow `GetObject` operations exclusively for a predefined set of IAM roles. Access control is implemented using the `aws:PrincipalArn` condition, which restricts permissions to explicitly identified principals. This ensures that only authorized roles associated

with security monitoring and analysis platforms can retrieve log data.

By enforcing access restrictions at the storage layer, this configuration provides a strong guarantee that sensitive audit logs remain protected from unauthorized access while still being accessible to the necessary processing components.

```
# Get by default is allowed for all iams within the account
# Allow Prisma Cloud Role and Splunk to get objects
- Sid: AWSAllowGetPrismaCloud
  Effect: Allow
  Principal: '*'
  Action: 's3:GetObject'
  Resource:
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}"
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}/*"
  Condition:
    StringEquals:
      "aws:PrincipalArn": [
          "arn:aws:iam::707058398158:role/PrismaCloudRole-807098488675427328",
          "arn:aws:iam::482849952454:role/Kinesis-Test-Role",
          "arn:aws:iam::082886357466:role/TCH-PII-NPCI-STG-AtlasMongo-s3-splunk-Role"
      ]
```

Figure 4.8: Read Access Policy for Security Tools

Restricting Listing Operations (ListBucket)

In addition to controlling read and write operations, it is necessary to regulate the ability to list objects within the bucket. Listing operations may expose metadata about stored logs, such as object structure, naming conventions, and timestamps, which can provide valuable information to unauthorized entities.

To prevent such exposure, a policy was defined to restrict `ListBucket` operations based on organizational boundaries. This is achieved through a combination of explicit deny and allow statements:

- An explicit deny rule prevents any principal outside the AWS Organization from listing bucket contents.
- An allow rule grants listing permissions only to principals that belong to the designated organization.

The enforcement of this restriction relies on the `aws:PrincipalOrgID` condition, which validates the organizational membership of the requesting entity. This mechanism ensures that only trusted entities within the organization can enumerate objects stored in the bucket.

```yaml
# Deny ListBucket objects from everyone outside the organization
- Sid: AWSDenyListBucketForExternalAccounts
  Effect: Deny
  Principal: "*"
  Action: 's3:ListBucket'
  Resource:
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}"
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}/*"
  Condition:
    StringNotEquals:
      "aws:PrincipalOrgID": !Sub "${OrgId}" # even if the principal is not member of any organization, he will be denied access
# Allow ListBucket objects for whoever member of the organization
- Sid: AWSAllowListForAccountsInTheOrg
  Effect: Allow
  Principal: "*"
  Action: 's3:ListBucket'
  Resource:
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}"
    - !Sub "arn:aws:s3:::amadeus-soc-logs-cloudtrail-mgt-${OrgId}/*"
  Condition:
    StringEquals:
      aws:PrincipalOrgID: !Sub "${OrgId}"
```

Figure 4.9: ListBucket Restriction Policy

Security Guarantees

The combination of the previously described policies establishes a robust and comprehensive security posture for the storage layer. Specifically, the implemented configuration ensures:

- **Confidentiality:** Access to log data is restricted to explicitly authorized principals.
- **Integrity:** Unauthorized entities are prevented from injecting or modifying stored data.
- **Authentication:** All requests are validated based on IAM identities and organizational context.
- **Least privilege enforcement:** Permissions are granted only where strictly necessary.

Furthermore, the use of explicit deny rules guarantees that restrictive conditions cannot be bypassed by broader permissions defined elsewhere, reinforcing the overall security model.

Summary

The implementation of fine-grained S3 bucket policies constitutes a fundamental component of the proposed logging architecture. By enforcing strict control over write, read, and listing operations, the storage layer ensures that all collected logs

are securely stored, protected from unauthorized access, and accessible only to designated security services.

This secure foundation is essential for enabling reliable downstream processing and analysis, including the ingestion of logs into the SIEM platform described in the subsequent sections.

4.2.8 Audit Log Ingestion into Splunk

The integration of AWS audit logs into the SIEM platform is a critical component of the implementation, enabling centralized monitoring, event correlation, and incident detection. This section describes the implementation of the log ingestion pipeline, including the architecture, design rationale, and security considerations.

During the implementation phase, multiple ingestion approaches were evaluated. A basic S3 polling mechanism was initially considered due to its simplicity. However, this approach revealed several limitations, including increased latency, tight coupling between components, and inefficiencies in handling large volumes of logs.

To address these limitations, an event-driven architecture based on Amazon Simple Notification Service (SNS) and Amazon Simple Queue Service (SQS) was implemented. This approach enables scalable, reliable, and near real-time log ingestion into Splunk.

Ingestion Architecture

The implemented ingestion architecture is illustrated in Figure 4.10. It relies on AWS-native services to ensure efficient and decoupled log processing.

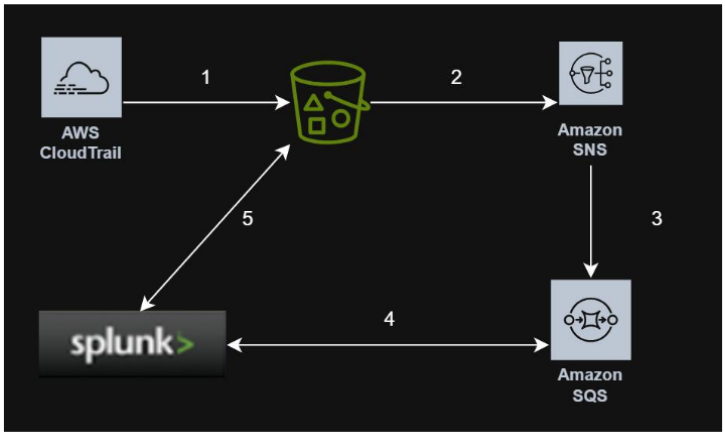

Figure 4.10: Splunk Ingestion Architecture

The ingestion workflow is composed of the following steps:

1. **Log Delivery to Storage:** AWS CloudTrail (management and data events) delivers logs to a centralized Amazon S3 bucket located in the security account.
2. **Event Notification via SNS:** Each time a new log file is written to the S3 bucket, a PUT event triggers a notification to an Amazon SNS topic.
3. **Message Queuing with SQS:** The SNS topic forwards notifications to an Amazon SQS queue. Each message contains metadata describing the newly created log file.
4. **Log Retrieval by Splunk:** A Splunk Heavy Forwarder continuously polls the SQS queue using long polling. Upon receiving a message, it retrieves the corresponding log file from S3.
5. **Processing and Forwarding:** The log file is parsed and forwarded to the Splunk indexing layer for analysis.
6. **Message Acknowledgment:** After successful processing, the message is deleted from the SQS queue to prevent duplicate ingestion.

Design Rationale

The decision to adopt an SNS/SQS-based ingestion pipeline was driven by practical observations made during implementation.

First, this architecture enables **near real-time ingestion**, as log processing is triggered immediately after log delivery. This significantly reduces detection latency compared to polling-based approaches.

Second, it introduces a strong **decoupling mechanism** between log producers (S3) and consumers (Splunk). The use of SNS and SQS ensures that components operate independently, allowing greater flexibility and easier maintenance.

Third, the SQS queue acts as a **buffering layer**, ensuring resilience. In case of temporary failures or downtime of the Splunk platform, messages remain in the queue and are processed later, preventing data loss.

Additionally, the architecture benefits from **automatic scalability**, as SQS can handle large volumes of messages without manual intervention.

Finally, the use of **long polling** improves cost efficiency and performance by reducing unnecessary API calls and optimizing resource usage.

Advantages of the Implemented Solution

The implemented ingestion pipeline provides several key benefits:

- **Near Real-Time Processing:** Logs are processed immediately after creation.

- **Scalability:** The system efficiently handles increasing volumes of log data.
- **Reliability:** Messages are retained until successfully processed.
- **Decoupling:** Independent operation of system components.
- **Cost Optimization:** Reduced API calls through long polling.

Queue Specifications

To ensure security, reliability, and performance, specific configurations were applied to the SQS queue. These configurations are summarized in Table 4.2.

Table 4.2: SQS Queue Specifications

Property	**Specification**
Confidentiality at rest	Server-side encryption enabled using AES-256
Confidentiality in transit	All communications secured using HTTPS
Integrity and Authentication	Access controlled via IAM policies; only authorized services can interact with the queue
High Availability	Messages replicated across multiple infrastructure components and data centers
Scalability	Automatically handles increasing workloads without manual intervention
Durability	Messages redundantly stored to prevent data loss, even in case of hardware failures

Conclusion

The implementation of an event-driven ingestion pipeline based on Amazon SNS and SQS significantly enhances the efficiency, scalability, and reliability of log integration into Splunk. This approach addresses the limitations of traditional ingestion mechanisms and provides a robust foundation for centralized security monitoring within the proposed architecture.

4.2.9 Syslog Integration for Virtual Machines

While cloud-native logging services enable centralized collection of infrastructure and API-level events, system-level logs generated within virtual machines cannot be aggregated at the organization level by default. To address this limitation, a standardized syslog-based logging mechanism was implemented.

Design Rationale The objective of this approach is to ensure consistent and centralized collection of operating system logs across all virtual machines. Since these logs are not captured by services such as CloudTrail or VPC Flow Logs, an additional logging layer is required.

To guarantee secure and efficient log transmission, the solution relies on a private network connection between the virtual infrastructure and the AMACP network, where the centralized logging platform (Splunk) is deployed. This design avoids exposure to the public internet, thereby reducing attack surfaces associated with threats such as Man-in-the-Middle (MITM) and Distributed Denial of Service (DDoS). Furthermore, it ensures low latency and high throughput for log delivery.

Configuration Guidelines A configuration guideline was defined to be systematically applied during the creation of virtual machine images. The following steps describe the required setup:

- Install the `rsyslog` service on the virtual machine
- Edit the configuration file `/etc/rsyslog.conf`
- Add the following forwarding rule:

```
. @@logs.gsoc.global.amadeus.net:PORT
```

Configuration Details The forwarding rule is defined as follows:

- `.`: captures all log messages
  - The first wildcard () includes all facilities (e.g., authentication, cron)
  - The second wildcard () includes all severity levels (e.g., info, warning, error)
- `@@`: specifies the use of TCP for reliable log transmission (a single `@` would indicate UDP)
- `logs.gsoc.global.amadeus.net`: DNS name of the centralized syslog server, accessible only within the internal network
- `PORT`: destination port, selected within the following ranges:
  - TCP: 6140–6180
  - UDP: 5140–5180

Deployment Considerations After updating the configuration, the rsyslog service must be restarted to apply the changes. This configuration is intended to be embedded in virtual machine templates to ensure consistency and scalability across deployments.

Summary This syslog-based integration complements cloud-native logging mechanisms by providing visibility into system-level events. It enables centralized monitoring, improves security observability, and ensures compliance with logging requirements across all virtual machines.

Chapter 5

Experimental Results

This chapter presents the experimental evaluation of the proposed log management and ingestion architecture. The objective of this analysis is to validate both the correctness and the efficiency of the implemented solution in a real-world environment.

The evaluation is structured around two main aspects. First, a series of functional tests are conducted to verify that the storage layer enforces the expected security policies. These tests ensure that access to log data is properly restricted, that only authorized AWS services can write logs, and that data retrieval is limited to approved entities.

Second, a quantitative analysis is performed to assess the behavior of the system in terms of log volume, storage consumption, and associated costs. This includes the comparison between different logging configurations, such as read-write and write-only trails, in order to identify the most cost-effective strategy while maintaining sufficient visibility for security monitoring.

Additionally, the experiments evaluate the system's ability to handle logs across multiple AWS accounts and regions, ensuring that the centralized storage solution scales correctly within an organizational setup.

The results presented in this chapter provide insights into the trade-offs between security, scalability, and cost, and serve as a basis for validating the design choices made during the implementation phase.

5.1 Tests on Log Storage

This section presents the validation of the centralized log storage configuration. The goal of these tests is to ensure that the implemented bucket policies correctly enforce access control rules, preventing unauthorized interactions while allowing only trusted AWS services and designated IAM roles to operate on the stored logs.

The evaluation focuses on three fundamental operations:

- Object upload (`s3:PutObject`)
- Object retrieval (`s3:GetObject`)
- Bucket listing (`s3:ListBucket`)

For each operation, expected behaviors derived from the defined policies are compared with the observed results in the deployed environment.

5.1.1 Object Retrieval Control

The first set of tests verifies whether access to stored log data is restricted to authorized entities. According to the implemented policies, only specific roles—namely the Prisma Cloud integration role and the Splunk ingestion role—are allowed to retrieve objects from the storage buckets.

Test	**Expected Results**	**Results Obtained**
GetObject	Allow access only to Prisma Cloud and Splunk roles Deny all other IAM entities	Prisma Cloud: allowed Splunk: allowed All other IAM entities: denied

Table 5.1: Validation of object retrieval policies

The results confirm that the confidentiality of stored logs is preserved by strictly limiting read access to authorized components.

5.1.2 Object Upload Control

The second set of tests evaluates whether only designated AWS services are allowed to upload logs into the storage buckets. Different policies were enforced depending on the type of logs being collected.

- **Flow logs**: only the `delivery.logs.amazonaws.com` service is allowed
- **Management events**: only AWS CloudTrail is allowed
- **Data events**: currently denied for all entities

Test	Expected Results	Results Obtained
PutObject (Flow Logs)	Allow delivery.logs.amazonaws.com Deny all others	delivery.logs.amazonaws.com: allowed All other IAM entities: denied
PutObject (Management Events)	Allow CloudTrail Deny all others	CloudTrail: allowed All other IAM entities: denied
PutObject (Data Events)	Deny all entities	All IAM entities: denied

Table 5.2: Validation of object upload policies

These results demonstrate that the ingestion pipeline is protected against unauthorized data injection, ensuring that only trusted AWS services can deliver logs into the storage.

5.1.3 Bucket Listing Control

The final set of tests verifies whether the visibility of stored objects is restricted to entities within the organization. The implemented policy enforces that only IAM principals belonging to the organization are allowed to list bucket contents, while all external entities are explicitly denied.

Test	Expected Results	Results Obtained
ListBucket	Allow IAM principals within the organization Deny all external entities	IAM entities in the organization: allowed External entities: denied

Table 5.3: Validation of bucket listing policies

The results confirm that access to metadata about stored logs is limited to trusted accounts, preventing unauthorized discovery of stored resources.

5.1.4 Discussion

Overall, the conducted tests validate that the storage configuration correctly enforces the intended security controls. The use of explicit deny rules ensures that unauthorized access attempts are systematically rejected, even in the presence of permissive IAM configurations.

Furthermore, the use of organization-based conditions strengthens isolation by preventing access from entities outside the organizational boundary. This

guarantees that the centralized log storage complies with the principle of least privilege while maintaining secure and controlled access for authorized services.

5.2 Tests on Flow Logs

This section evaluates the behavior of the log ingestion architecture when handling VPC Flow Logs across multiple AWS accounts and regions. The objective is to verify that the centralized storage solution correctly aggregates logs while preserving a consistent and organized structure.

5.2.1 Cross-Region and Cross-Account Collection

A key requirement of the proposed architecture is the ability to collect logs from different AWS accounts and regions into a single centralized storage location. To validate this capability, an experimental setup was deployed using two AWS accounts belonging to the same organization.

The storage solution was deployed in the dedicated security account. A second account, part of the same organization, was used to generate flow logs. Within this account, VPC Flow Logs were enabled in two distinct regions: `eu-west-1` and `us-east-1`.

To generate network activity, virtual machines were instantiated in both regions and basic operations such as system updates (e.g., `apt update`) were executed. This ensured the production of network traffic and the generation of flow log entries.

The experiment aimed to verify the following aspects:

- Correct segregation of logs by region
- Proper organization of logs by account
- Integrity of the stored log files
- Availability of metadata such as checksums and signatures

The results obtained from the production environment are summarized in Table 5.4.

Test	Expected Results	Results Obtained
Cross-region	A folder per region	A folder per region
Cross-accounts	A folder per account	A folder per account
Checksum	Checksum generated for each file	Checksum generated for each file
Digital signature	A digital signature for each file	Not provided by AWS

Table 5.4: Validation of flow logs ingestion across regions and accounts

5.2.2 Discussion

The results confirm that the centralized storage architecture correctly handles multi-region and multi-account log ingestion. Logs are automatically organized into a hierarchical structure that separates data by account and region, facilitating efficient querying and analysis.

The presence of checksums for each log file ensures data integrity during storage and transfer. However, the absence of digital signatures indicates that authenticity verification must rely on AWS-managed mechanisms rather than cryptographic signing at the log file level.

Overall, the system demonstrates strong scalability and reliability, successfully aggregating distributed log sources into a unified storage solution without loss of structure or integrity.

5.3 Management API Calls Logs Analysis

Two days after activating the organizational CloudTrail, several account owners reported an increase in their billing. This observation motivated a detailed analysis of management audit logs in order to evaluate their impact on storage consumption and operational costs.

To conduct this analysis, we initially enabled a CloudTrail trail capturing both read and write events. After a short observation period, the configuration was modified to a write-only trail to assess the potential cost reduction.

5.3.1 Methodology

AWS CloudTrail charges $2 per 100,000 management events delivered beyond the first free copy. Since logs are aggregated across all accounts in the organization, the total cost scales rapidly with the number of recorded API calls.

Given the skewed distribution of log events, statistical indicators were used instead of simple averages:

- First quartile (Q1): lower 25% of data
- Median (Q2): lower 50% of data
- Third quartile (Q3): lower 75% of data
- Interquartile Range (IQR): spread of the middle 50%

5.3.2 Cost Estimation with Read-Write Trail

Observation period: 3.19 days (May 12 to May 15)

Metric	Result
Total log files	1,030,524
Total log events	51,319,941
Log per file (Q1, Median, Q3, IQR, Max)	(7, 20, 38, 31, 32500)
Total storage (GB)	7.44447
Total cost ($)	1026.4

Table 5.5: Findings with read-write trail (3.19 days)

The results indicate a very high logging activity. While the median number of events per file remains moderate (20), the maximum value reaches 32,500, highlighting a highly skewed distribution with significant outliers.

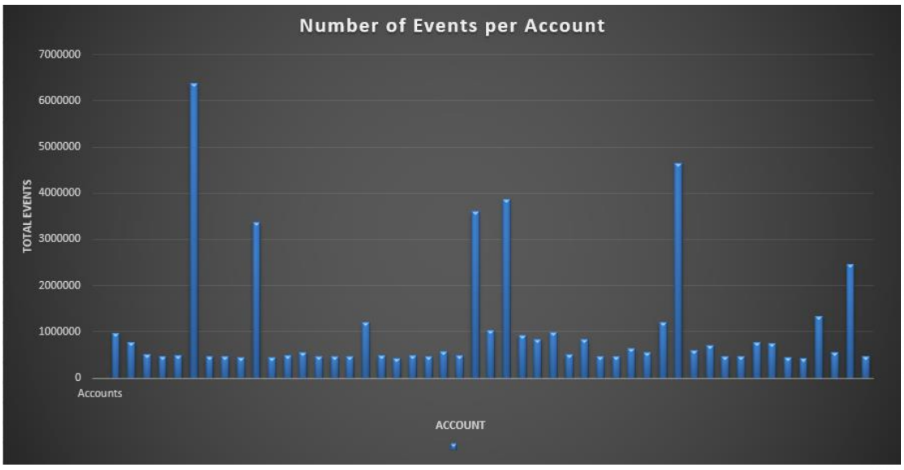

Figure 5.1: Number of events per account

Figure 5.1 shows that a small subset of accounts generates a disproportionately high number of events, while the majority of accounts produce relatively low volumes.

This imbalance explains the skewness observed in the statistical distribution and indicates that logging activity is concentrated in high-usage accounts.

Network Transfer Costs

- Log storage region: eu-west-1
- Prisma Cloud (AWS): $0.02 per GB
- Splunk (external): $0.09 per GB
- Prisma Cloud: $7.44447 \times 0.02 = 0.14$
- Splunk: $7.44447 \times 0.09 = 0.67$

Monthly Estimation

Metric	Projected Value
Total log files	9,686,926
Total log events	482,407,446
Total storage (GB)	70
Total cost ($)	9,649

Table 5.6: Monthly estimation for read-write trail

The projection shows that full read-write logging leads to substantial storage requirements and high operational costs, making it difficult to sustain at scale.

5.3.3 Cost Estimation with Write-Only Trail

Observation period: 3.94 days (May 12 to May 16)

Metric	Result
Total log files	44,522
Total log events	1,194,745
Log per file (Q1, Median, Q3, IQR, Max)	(2, 5, 16, 14, 2041)
Total storage (GB)	0.258031
Total cost ($)	23.9

Table 5.7: Findings with write-only trail (3.94 days)

Compared to the read-write configuration, the number of events and storage requirements are drastically reduced. The median drops to 5 events per file, and

the maximum value is significantly lower, indicating a more controlled and less skewed distribution.

Network Transfer Costs

- Prisma Cloud: $0.258031 \times 0.02 = 0.00516$
- Splunk: $0.258031 \times 0.09 = 0.02322$

Monthly Estimation

Metric	Projected Value
Total log files	342,820
Total log events	9,199,537
Total storage (GB)	1.98
Total cost ($)	184

Table 5.8: Monthly estimation for write-only trail

5.3.4 Discussion and Final Considerations

The comparison between the two configurations highlights a significant trade-off between observability and cost.

The read-write configuration provides full visibility into both access and modification events, which is essential for advanced auditing, anomaly detection, and forensic analysis. However, this level of visibility comes at a very high cost and generates a large volume of data.

In contrast, the write-only configuration significantly reduces both storage requirements and operational costs while still capturing all modifications to resources. This makes it a more efficient solution for large-scale environments where cost optimization is a priority.

To ensure an effective logging strategy, organizations must balance these two aspects. While read events provide valuable insights into access patterns, write events remain the most critical for tracking changes and ensuring accountability.

Additionally, to avoid redundant data collection and unnecessary costs, it is recommended to disable local trails in individual accounts when using an organizational CloudTrail. This prevents duplication of logs and contributes to a more efficient centralized logging architecture.

5.4 Validation Summary

This section consolidates the key findings obtained during the experimental evaluation of the proposed logging architecture. The objective of this evaluation was to assess its feasibility, scalability, security, and alignment with operational and compliance requirements within a multi-account cloud environment.

5.4.1 Functional Validation

The results confirm that the architecture successfully enables centralized log collection across multiple AWS accounts and regions. Logs from different sources, including flow logs, management audit logs, and system-level logs were consistently captured and stored in the designated centralized storage.

Integration with external platforms was validated through near real-time ingestion into both Splunk (SIEM) and Prisma Cloud (CNAPP), demonstrating end-to-end interoperability. In addition, security mechanisms such as encryption, fine-grained access control, and integrity validation were effectively enforced and tested. Overall, the system exhibited consistent and reliable behavior across all evaluated scenarios.

5.4.2 Scalability and Replicability

The use of infrastructure-as-code through CloudFormation enabled automated and reproducible deployment of the architecture. This ensures that the solution can be consistently replicated across multiple AWS organizations with minimal manual intervention.

Initial Deployment — 49-Account Organization

The architecture was first deployed across an organization comprising 49 AWS accounts. Using the CloudFormation StackSet mechanism, the security stack was propagated automatically to all accounts from a single template. The full organization-wide deployment completed in approximately 4 minutes 20 seconds, with an average per-account provisioning time of 53 seconds. Stack drift detection across all 49 accounts ran in under 90 seconds, and no manual remediation was required after initial rollout.

Incremental Onboarding — 10 New Accounts

To validate replicability, 10 new accounts were subsequently added to the organization. Thanks to the CloudFormation StackSet auto-deployment trigger, each

new account was automatically brought into compliance without any manual intervention. All 10 accounts were onboarded simultaneously in parallel, reaching full alignment in approximately 53 seconds regardless of the number of accounts added, confirming that the architecture's onboarding time is independent of batch size rather than introducing compounding overhead.

Summary

Table 5.9 summarizes the key deployment metrics observed across both scenarios.

Table 5.9: Scalability validation summary

Metric	Initial deployment (49 accounts)	+10 accounts
Total deployment time	∼4 min 20 s	∼53 s
Deployment model	Sequential + parallel	Fully parallel
Manual intervention required	None	None
Deployment method	StackSet (org-level)	Auto-triggered StackSet
Drift detection	<90 s (49 accounts)	<20 s (10 accounts)
Regions covered	eu-west-1, us-east-1	eu-west-1, us-east-1
Time scaling	—	Constant $O(1)$

Conclusion

These results confirm that the proposed architecture supports fully parallel onboarding, with all newly added accounts reaching compliance simultaneously rather than sequentially. The consistent deployment time of approximately 53 seconds observed for the 10-account batch, regardless of batch size demonstrates that the CloudFormation StackSet mechanism introduces no compounding overhead as the organization grows. This constant-time onboarding behaviour makes the solution well-suited for large and dynamic enterprise environments where multiple accounts may be added at once.

5.4.3 Performance and Cost Analysis

A detailed analysis was conducted to evaluate both the performance characteristics and the financial impact of different logging strategies. Two configurations were compared: a full read-write trail and a write-only trail.

Ingestion Latency

Log delivery from AWS services to the centralized S3 bucket occurs within seconds of generation. CloudTrail management events are typically delivered within 5 to 15 minutes of the recorded API call, in line with AWS service guarantees. VPC

Flow Logs, configured with a 10-minute aggregation window, are delivered to S3 within approximately 10 to 20 minutes of the capture window closing. Once stored in S3, the SQS-based notification mechanism triggers near-real-time forwarding to both Splunk and Prisma Cloud, with an observed end-to-end ingestion latency of under 3 minutes from S3 delivery to availability in the SIEM.

Throughput and Behaviour Under Load

During the read-write trail observation period (3.19 days), the pipeline processed a total of 51,319,941 events, corresponding to an average throughput of approximately 186,000 events per hour. The distribution was highly skewed: while the median file contained 20 events, a small subset of high-activity accounts generated files with up to 32,500 events. Despite these peaks, no ingestion errors or SQS queue backlogs were observed, demonstrating that the architecture absorbs burst traffic without degradation. Under the write-only configuration, average throughput dropped to approximately 12,600 events per hour, well within comfortable operating margins and confirming the pipeline's ability to handle a wide dynamic range of load.

Cost Analysis

The results highlight a trade-off between visibility and cost. While the read-write configuration provides comprehensive observability, capturing both resource modifications and access patterns, it generates significantly higher storage and processing costs. Conversely, the write-only configuration reduces costs substantially but limits visibility into read operations, which are critical for detecting unauthorized access.

To address this trade-off, the architecture adopts a centralized organizational trail while recommending the deactivation of redundant local trails. In addition, a phased rollout strategy targeting high-volume accounts was defined to optimize cost control and scalability.

Summary

Table 5.10 consolidates the key performance and cost metrics observed across both trail configurations.

These results demonstrate that the pipeline maintains stable performance across a wide range of load conditions, with no degradation even during high-volume burst events. The write-only configuration offers a 52× reduction in event volume and a 98% cost reduction compared to the read-write trail, while preserving end-to-end ingestion latency. The architecture is therefore well-suited for production environments where both performance and cost efficiency are critical constraints.

Table 5.10: Performance and cost comparison: read-write vs. write-only trail

Metric	Read-write trail	Write-only trail
Observation period	3.19 days	3.94 days
Total events processed	51,319,941	1,194,745
Avg. throughput	∼186,000 ev/h	∼12,600 ev/h
Median events per file	20	5
Peak events per file	32,500	2,041
End-to-end ingestion latency	<3 min	<3 min
Ingestion errors / SQS backlogs	None observed	None observed
Total storage	7.44 GB	0.26 GB
Estimated monthly cost	∼$9,649	∼$184

5.4.4 Security and Compliance Readiness

The proposed architecture aligns with key regulatory and organizational requirements, including PCI DSS and GDPR, by ensuring centralized log management, encryption, and fine-grained access control. These mechanisms provide full auditability and traceability of activities across the cloud environment.

Beyond technical compliance, this work contributes to the continuous improvement of the Information Security Management System (ISMS) under ISO 27001. In particular, the centralized logging framework supports the integration of newly onboarded entities into the existing security governance model, ensuring that logging, monitoring, and audit capabilities remain consistent across the organization.

By standardizing log collection and enforcing uniform security controls, the architecture helps maintain the effectiveness of the ISMS as the organizational scope evolves. This is especially critical in dynamic environments where new accounts, regions, or business units are regularly integrated.

Furthermore, mechanisms ensuring log integrity, such as digital signatures for flow logs were incorporated into the design, strengthening trust in the collected data and supporting compliance verification processes.

5.4.5 Stakeholder Validation

The evaluation was conducted in collaboration with internal stakeholders, including the Splunk and Prisma Cloud teams. Their feedback confirmed the correctness of the integrations and the operational usability of the solution.

Additionally, input from account owners, particularly regarding cost implications was incorporated into the final design. This collaborative approach ensured alignment between technical implementation and organizational constraints.

5.4.6 Conclusion

The experimental evaluation demonstrates that the proposed logging architecture is a robust and scalable solution for centralized log management in multi-account cloud environments. It integrates effectively with existing security platforms, enforces strong security controls, and supports compliance requirements.

Although cost remains a key consideration, the analysis provides clear guidance on balancing visibility and efficiency through appropriate logging strategies. Overall, the results validate the effectiveness of the proposed approach and support its adoption in production environments.

This work also provides a foundation for future improvements in automated log governance and cost-aware security monitoring in large-scale cloud infrastructures.

Chapter 6

Limitations and Future Work

This chapter discusses the limitations identified during the design and evaluation of the proposed logging architecture and presents potential enhancements to address these constraints. While the solution enables centralized, scalable, and secure log collection across multiple AWS accounts and regions, several challenges remain. In particular, ensuring the integrity of VPC Flow log data at rest emerges as a key limitation requiring further architectural improvements.

6.1 Limitations

One of the primary limitations concerns the integrity of vpc flow log data at rest. While AWS ensures reliable delivery and storage durability, it does not provide native guarantees for detecting post-storage modifications of log files. This creates a potential risk in scenarios requiring strong auditability and non-repudiation.

Another limitation lies in the strong dependency on AWS-native services. The architecture leverages CloudTrail, VPC Flow Logs, S3, and IAM, resulting in a tightly coupled design. While this enables deep integration within AWS, it limits portability and complicates adoption in multi-cloud environments without substantial redesign.

Cost management also represents a critical constraint. The analysis demonstrated that collecting both read and write events provides comprehensive visibility but significantly increases storage and ingestion costs. Even with centralized logging, the volume of generated data can rapidly scale, requiring careful trade-offs between observability and financial impact.

Additionally, the architecture relies on consistent configuration across multiple accounts within an organization. Misconfigurations such as redundant local trails

or incorrect access policies can lead to duplicated logs, increased costs, or gaps in monitoring coverage.

6.2 Proposed Enhancement: Log Integrity Architecture

To address the lack of integrity guarantees for log files at rest, an enhancement based on digital signatures was designed using a serverless architecture.

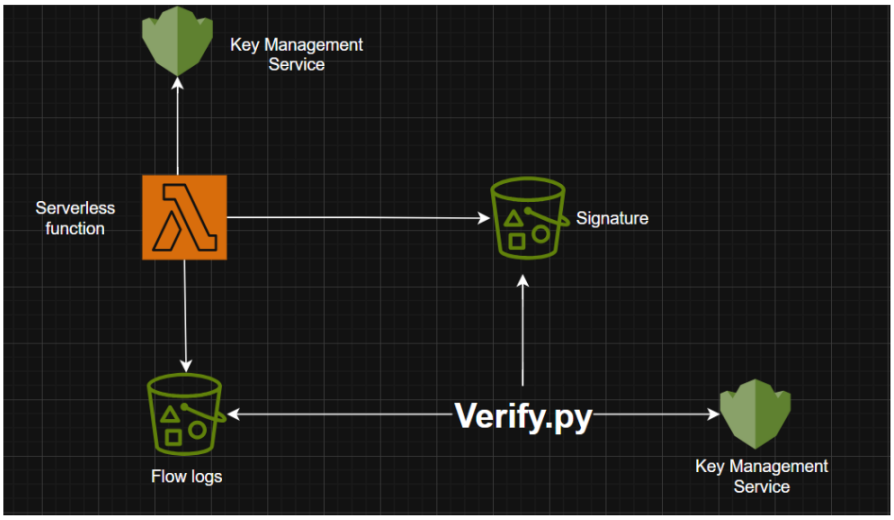

Figure 6.1: Flow Logs DSA Architecture

The proposed solution relies on a serverless function triggered upon the delivery of flow log files to the storage layer. This trigger is initiated by API calls from `delivery.logs.amazonaws.com`. Upon activation, the function retrieves a private cryptographic key from a key management service and computes a SHA-256 hash of the log file. This hash is then signed to produce a digital signature.

The generated signature is stored in a dedicated storage location, using the same filename as the original log file with an additional `.sig` extension. This ensures a clear and consistent mapping between each log file and its corresponding signature.

To verify integrity, a dedicated verification process retrieves both the log file and its associated signature. The log file is rehashed using the same SHA-256 algorithm, and the resulting hash is validated against the stored signature via the key management service. This process enables the detection of any unauthorized modification of the log files, thereby ensuring authenticity, integrity, and non-repudiation.

While this architecture effectively addresses the identified limitation, it introduces additional operational overhead. In particular, generating a digital signature for each log file increases computational load and storage consumption. At large scale, this overhead may become significant and impact cost efficiency. For this reason, the solution was designed but not deployed in production.

6.3 Future Work

Several directions can be explored to improve the proposed architecture and address its limitations.

A first area of improvement concerns optimizing the integrity mechanism. Instead of signing each individual log file, more efficient approaches such as batch-based signing or Merkle tree–based verification could be investigated. These techniques would reduce storage and computational overhead while preserving strong integrity guarantees.

Extending the architecture to a multi-cloud context represents another important direction. Supporting platforms such as Google Cloud Platform (GCP) and Microsoft Azure would enable a unified logging strategy across heterogeneous environments. This would require abstracting cloud-specific services and standardizing log formats and ingestion pipelines.

Further optimization of cost management strategies is also essential. Techniques such as log filtering, adaptive sampling, and tiered storage policies could reduce storage and ingestion costs while maintaining sufficient visibility for security monitoring.

From a governance perspective, future work could focus on strengthening automation and compliance enforcement. Integrating policy-as-code and continuous compliance monitoring would ensure consistent configurations across accounts and support the onboarding of new entities, while maintaining the effectiveness of the Information Security Management System (ISMS) as part of a continual improvement process.

Finally, enhancing analytical capabilities through advanced techniques such as anomaly detection and machine learning could further increase the value of centralized logging. These approaches would enable more proactive threat detection and improve incident response capabilities.

Glossary

Amadeus Cloud Platform (AMACP) Amadeus platform on Azure.

Amazon Machine Images (AMIs) Image that provides the software required to set up and boot an Amazon EC2 instance.

Amazon Organization Service that helps centrally manage and govern environments by creating accounts, allocating resources, grouping accounts, and applying policies.

- **Management account**: Root account of the organization, mainly used for defining service control policies.
- **Member account**: Account within the organization distinct from the management account. In this project, the security account acts as a member account with delegated administrative permissions.

Amazon Organization Unit (OU) Logical grouping of accounts within an AWS Organization used for governance and policy application.

Amazon Web Services (AWS) Subsidiary of Amazon providing on-demand cloud computing services on a pay-as-you-go basis.

Amazon Resource Name (ARN) Unique identifier of an AWS resource.

Amazon Simple Notification Service (SNS) Fully managed messaging service for application-to-application communication.

Amazon Simple Queue Service (SQS) Fully managed message queuing service enabling decoupling of distributed systems.

Address Resolution Protocol (ARP) Protocol used to map an IP address to a physical MAC address within a network.

Application Programming Interface (API) Set of rules enabling software systems to communicate and exchange data.

AWS Systems Manager (SSM) Service used to centrally manage and operate infrastructure across AWS and hybrid environments.

AWS CloudTrail Service that enables auditing, governance, and compliance by recording API activity within AWS accounts.

Cloud Function Google Cloud serverless compute service allowing execution of code without infrastructure management.

Cloud Infrastructure Entitlement Management (CIEM) Security capability that centralizes permission management across cloud environments and enforces least privilege.

Command Line Interface (CLI) Unified tool for interacting with AWS services through command-line commands.

Cloud-Native Application Protection Platform (CNAPP) Integrated security platform combining CSPM, CWPP, and CIEM to secure cloud-native applications.

Cloud Security Posture Management (CSPM) Security solution that identifies misconfigurations and risks across cloud environments.

Cloud Workload Protection Platform (CWPP) Security solution providing runtime protection for workloads such as VMs, containers, and Kubernetes.

Delegated administrator Account authorized to perform administrative actions across an AWS Organization.

Distributed Denial of Service (DDoS) Attack that overwhelms a system using large-scale distributed traffic.

Dynamic Host Configuration Protocol (DHCP) Protocol that automatically assigns network configurations to devices.

Domain Name System (DNS) System translating domain names into IP addresses.

Elastic Compute Cloud (EC2) AWS service providing scalable virtual servers.

Elastic Container Service (ECS) AWS container orchestration service.

Elastic Kubernetes Service (EKS) AWS managed service for running Kubernetes clusters.

Identity and Access Management (IAM) Service for managing users, roles, and permissions in AWS.

IAM Role Identity that can be assumed by services or users to perform actions.

IAM Service-linked Role IAM role directly associated with a specific AWS service.

IAM User Identity representing a person or application within an AWS account.

Lambda AWS serverless compute service enabling code execution without infrastructure management.

Man-in-the-Middle (MITM) Attack where an adversary intercepts communication between two parties.

Network Address Translation (NAT) Mechanism allowing private resources to access external networks.

(Elastic) Network Interface (ENI) Virtual network interface enabling communication between AWS resources.

Nitro-based instances AWS instances built on Nitro system providing enhanced performance and security.

Parquet format Columnar storage format optimized for analytics and compression efficiency.

Principal vs Service Principal A principal represents an entity performing actions in AWS, while a service principal represents an AWS service acting on behalf of a user.

Private Subnet Subnet without direct access to the internet.

Public Subnet Subnet with a route to the internet.

Reserved subnet IP addresses AWS reserves specific IP addresses within each subnet for internal use.

Serverless Cloud model where infrastructure management is abstracted from the user, with automatic scaling and pay-as-you-go pricing.

Security Information and Event Management (SIEM) Platform that collects, aggregates, and analyzes logs and security events in real time.

S3 Bucket AWS object storage container.

S3 Intelligent-Tiering Storage class that automatically moves objects between tiers based on access patterns to optimize cost.

Transmission Control Protocol (TCP) Reliable transport layer protocol ensuring ordered data delivery.

User Datagram Protocol (UDP) Transport protocol providing fast but unreliable data transmission.

Virtual Private Cloud (VPC) Logically isolated virtual network in AWS.

VPC Endpoints Enable private connectivity between VPC resources and AWS services without using the public internet.

Bibliography

[1] S. Susnjara and J. Smalley, IBM (2025), *What is cloud computing?*. Available at: `https://www.ibm.com/topics/cloud-computing`

[2] D. Koeppen and C. Winckless, Gartner (2024), *Market Guide for Cloud-Native Application Protection Platforms*. Available at: `https://www.gartner.com`

[3] E. Perrault, Amadeus (2023), *Cloud Security Posture (CISO) Dashboard Introduction*.

[4] A. Shreve, Frost Sullivan (2024), *Palo Alto Networks Named CNAPP Company of the Year*. Available at: `https://www.paloaltonetworks.com`

[5] AWS Documentation, *Welcome to AWS Documentation*. Available at: `https://docs.aws.amazon.com`

[6] Microsoft (2025), *What is a CNAPP?*. Available at: `https://learn.microsoft.com`

[7] AWS Sustainability, *Amazon Sustainability*. Available at: `https://sustainability.aboutamazon.com`

[8] Accenture and AWS (2023), *How cloud drives economic and societal impact through micro, small, and medium-sized businesses*. Available at: `https://aws.amazon.com`

[9] Microsoft (2024), *Shared responsibility in the cloud*. Available at: `https://learn.microsoft.com`

[10] G. Lindemulder and M. Kosinski, IBM (2025), *What is a man-in-the-middle (MITM) attack?*. Available at: `https://www.ibm.com`

[11] MongoDB (2025), *Cloud Computing Stack Explained*. Available at: `https://www.mongodb.com`

[12] ADEME, *Base Empreinte*. Available at: `https://base-empreinte.ademe.fr`

[13] AWS, *Customer Carbon Footprint Tool*. Available at: `https://aws.amazon.com`