



Expertise
and insight
for the future

Rizwan Ahmad

Cloud Security and Governance

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

31 March 2021

PREFACE

In the name of Allah, the Gracious, the Merciful.

The basis for this research originally stemmed from my passion for cloud security and governance. As the world moves rapidly into the digital age and cloud environments, there is and will be a greater need to develop tools to secure and govern different environments. During this research, I felt the biggest challenge was governance of cloud environments. I found out some security risks as well and developed tools to remediate these risks through automation.

I would like to express my deep gratitude to my supervisor Ville Jääskeläinen for his detailed and constructive comments, feedback and for his important support throughout this work.

I would like to express my special thanks to my parents and wife for their help, support, love and prayers to complete my studies at this institution.

Finally, I am thankful to my friends for their help and everlasting support during my studies.

Helsinki, 31st March 2021.

Author Title	Rizwan Ahmad Cloud Security and Governance
Number of Pages Date	51 pages + 5 appendices 30 June 2021
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Ville Jääskeläinen, Principal Lecturer
<p>Cloud security is a diverse challenge and the main reason is the loss of control of the infrastructure provisioning and visibility of the underlying virtual network. These cloud computing properties make cloud security and governance more challenging.</p> <p>This thesis discusses many security and governance solutions and categorizes them either security or governance related issues. Existing processes that rely on manual operations are not efficient enough because the infrastructure is expandable in minutes (in relation to human resources). How security and audit processes can be deployed on such an agile infrastructure? How does security and governance can be monitored at scale? How does one not compromise agility and security at scale? How to provide security assurance to top management while following agile methodology.</p> <p>This thesis discusses security and governance importance, implementation, automation at scale, enforcing and testing. AWS multi-account environment strategy was implemented to achieve security at scale. Centralizing security and logging enable security and audit teams to manage and view all resources by a central dashboard. Automation tools were deployed on each provisioned account to send logs, audit trails, resource inventory details to central service accounts for central management.</p> <p>The results strongly indicate that security automation is a key component of cloud security and cloud security can be achieved at scale without compromising agility.</p>	
Keywords	security, governance, cloud, AWS, data security, automation

Contents

Preface

Abstract

List of Figures (Tables)

List of Abbreviations

1	Introduction	1
1.1	Research Methodology	2
1.2	Thesis Structure	4
2	Cloud Computing	5
2.1	Cloud Deployment Models	5
2.2	Cloud Service Models	7
2.3	Amazon Web Services	8
2.4	Obstacles Preventing Cloud Adoption	9
2.5	Cloud Performance and Cost Factor	10
3	Cloud Security and Governance	12
3.1	Cloud Security Concerns	12
3.2	Cloud Governance Concerns	14
4	Security and Governance Assessment	16
5	Architecting Solution	18
5.1	AWS Multi-account Architecture	18
5.2	Core and Project Accounts	19
5.3	Security Consideration for AWS Cloud	21
5.4	Cloud policies for Defence in Depth	22
5.4.1	Network Layer	22
5.4.2	Platform Layer	23
5.4.3	Application Layer	23
5.4.4	Data Layer	23
5.4.5	Incident Response	25
5.4.6	AWS Identity Access Management	25
5.4.7	Logging and Monitoring	25

6	Deploying the Solution	27
6.1	AWS Organization and Organization Unit	27
6.2	Service Control Policies	28
6.2.1	Security and Governance Policies	28
6.2.2	Core-OU Policies	29
6.2.3	Dev-OU Policies	29
6.2.4	Test-OU Policies	30
6.2.5	Prod-OU Policies	30
6.3	CloudFormation Automation tool	30
6.3.1	Network layer security automation	31
6.3.2	Platform Layer Security Automation	32
6.3.3	Application Layer Security Automation	35
6.3.4	Data Layer Security Automation	38
6.3.5	Incident Response Automation	39
6.3.6	IAM Automation	39
6.3.7	Logging and Monitoring Automation	40
7	Results and Analysis	42
8	Conclusion	51

References

Appendices

- Appendix 1: Network Baseline
- Appendix 2: Enforcing MFA Authentication
- Appendix 3: Enforcing Encryption
- Appendix 4: Governing Data Encryption
- Appendix 5: Incident Response

List of Figures

Figure 1. The research process used in this study	3
Figure 2. Cloud Computing Models [5]	6
Figure 3. Cloud Services Models	7
Figure 4. Cloud Security Domains	13
Figure 5. AWS shared responsibility model [20]	15
Figure 6. AWS account architecture	17
Figure 7. AWS Multi-account Structure [21]	19
Figure 8. Multi-account governance	20
Figure 9. Cloud Adoption Strategy	22
Figure 10. Data Classification	24
Figure 11. AWS data policy	24
Figure 12. AWS organization OU	27
Figure 13. AWS organization accounts	28
Figure 14. Resource Security Policy	29
Figure 15. Core-OU Policy	29
Figure 16. Dev-OU Policy	30
Figure 17. AWS CloudFormation Stackset	31
Figure 18. Network Architecture	32
Figure 19. Trend Micro Deep Security Software Architecture	33
Figure 20. Deep Security portal	33
Figure 21. Trend Micro Dashboard1	34
Figure 22. Trend Micro Dashboard2	34
Figure 23. Web Application Firewall Dashboard	36
Figure 24. Web Application Firewall Workflow	36
Figure 25. AWS Inspector Installation	37
Figure 26. Inspector dashboard	37
Figure 27. Inspector findings	38
Figure 28. Data-layer security	38
Figure 29. Incident response	39
Figure 29. IAM enforcement	40
Figure 30. Logging and monitoring	40
Figure 31. Monitoring aggregated view	41
Figure 32. Dev-OU results	42
Figure 33. Prod-OU results	43
Figure 34. Core-OU results	43
Figure 35. Core-OU OPS-Role results	44
Figure 36. Network security	44
Figure 37. Network flow log	45
Figure 38. Web Application Firewall results	46
Figure 39. AWS inspector results	46
Figure 40. Data security S3-results	47
Figure 41. Data security EBS-results	47
Figure 42. Incident response results	48
Figure 43. IAM results	48
Figure 44. IAM results	49
Figure 45. Log account results	49
Figure 46. Security account results	50

List of Abbreviations

AWS	Amazon Web Services
AZ	Availability Zone
CLI	Command Line Interface
DC	Data Center
EBS	Elastic Block Storage
IaaS	Infrastructure as a Service
IaC	Infrastructure as Code
IDS	Intrusion Detection Systems
IdP	Identity Provider
IPS	Intrusion Prevention Systems
MFA	Multi Factor Authentication
NIST	National Institute of Standards and Technology
OU	Organization Unit
OWASP	Open Web Application Security Project
PaaS	Platform as a Service
RDS	Relational Database Service
S3	Simple Storage Service
SaaS	Software as a Service
SCP	Service Control Policy
SIEM	Security Information and Event Management
SNS	Simple Notification Service
TTP	Trusted Third Party
VM	Virtual Machine
VPC	Virtual Private Cloud
WAF	Web Application Firewall

1 Introduction

One of the most difficult challenges identified with cloud computing revolves around the security and compliance issues related to it. This isn't unexpected because when moving an organization's key applications, their IT foundation or their corporate and client data to either an internal or external cloud supplier has risks. For an IT security official, these risks to the business are a significant and needs to be thought thoroughly when starting to use a cloud service provider.

CIA triad describes the security and its characteristic in a detailed manner. The acronym "CIA" stands for three crucial components: confidentiality, integrity and availability. Analysis of these components in a system shows how secure is that system.[1]

- Confidentiality: To protect information from unauthorized persons/systems, many security protocols uses this approach to protect data.
- Integrity: It assures data consistency and accuracy over its entire lifecycle.
- Availability: It assures that a resource is obtainable and ready for use on the request from authorized systems/users.

To achieve security goals proper governance should be in place. Cloud governance can increase predictability and reduce uncertainty of security operations, structure to optimize the allocation of resources, assurance of security policy compliance, accountability for safeguarding information, level of assurance that critical decisions are not based on faulty information and foundation for effective risk management.[2]

Security risks are increased when an organization has huge cloud infrastructure used by many discrete teams following agile methods. It is impossible to implement proper cloud security and governance by using the processes and tools established for on-prem environment.

This research was done for a case company. This company is providing different software solutions to their customers. According to the software, their cloud operations team needs to provision different infrastructures. Most of the provisioning and governing processes were manual and because of that it was big hurdle for their development team to be more productive and agile. The operations team is also responsible for managing

and securing other customer environments. Their operations team has policies and processes in place but because of improper policies and lack of automation it was not really effective.

This project explains and highlights the key security and governance issues encompassing to cloud adoption and gives detailed supportive knowledge into how they can be tended to. The objective is to find out security and governance activities and insurances that will help the organization to deploy security and governance in their cloud environment.

In particular, this research answers the following questions:

- Why current security governance processes and tools are not effective?
- How to achieve central management of AWS environment?
- What security processes can be automated?
- How to implement security at scale?

1.1 Research Methodology

To accomplish the objectives of this research, various strategies and procedures were used. An exploratory approach was used to solve the challenges and addressing objectives of this research. This approach was used to investigate the problems which were not clearly defined.[3]

This thesis is written based on research papers, literature review, feasibility review of existing security and governance tools, processes, design principles, customization of available tools, new tools introduced, security best practices and shared responsibility model. This research describes cloud security and governance challenges, customized solution, enforcement of security and privacy policies, solutions to be compliant with organization.

Figure 1 below illustrates the research process used in this study. First, existing policies were reviewed to understand the current level of security and governance. A number of interviews had been scheduled with security and operation team leads to understand the challenges they were facing. After the interviews, a clear view of security, governance

and AWS central management challenges were identified. New multi-account structure, policies, processes, automation scripts and service accounts were created according to AWS best practices. AWS native security and governance tools were configured to achieve security and governance automation and control and restrict special infrastructure provisioning without approval.

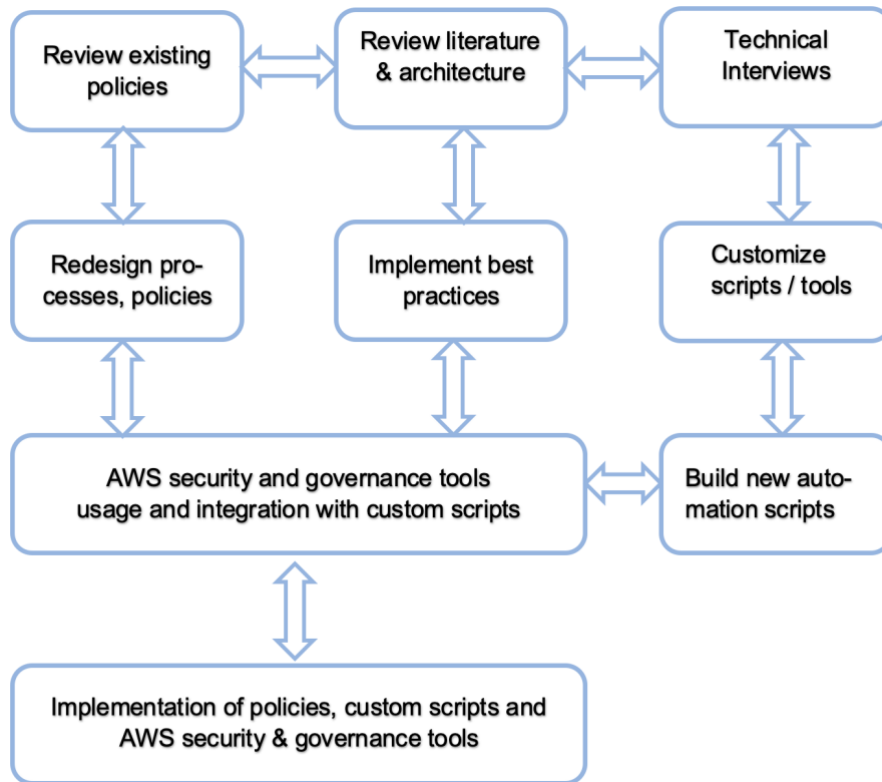


Figure 1. The research process used in this study

The main contributions of this thesis are:

- 1) Proposing categorization of cloud security and governance issues
- 2) Identifying the security and governance problems encountered by the cloud security team and how to automate remediation.
- 3) Implementing the best security and governance solution which fulfils the organization requirements without any security risks while teams are utilizing AWS services.

1.2 Thesis Structure

Section 1 describes the research problem, the objectives of the study, and research methodology. Section 2 describes an overview of cloud computing, cloud models and AWS cloud services. Section 3 describes and provides further insights of cloud security and governance issues. Section 4 presents current cloud environment setup and assessment of cloud security and governance. Section 5 describes the architected solution for the AWS cloud environment. Section 6 shows the implementation of the architected solution. Section 7 shows the results and last section concludes this research.

2 Cloud Computing

Cloud computing is providing different on-demand different services through internet especially compute and data services. These services can be used easily without any interaction with a cloud provider. A cloud customer can provision and release resources within minutes. [5]

National Institute of Standards and Technology (NIST) has defined five essential characteristics that comprise cloud computing.[4]

1. On-demand self-service: A consumer can provision cloud services without requiring service provider administrator to fulfil the request such as provisioning the server, data storage etc.
2. Network access: Cloud services hinge on internet infrastructure and accessed through standard mechanisms by using cloud providers API's or web.
3. Resource pooling: Cloud service provider resources are pooled and used by multiple customers using a multi-tenant model. These physical and virtual isolated resources are assigned to different customers according to their demand.
4. Rapid elasticity: Cloud service provider resources rapidly scale-in and scale-out according to consumer demands. In some cases, these are configured to scale in and scale out automatically.
5. Measured service: Cloud providers also provide the tool to automatically control and optimize resource use by processing metrics of cloud resources (e.g. compute, storage, bandwidth and active user accounts).

2.1 Cloud Deployment Models

There are four cloud deployment models as shown in the Figure 2. Each of these models has different attributes and implications for the end users.[5]

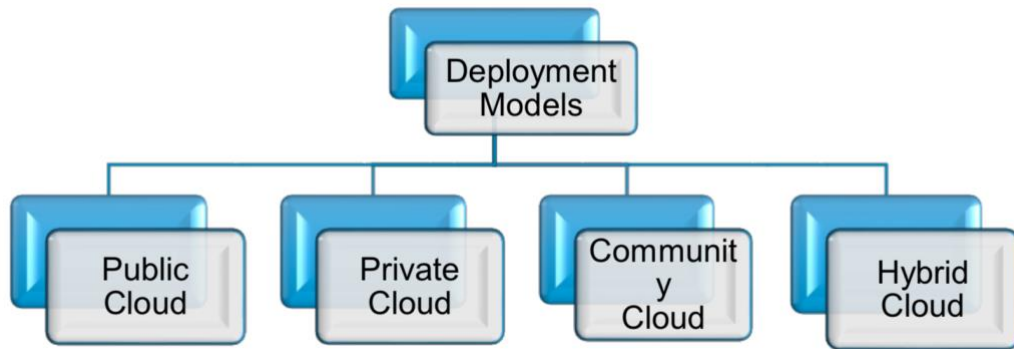


Figure 2. Cloud Computing Models [5]

Public Cloud

Public cloud is widely used cloud model and generally people have a picture in their mind for cloud computing. In public cloud, a cloud provider services such as compute, storage and other services are available all end users, companies and governments.

Private Cloud

Private cloud is a dedicated cloud environment to a specific customer and it is not shared with any other organization. It may be managed by the organization or a third party and may exist on-premise or off-premise. This model allows IT groups to be more responsive to their own company's needs by provisioning servers and applications much more quickly than the traditional method.

Hybrid Cloud

Hybrid cloud combines more than one cloud deployment models. Hybrid Clouds could be used, for example, to augment a private Cloud with public Cloud resources to handle severe workload fluctuations. One challenge is with the transmittal of data to the public Cloud in the case of excessive services demands.

Community Cloud

Community cloud shares cloud resources with a limited number of organization or employees. This model allows for customers to work together to leverage a common solution – an option that has appeal when a public cloud would not support the requirements of a few select customers.

2.2 Cloud Service Models

Cloud computing has three service models as shown in the figure below.

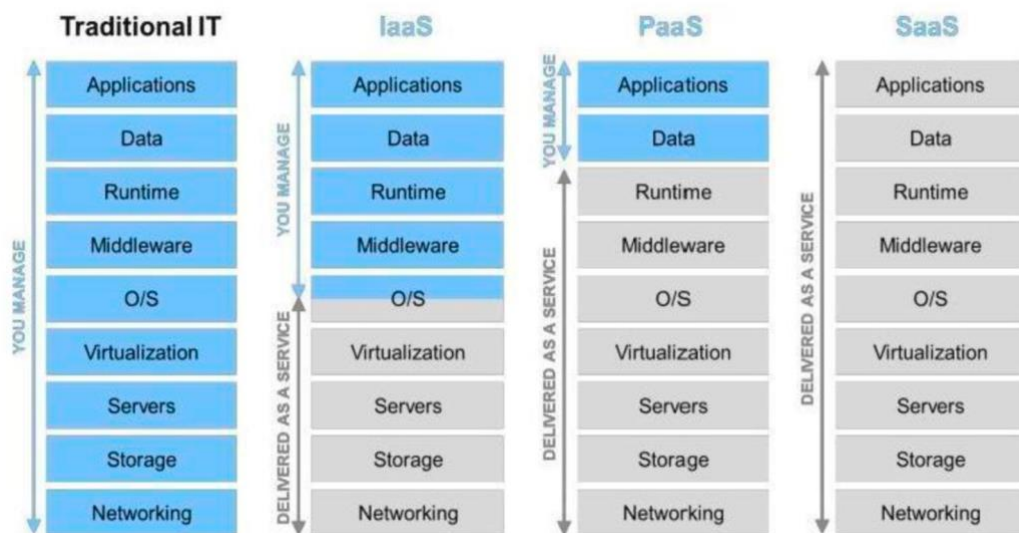


Figure 3. Cloud Services Models

Infrastructure as a Service (IaaS)

Infrastructure as a Service provides storage, compute, host provisioning, load balancing, public and private connectivity over the network. These common IaaS services provides a way for organizations to replace their data center hardware needs. Additionally, all of the infrastructure and dependencies for these services are also provided.[7]

Platform as a Service (PaaS)

PaaS provides a platform to run specific platform applications. Different PaaS providers support different technology stacks. In PaaS, it is service providers responsibility to manage operating system and middleware for customer.[7]

Software as a Service (SaaS)

In SaaS model, an application is delivered as a service to customers. The customer needs to configure the application according to their requirement and manage users. The service provider handles application servers, application updates, deployments and other things related to delivery of the service. Figure 3 shows that in SaaS model consumer does not need to manage anything.

2.3 Amazon Web Services

Amazon Web Services is an extensive, evolving and advanced cloud platform by Amazon. It includes a combination of PaaS, IaaS and SaaS offerings. The followings are five areas of security best practices in the cloud:

1) Identity Access Management

IAM is a key part of an information security program, ensuring that only authenticated, authorized and allowed users are able to access specified resources. AWS IAM service is primarily managing all privileges which allows one to control programmatic access, temporary access, IAM roles and users. [8]

2) Detective Controls

Detective controls are used to detect a security threat or incident. AWS native detective controls can be used to detect potential threat and also do action based on threat type such as auto-remediation, alarms etc. Detective tools process event, data and audit logs to detect threats. [9]

3) Infrastructure Protection

It is a key part of information security system. Infrastructure protection assure that all systems and services are protected against unauthorized and unauthenticated access and other potential vulnerabilities. Control methodologies such as security-in-depth are critical for successful implementation of infrastructure security. [9]

4) Data Protection

AWS provides multiple ways to protect data, encrypting data at rest and in transit. AWS services make sure that it is easier to encrypt critical data. [9]

5) Incident Response

AWS provides many tools to react quickly for different incidents automatically. Actions can be detective and preventive as well.

2.4 Obstacles Preventing Cloud Adoption

The following are the most common obstacles preventing cloud adoption:

1) Security and privacy challenges

The security and privacy challenges are serious concerns for most of the organizations. Cloud environments are multi-domain environments in which each domain security policies can be different. One organization can consider the security challenges as hurdle to move to the cloud even though there are many security and privacy solutions which can assure the security and privacy of their data.[1]

2) Internal Resistance

Cloud can reduce the number of administration tasks by automation and managed services by cloud provider. Subsequently, it can help employees to work more on the actual development. Therefore, it may result in a significant reduction in the size of the IT department staff [2]. Hence, the specialists in IT unit may see cloud adoption as a threat. Such individuals may be afraid of learning cloud technologies or losing their jobs.

3) Integration and interoperability

The industry lacks standards for different API's and cloud integration, interoperability standards, and associated other standards that allow interoperability of private to onprem cloud, public to private clouds, and so on [4].

4) Reliability and trust

Cloud is very reliable now a days, but cloud outages experienced in past from giant providers such as Google and Amazon were happened, and some organizations are concerned for that outage. This discouraged some organizations who were planning to migrate to the cloud. Cloud adoption is a trust based model where lack of trust inhibits its adoption. [11]

5) Governance, SLA and QoS

Governance of cloud environment is a challenge because cloud resources can be provisioned in minutes and if proper governance policies are not configured it could be impossible to find the ghost servers running in the environment. Cloud providers have their own SLA's which could be higher than one's expectation.

2.5 Cloud Performance and Cost Factor

The advantages of cloud computing and analysis of the performance and cost factors are described below.

1) Security

Cloud computing provides secure environments by implementing security standards, providing cloud native tools to manage security and privacy. Cloud providers are very strict regarding their security because of their trust model.[4]

2) Network bandwidth

Network load or network outage can affect the performance of the cloud services for such a time until the issue is resolved.

3) Service level agreement

A service level agreement (SLA) is a contractual agreement between a cloud provider and a consumer. It describes all the terms for using the services such as service requests, service quotas, penalty, fees, etc. [5]

4) Data recovery

In case of a disaster, data recovery from replica or the latest backup requires timeline and it can affect the performance of the system.[6]

5) Number of users

Concurrent users of the cloud services can affect the performance of the service if it exceeds the service capacity. [5]

6) Fault tolerance

Cloud environments architecture are usually fault tolerance which means that the design enables a system to continue its intended operation, possibly at a reduced level, rather than failing completely. [10]

7) Other factors

There are also other factors that can affect cloud performance such as latency, network bandwidth, redundancy and compute processing provisioning. [10]

3 Cloud Security and Governance

This chapter will explain briefly most concerns of cloud security and governance.

3.1 Cloud Security Concerns

Since the beginning of the cloud computing era in the past decade, the search for convincing evaluation of cloud-specific risks and security vulnerabilities has occupied IT analysts and continues today [12]. Cloud computing benefits have been vigorously marketed.

Concerns over security, privacy and governance are also the most commonly cited factors [14]. When one considers moving to the Cloud, the first thing most enterprises concern about is how their data will be protected, whether their applications will be protected from unauthorized access and how they can enforce compliance with key security regulations [1].

In particular, they recognized management and contractual issues as strikingly important for cloud computing and put considerable emphasis on them. In short, the most important risks they identified were [13]:

- Data protection
- Compliance risk
- Loss of governance
- Isolation failure
- Lock-in
- Management interface compromise
- Insecure data deletion
- Malicious insider

This list blends client-side and Cloud provider side risks, yet it is especially fascinating in light of the fact that three out of the eight dangers are not technical and generally and mostly based on contractual agreements and governance of information systems (i.e., loss of governance, lock-in, compliance) one is identified with the high repetition and dynamical migration of data in distributed computing (i.e., insecure or incomplete data deletion), the others are so for the most generic threats not so much explicit to cloud computing. It is likewise fascinating to take a gander at the list of less significant risks

referenced by ENISA, the European Union Agency for Cybersecurity. A considerable lot of them are not cloud-specific risks, like those from data in transit on the net or distributed denial of service (DDoS), but some are cleverly cloud-specific: Loss of business reputation due to cotenant activities sounds awkward somehow. Cloud security can be categorized in these five domains:

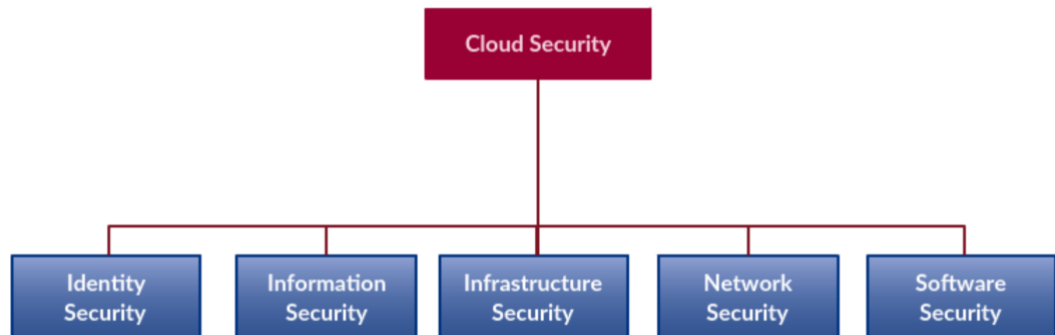


Figure 4. Cloud Security Domains

- 1) Information security: It is defined as a set of policies, tools, processes and controls to prevent, information system, data, document and other threats to information. [1]
- 2) Identity security: It is defined as an approach to secure the user or system that is used to authenticate and authorize the identities to access the secure resources. [1]
- 3) Infrastructure security: It is at the root of the entire organization security plan. It protects the critical business applications. To be able to check the business criteria that the underlying infrastructure is protected is completely necessary for an organization.
- 4) Software security: Although there is a wide variety of efforts in the development of software in terms of complexity and difficulty, they all need security assurances. Since there is no such principle as complete protection guaranteed, the goals are to build stable software with carefully built protection in it, not an afterthought add-on capability. [2]
- 5) Network security: Network security is any activity designed to protect the usability and integrity of your network and data. It is a core cloud computing requirement. It targets

a variety of threats. It involves taking preventive physical and software measures to protect the underlying networking from unauthorized access. [2]

3.2 Cloud Governance Concerns

Cloud governance requires policy-making and the implementation of an organizational structure with well-defined responsibilities for IT management, business processes, and applications, as these elements are moved out from the traditional IT environment to the cloud [2]. Governance is not a one-size-fits-all proposition, the structure and scale must take into account the complexity, the company culture, complexity and priorities of the enterprise. Moving IT governance into the cloud makes effective governance more difficult. Customers have to accept service provider control over a number of important issues and business process. Without cloud governance in place to provide risk navigation guidance and to procure and operate cloud services effectively, an organisation might face these common problems [16]:

- Stalled projects
- Incomplete risk assessments
- Misalignment with enterprise objectives
- Compliance or regulatory penalties or failures
- Frequent policy exception reviews
- Budget overruns

Cloud security and governance is mutual responsibility of the cloud provider and the cloud consumer as described in the following shared responsibility model.

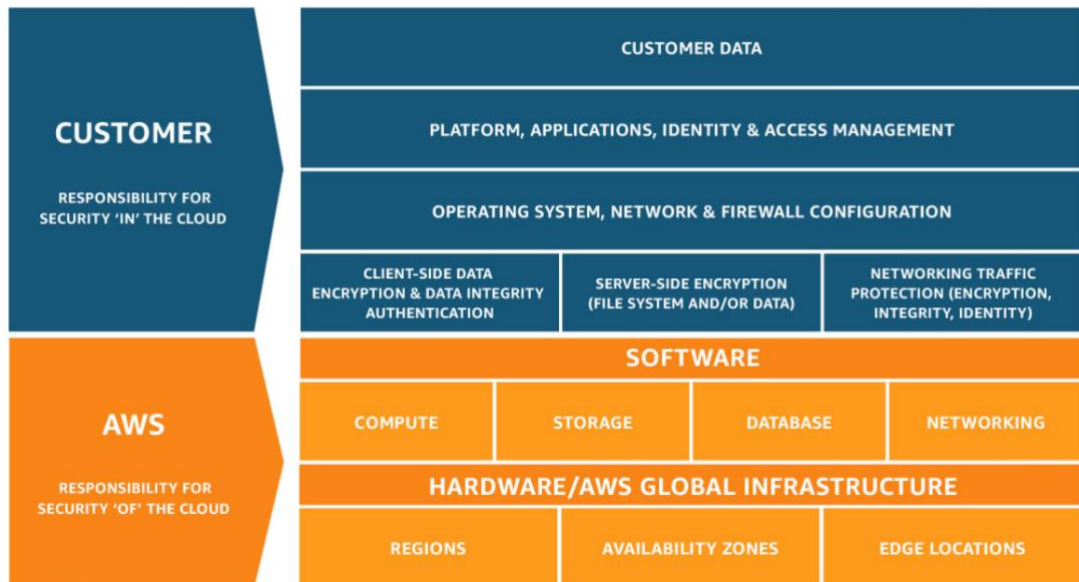


Figure 5. AWS shared responsibility model [20]

AWS responsibility is to protect the infrastructure of AWS which runs all the cloud services. Specifically, it includes all the hardware, networks, physical data centre locations, software and services that runs on AWS cloud. Customer responsibility depends on the usage of cloud services and customer is responsible for configuring the services in a secure way.

4 Security and Governance Assessment

This section describes the customer environment, architecture and setup before the implementation of this project.

The enterprise has old legacy applications and new applications projects. The projects are using different software stacks and different technologies. Some applications which are currently deployed on AWS are cloud native applications. Some applications are also deployed on Microsoft Azure, but this research focuses only on AWS cloud.

Individual and special projects can follow distinctive agile strategies and methodologies. Some applications are updated more often, and new versions are released to test newly added functionality which needs new testing environments according to the application. Others follow a progressively waterfall approach where their releases and updates are not that often. [14] The security team is responsible for security of all the applications and projects running in the company.

Currently, AWS accounts are provisioned based on application/project needs. Project manager is responsible for cost control and management of the account. The security team is ordering new AWS account, configure security checks and deliver account to project manager. There is no central logging, cross account access, central monitoring, central governance and security automation responses in place.

There security team was lacking the processes, policies and procedures to handle AWS accounts security and governance. Their security team members also mentioned that some resources are uselessly running in accounts which cause extra cost, time and security team resources to figure it out.

Each AWS account has different requirement as required by the application and compliance requirements. The security team in managing and handling all the requests according to demands. Since there is no automation, central management of accounts and deployments, it was very difficult for security team to manage and govern multiple AWS accounts as they have other cloud provider accounts and on prem as well. The following figure is showing the AWS account structure before this project implementation.

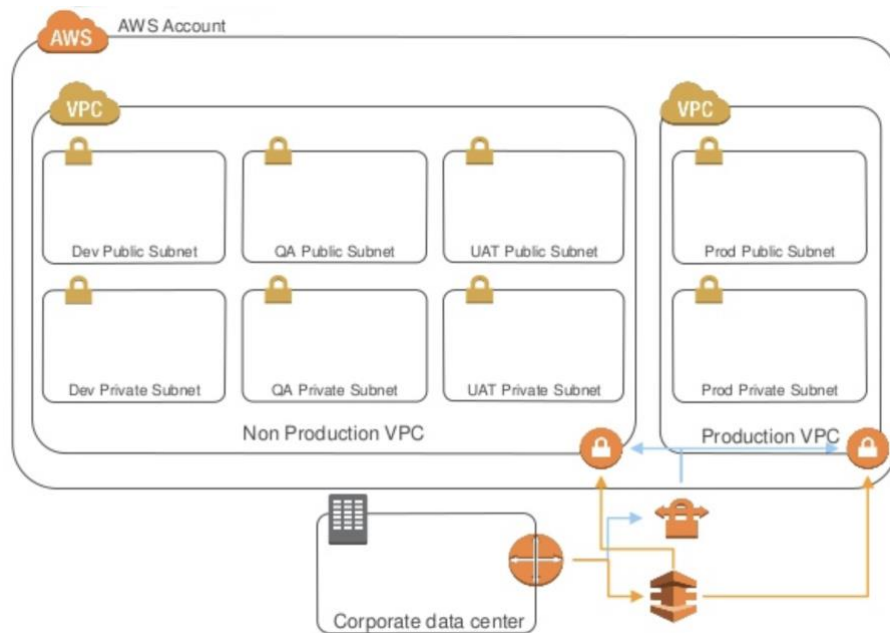


Figure 6. AWS account architecture

The figure above is describing the AWS account architecture. Some AWS accounts are connected to on-prem through VPN connection. Actually, these AWS accounts need to access on-prem servers for application dependencies but not all AWS accounts are connected with on-prem. As you can see, Same account is used for development and test environment which is not best practice as well.

5 Architecting Solution

This section describes the proposed architecture of AWS accounts, security, and governance controls. It is also explaining, how cloud applications are deployed in AWS accounts, and how applications are secured and governed for compliance.

5.1 AWS Multi-account Architecture

A well-defined AWS account structure lets one create groups of AWS accounts with central policy management and consolidated billing. It is very important to have a proper AWS account structure for security, monitoring, costs and governance aspect. Currently, AWS organization structure was not in use and all AWS accounts were provisioned individually.

A well-architected multiple account AWS environment is configured by leveraging AWS organization service that enables an enterprise to centrally manage and govern multiple accounts. The following are most used AWS organization features:

- Control access and permissions

AWS Organizations work with AWS Single Sign-On to enable access to accounts within your organization. By using Service Control Policies, access permissions can be controlled centrally across all organization accounts.

- Auditing, monitoring and compliance

By defining an organization-wide AWS CloudTrail trail to centrally log all actions performed across your environment and protect it from modification at the account level. AWS Organizations can use to centrally audit, monitor, and secure your AWS environment to ensure they are compliant with your corporate policies.

- Share resources across accounts

AWS services make it possible to centrally define critical resources and make them available to accounts across your organization.

- Centrally manage costs and billing

AWS Organizations consolidating billing feature can consolidate usage across all accounts in your organization into a single bill.

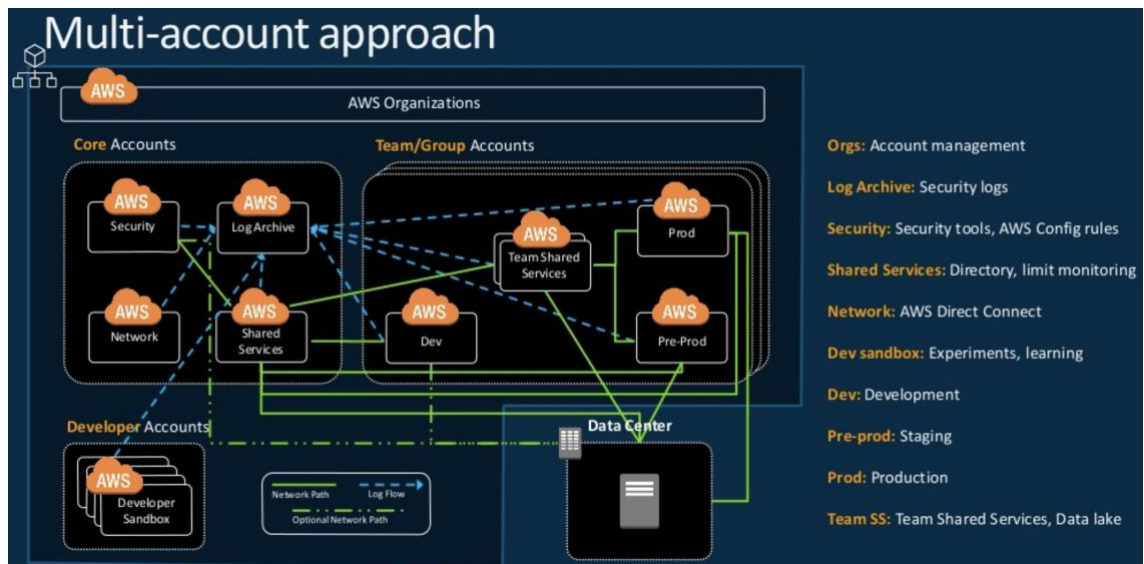


Figure 7. AWS Multi-account Structure [21]

Above figure 7 describes AWS multi account architecture, core accounts are used to centralize AWS environment. These accounts are used by other AWS accounts to send log data, security notifications, network management and share services.

5.2 Core and Project Accounts

As one can see in the following Figure 8, different Organization Units (OU) are created and different Service Control Policies (SCP) are configured according to these organizational units' policies. These policies are enforced on all accounts under these OU's to enforce governance and compliance regulations on all of your AWS accounts.

- Core Organization Unit

Core OU contains the security audit and logging archive accounts. All CloudTrail and Config logs are being saved in logging account. Security account is a restricted account for security individuals to get a central aggregated view of accounts configuration and programmatic read and write access to all accounts in the organization. These accounts often are referred to as shared accounts.

- Dev/Test Organization Unit

Dev/Test OU contains all projects dev and test accounts. Specific SCP policies are defined for this OU which will help to governed and limit user actions in the accounts as defined in the policies.

- Prod Organization Unit

Prod OU contains all projects production accounts. More strict access control and SCP policies are defined for this OU which will help to enforce governance and compliance regulations.

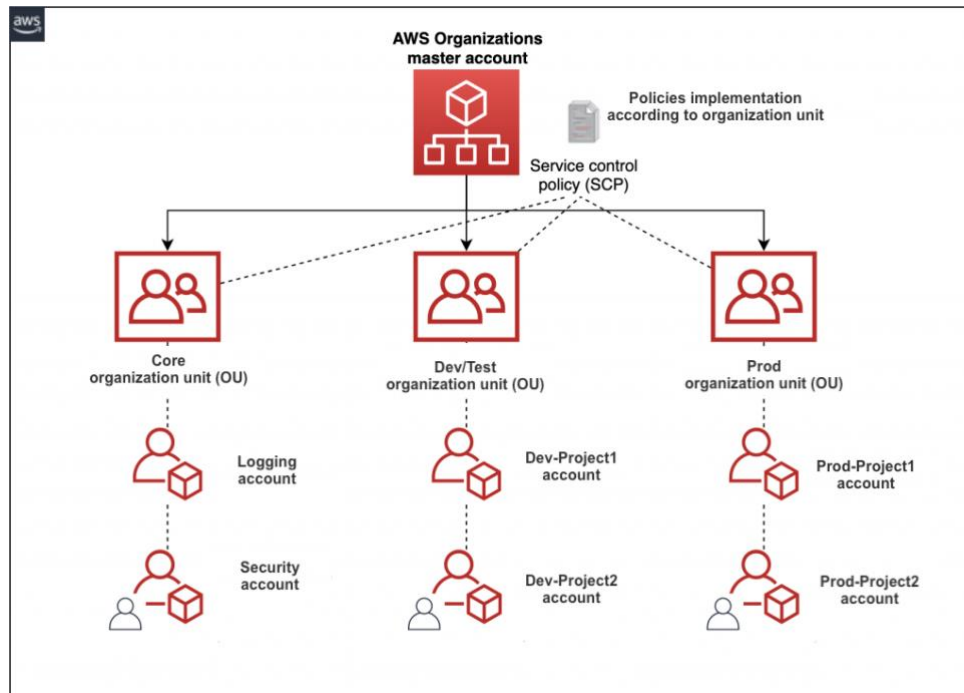


Figure 8. Multi-account governance

An administrative boundary is implemented by using multiple account structure, it is describing how permissive or restrictive my policies are based on the account type such as Dev, Test and Prod accounts. Separating user permission within an account can be complicated and error prone sometimes. Using separate accounts is often the best option. Workload boundary are also defined not to peer various workloads together within dev, test and prod accounts, ensuring that 'blast radius' for a poorly behaved application is minimized.

Segregating development and production environment in different virtual private networks (VPC's) but in the same AWS account has many challenges as explained earlier. Creating separate accounts for different environment simplifies configuration management, governance and granular control.

All projects development and production accounts were provisioned separately. Development accounts has less strict security policies and limitations of specific type of

instances usage to control cost. Data transfer between development and productions accounts were restricted.

5.3 Security Consideration for AWS Cloud

This section explains the security consideration for cloud adoption. Adopting the cloud brings with its new security issues. Organization data is stored on the cloud, in datacenters of cloud providers. Moving to the public cloud means that some of the security controls are not available in cloud provider environment. There might be other tools available for the same purpose but one must be aware of all security tools available for use.

AWS cloud adoption strategy was based on risk assessment and project delivery:

- New projects must consider to architect and deploy directly on the cloud.
- Risk assessment also considered for sensitive projects which has the sensitive data and data storage and compliance limitations such as public sector regulations. Projects that has such compliance requirement of accessing data from certain locations were not targeted to migrate to cloud.
- Internal development projects were moved to cloud to get benefits of cloud elasticity, services and fast provisioning of desired resources.
- Couple of on-prem less critical application were also selected to move to cloud because it was not required such significant changes for cloud adoption.
- Legacy applications were not moved to cloud because of application configurations and dependencies.

The following Figure 9 is illustrating the strategy for adoption of AWS cloud. With this clear cloud adoption strategy, new projects will adopt the cloud very easily and with minimum risks.

Projects on AWS
Upcoming projects without such privacy restrictions.
Development and test environments.
Developers individual sandbox environments
Projects with data for low latency used by CDN.
Projects with compute intensive workload.
Projects selected for migration to AWS
Less-mission critical projects.
Projects with significant variations over time.
Projects for rapid delivery requirements.
Small projects which can easily migrated to cloud.
Projects not migrated to AWS
Projects with secrets and sensitive data.
Projects with access limitation from specific location.
Projects with legacy application architecture.
Interconnected applications with other on-prem resources.

Figure 9. Cloud Adoption Strategy

5.4 Cloud policies for Defence in Depth

In security, Defence in Depth is a collective use of various security mechanisms to defend an organization from targeted attacks. Targeted attacks may describe as attacks against a system service's confidentiality, credibility, and availability. It is also called layered-based approach and each layer usually protects against different type of threat. Once these layers are paired together, they operate as shields to protect against several attacks. Let's have a deep understanding, how this approach has been implemented.

5.4.1 Network Layer

First layer is the network layer of the Defence in Depth approach to the cloud environment. This layer is the entry point for all traffic to the cloud environment, it is very important to configure this layer carefully. For the security of network layer:

- Access control list were implemented on network level.
- EC2 and RDS security groups inbound and outbound configured with limited access from company's network.
- DMZ and network segmentation implemented.

- PKI based authentication must be in use for accessing servers.

5.4.2 Platform Layer

This is the second layer of the Defence in Depth approach; it is actually the operating system layer which is exposed to the network. All the servers running some kind of services are listening on specific ports. Only specific ports are allowed to accept connections, server and application must be updated and patched for security updates.

- Host-based intrusion detection system is installed for application servers, Trend Micro Deep Security (IDS/IPS) is used from marketplace.
- OSSEC (HIDS) were installed also that automatically block the suspicious login attempts.

5.4.3 Application Layer

This layer is also considered very important because users directly interact with the applications through this layer. Now a days, web applications are open to world on ports 80 and 443 and anyone can access the application over the internet so there is more risk for attack.

- AWS WAF (Web Application Firewall) is the first defence to protect web applications and it was used to secure applications.
- AWS Inspector agent also installed on servers to automatically assesses applications for exposure, vulnerabilities, and deviations from best practices.
- There are also secure coding practice and static code analysis techniques in use.

5.4.4 Data Layer

This layer is used to protect data. Securing data is a critical piece of operating and building systems. To protect data, all transit-data and data at rest must be encrypted. It was organization's data classification policy that all data must be classified which are stored on AWS. Since the data classification policy was not changed, a data encryption policy is defined which describes the classification levels that must be encrypted during transit and at rest.

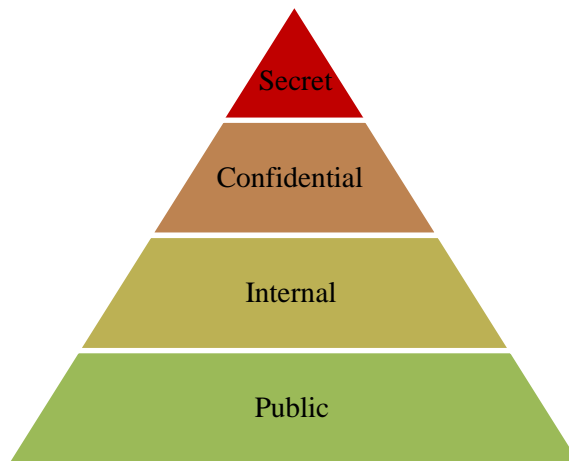


Figure 10. Data Classification

According to the policy, Public data can be unencrypted, but all other data classifications must be encrypted at rest and in-transit. The encryption policy also applied to other AWS services which are often used in projects. Organization data policy was easily implemented by leveraging AWS robust data encryption services to protect data throughout its lifecycle. The following Figure 11 illustrates the details:

EC2 EBS Volumes
Encrypted AMI must be used to provision new servers.
New volumes must be encrypted before attached to servers
KMS keys rotation must be automated.
S3 File Storage
Data which is classified as sensitive must be encrypted.
Encryption keys procedures must be implemented for S3 storage as well.
S3 server-side encryption also be used if there is no specific requirement.
RDS Database instances
RDS instances must be encrypted.
Encryption keys procedures must be implemented for RDS as well.
SSL Certificates
ACM must be used to create private and public certificates for domains.

Figure 11. AWS data policy

5.4.5 Incident Response

Incident response is the technique where all the monitoring-related activities happen. Incident response automation process improves reliability and increases the speed of the response. AWS CloudWatch alarms are configured to send alerts for:

- Root user login activity
- Unauthorized login attempts
- SSH rejected attempts
- Audit trail changes
- VPN state monitoring
- Billing alerts

5.4.6 AWS Identity Access Management

AWS IAM is a key part of an information security system. IAM allows to control access to AWS accounts and resources. AWS users, federated users, applications and other services permissions are managed through it by applying granular policies. These policies grant permissions to a user, role, group or AWS resource.

The following IAM user policies are implemented:

- IAM user accounts name must be similar to their AD emails.
- All users must get the permissions through groups and no permission policies should be attach directly to the user.
- All password policies (minimum length, rotation, complexity) are similar to corporate AD but multifactor authentication is enforced to get access to account resources.

5.4.7 Logging and Monitoring

Logging and monitoring are very important for governance and compliance. AWS offers many services for logging and monitoring such as AWS Config, AWS Cloud Watch, AWS Cloud Trail and AWS Security Hub.

The following AWS services are used for logging and monitoring:

- A dedicated account named logging account is created for central logging from all AWS accounts. AWS CloudTrail and AWS Config services are used to send logs and configuration data to central logging account.
- A dedicated account named Security account is created for central security and governance monitoring. AWS config service in this account is showing all

aggregated configuration of all accounts in a single dashboard and compliance status. SNS topics were also configured to get different configuration, compliance and security alerts to get notified security teams.

- AWS Security Hub is also configured to aggregates, prioritizes security alerts and findings from multiple AWS services such as Amazon Guard Duty, Amazon Inspector, IAM and Access Analyzer.

6 Deploying the Solution

This chapter describes the actual deployment of the solution which is architected in the previous chapter.

6.1 AWS Organization and Organization Unit

AWS organization was setup in the organization root account, two core accounts were created for logging and security. A new organization unit (OU) was created with the name Core and newly created core accounts are added in that organization unit (OU) to apply relevant service control policies (SCP). Service control policies for core accounts were applied on the organization unit level. Additional organization unit were also created for Dev, Test and production accounts. The following figure 12 show AWS organization unit structure.

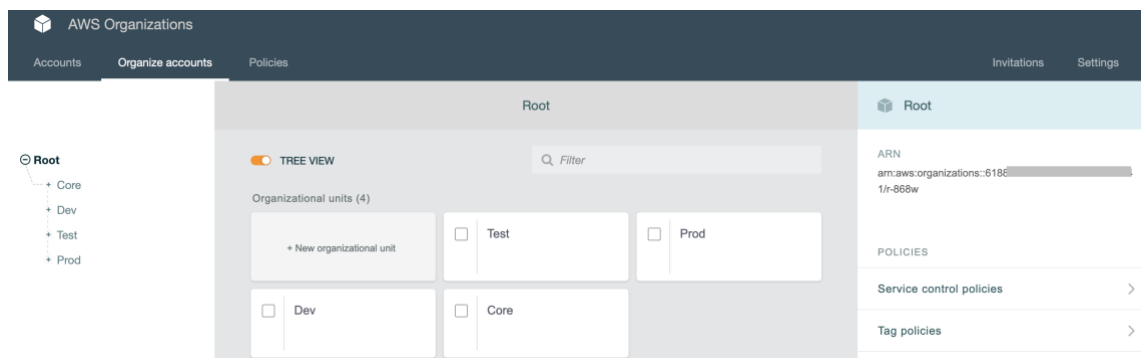


Figure 12. AWS organization OU


```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN": "arn:aws:iam::*:role/AwsOpsExecution"
        }
      },
      "Action": [
        "cloudtrail:DeleteTrail",
        "cloudtrail:PutEventSelectors",
        "cloudtrail:StopLogging",
        "cloudtrail:UpdateTrail"
      ],
      "Resource": [
        "arn:aws:cloudtrail::*:trail/aws-ops-*"
      ],
      "Effect": "Deny",
      "Sid": "GRCLOUDTRAILENABLED"
    }
  ],
}

```

Figure 14. Resource Security Policy

6.2.2 Core-OU Policies

Core organization unit policies were implemented at two accounts. Logging account which was used for central logging and security account which was used for security audits, monitoring and alerts. This policy protects central logging bucket from anyone's access to make changes in bucket attributes such as logging bucket encryption, access-logging, deletion of bucket and retention period. Only enterprise operations team can change the attributes through a specific role named AwsOpsExecution. A snippet from policy is shown below but the full policy can be found at appendix 2.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN": "arn:aws:iam::*:role/AWSOpsExecution"
        }
      },
      "Action": [
        "s3:PutEncryptionConfiguration"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Deny",
      "Sid": "SP-AUDITBUCKETENCRYPTIONENABLED"
    }
  ],
}

```

Figure 15. Core-OU Policy

6.2.3 Dev-OU Policies

All the Development and sandbox accounts for developers are provisioned under this organization unit. A special policy was created for this organization unit to limit the usage of specific instances type and limit the usage of only AWS free-tier services for cost control and also it limits the usage of only AWS Ireland region (eu-west-1).

A snippet from policy is shown below but the full policy can be found at appendix 3.

```

{
  "Version": "2012-10-17",
  "Statement": [ {
    "NotAction": [
      "iam:*",
      "organizations:*",
      "route53:*",
      "budgets:*",
      "waf:*",
      "cloudfront:*",
      "globalaccelerator:*",
      "importexport:*",
      "support:*",
      "health:*",
      "route53domains:*"],
    "Resource": "*",
    "Effect": "Deny",
    "Condition": {
      "StringNotEquals": {
        "aws:RequestedRegion": [
          "eu-west-1"]
        }
      }
    },
    {
      "Sid": "EC2limit",
      "Action": "ec2:RunInstances",
      "Resource": "*",
      "Effect": "Deny",
      "Condition": {
        "ForAnyValue:StringNotEquals": {
          "ec2:InstanceType": [
            "*.nano"
            "*.micro" ]
          }
        }
      }
    }
  ],
}

```

Figure 16. Dev-OU Policy

6.2.4 Test-OU Policies

Accounts under this organization unit were pre-prod/staging environment. Security and governance policies were only applied on these accounts. Cost control and other restrictions were applied individually on the project level.

6.2.5 Prod-OU Policies

Accounts under this organization unit were production accounts. Security and governance policies were only applied on these accounts. Disaster recover backup policies were implemented to securely transfer backups to the backup account and IAM users cannot change backup policies in Prod-OU accounts. More granular policies were applied on individual account level to do different security checks according to the project. The policy can be found in the Appendix 4.

AWS organization account structure and policies were in place. Now, Let's configure the security and governance automation of all the accounts.

6.3 CloudFormation Automation tool

Since the main cloud provider for this research is AWS, it was decided to use AWS native CloudFormation tool to create and manage AWS resources. Cloud Formation service allows to use Yaml/Json languages to model and provision resources in an automated manner. CloudFormation service was used to deploy resources across all regions and accounts.

CloudFormation StackSets extends the functionality of stacks that allows to update, delete and create stacks across different regions and accounts with a single api operation. Using the Master account, CloudFormation templates can be easily define, manage, and use the template for deploying resources into different target accounts across different regions and environments.

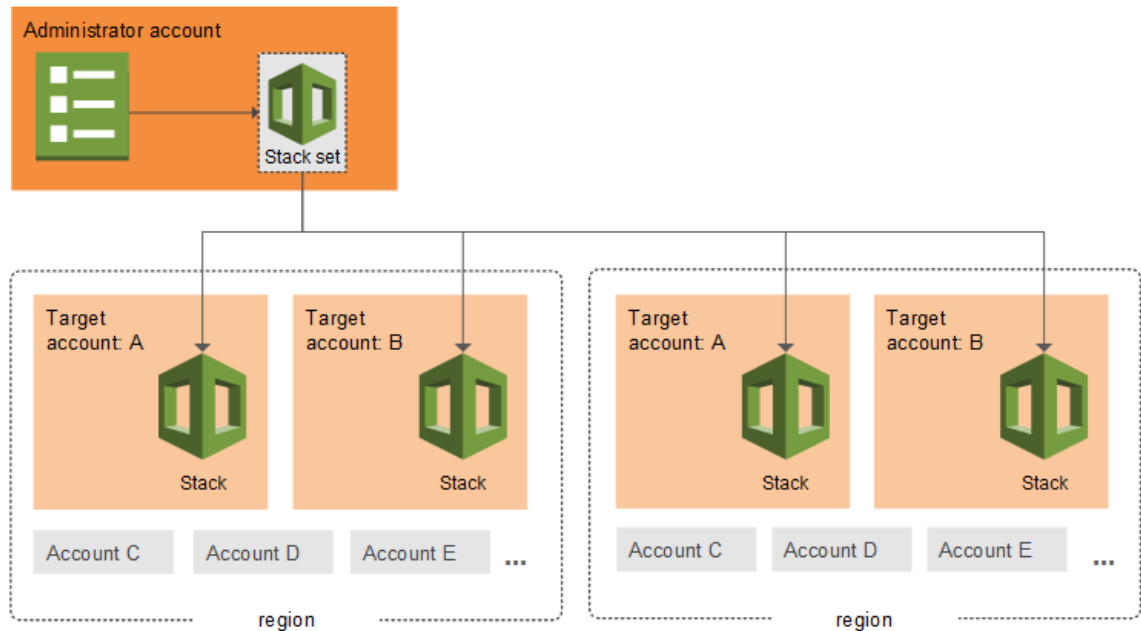


Figure 17. AWS CloudFormation Stackset

The following CloudFormation templates scripts were written to configure AWS accounts security and governance in a larger scale.

6.3.1 Network layer security automation

Network layer is the first layer of defence and this layer is the entry point for all traffic to the cloud. It was very important to configure this layer carefully. For the security of network layer, virtual private cloud (VPC), network access control lists (NACL), security group and network traffic flow logs were configured through CloudFormation. An automation was configured to send alerts if any mentioned network resource property changed. This network baseline template resource creation and mapping are shown below in the picture. Network security script can be found in Appendix. The following figure shows the network architecture.

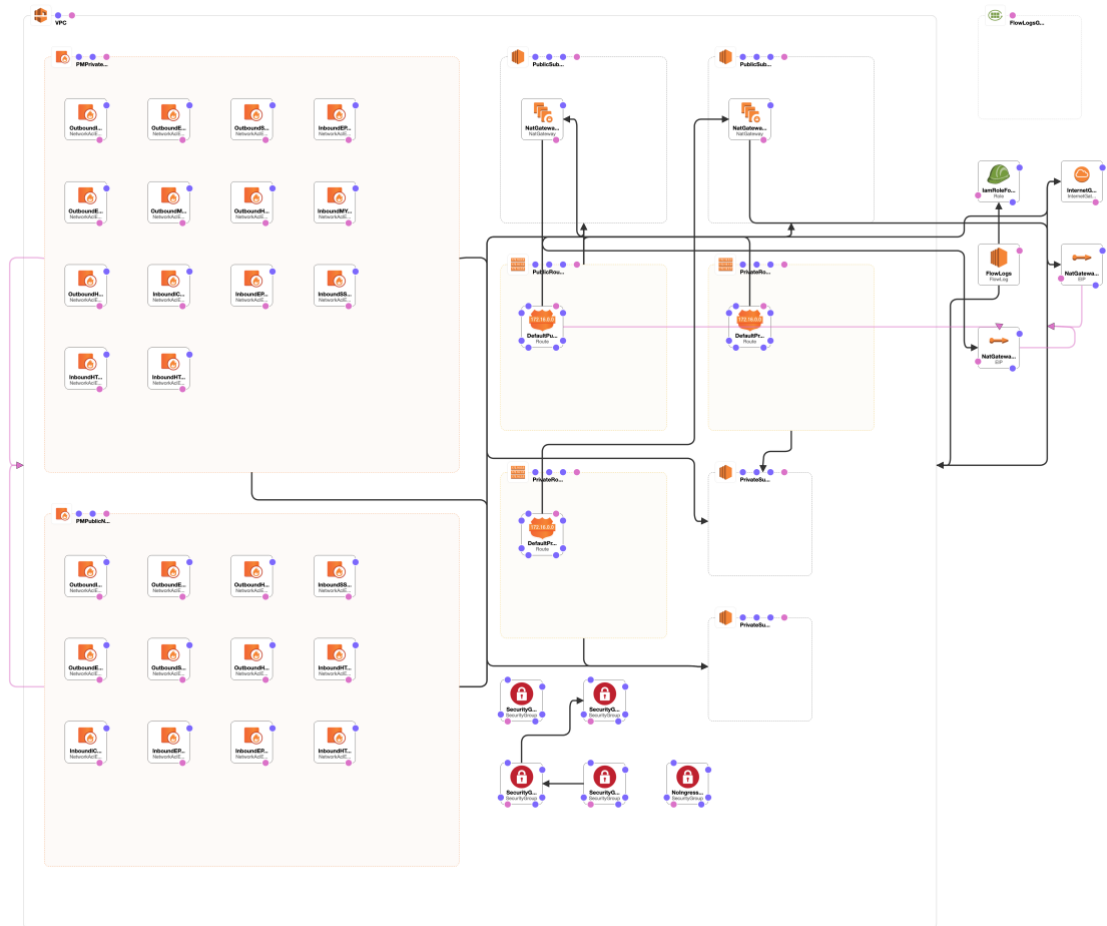


Figure 18. Network Architecture

The above figure 18 is describing the network architecture and connection between different VPC components like public subnets, private subnets, NACL for public and private subnets, IGW, NatGW, security groups and flow logs.

6.3.2 Platform Layer Security Automation

Trend Micro Deep Security is a comprehensive security platform for physical, virtual and cloud servers. It was installed on application servers to implement host level security. It is host-based security solution that provides intrusion prevention (IPS), anti-malware, host firewall, file integrity, WAF, log inspection and content filtering in different operating systems such as Windows, Linux, Solaris and Unix operating system.

Deep Security solution has created two EC2 compute instances, elastic load balancers and RDS (Postgre) database instance to run the solution on AWS. The following figure 19 shows the solution architecture.

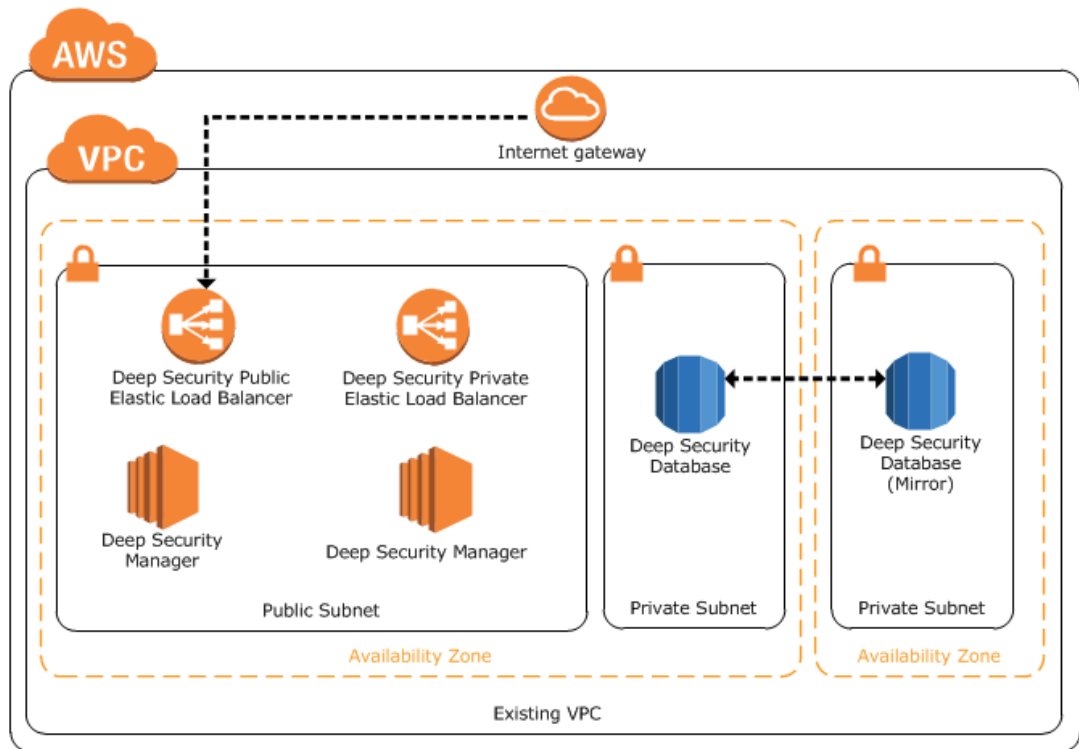


Figure 19. Trend Micro Deep Security Software Architecture

Trend Micro dashboard widgets and more widgets are available to monitor different events. This admin portal can be customized according to user preferences.

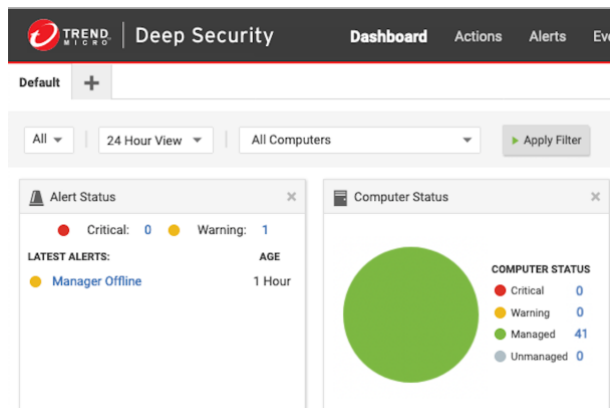


Figure 20. Deep Security portal

Deep security portal shows the status of all instances in the virtual private cloud and also trend micro deep security manager status.

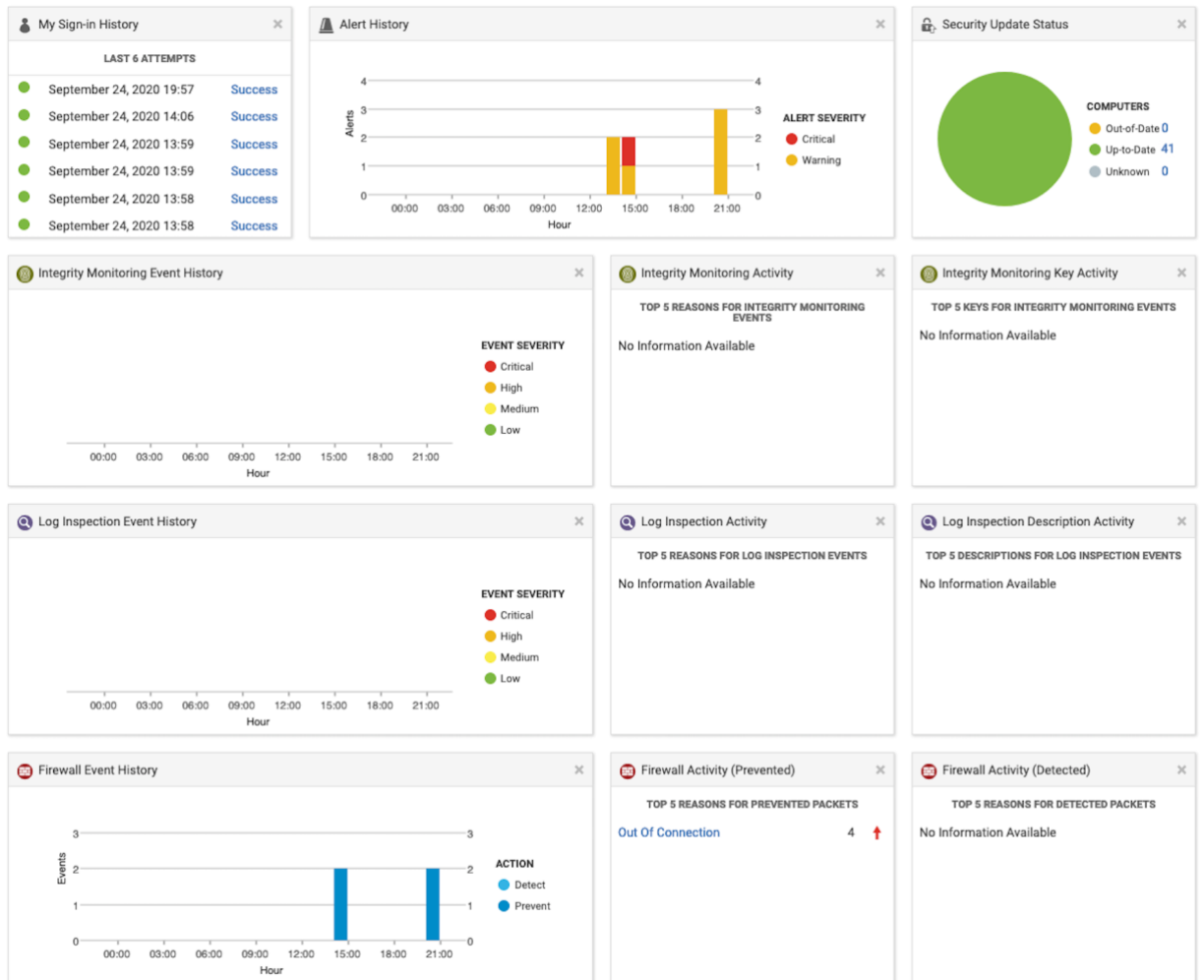


Figure 21. Trend Micro Dashboard1

Trend Micro deep security was configured to inspect logs, firewall events analysis, integrity monitoring and send alerts based on suspicious activity.

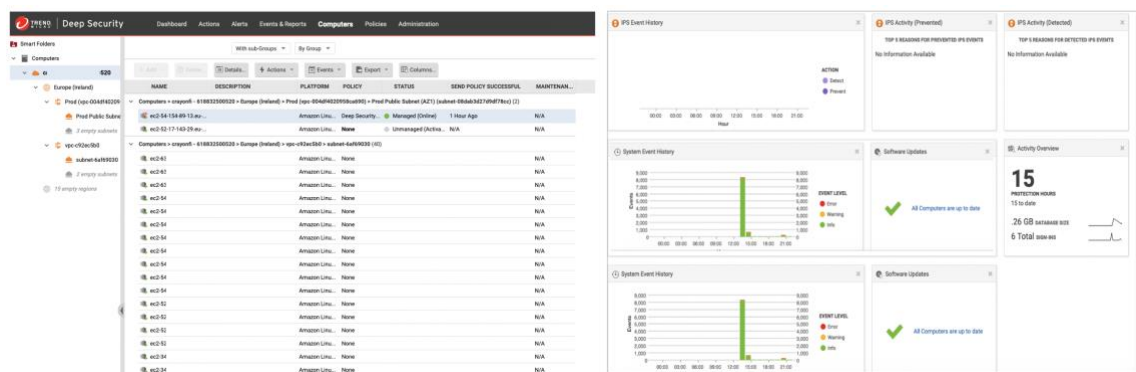


Figure 22. Trend Micro Dashboard2

6.3.3 Application Layer Security Automation

To secure the application layer, AWS WAF (Web Application Firewall) was configured for applications which were exposed to internet. AWS WAF can configure to control how an API Gateway, Application Load Balancer or CloudFront distribution responds to web requests. Open web application security (OWASP) top 10 security risks were considered and AWS WAF was configured to mitigate these risks mentioned below. CloudFormation WAF configuration code can be found in appendix.

- Mitigate SQL Injection Attacks
- Cross-Site Scripting (XSS)
- Sensitive Data Exposure
- Broken Authentication and Session Management
- Cross-Site Request Forgery (CSRF)
- Under protected APIs
- Broken Access Control
- Security Misconfiguration
- Insufficient Attack Protection
- Using Components with Known Vulnerabilities

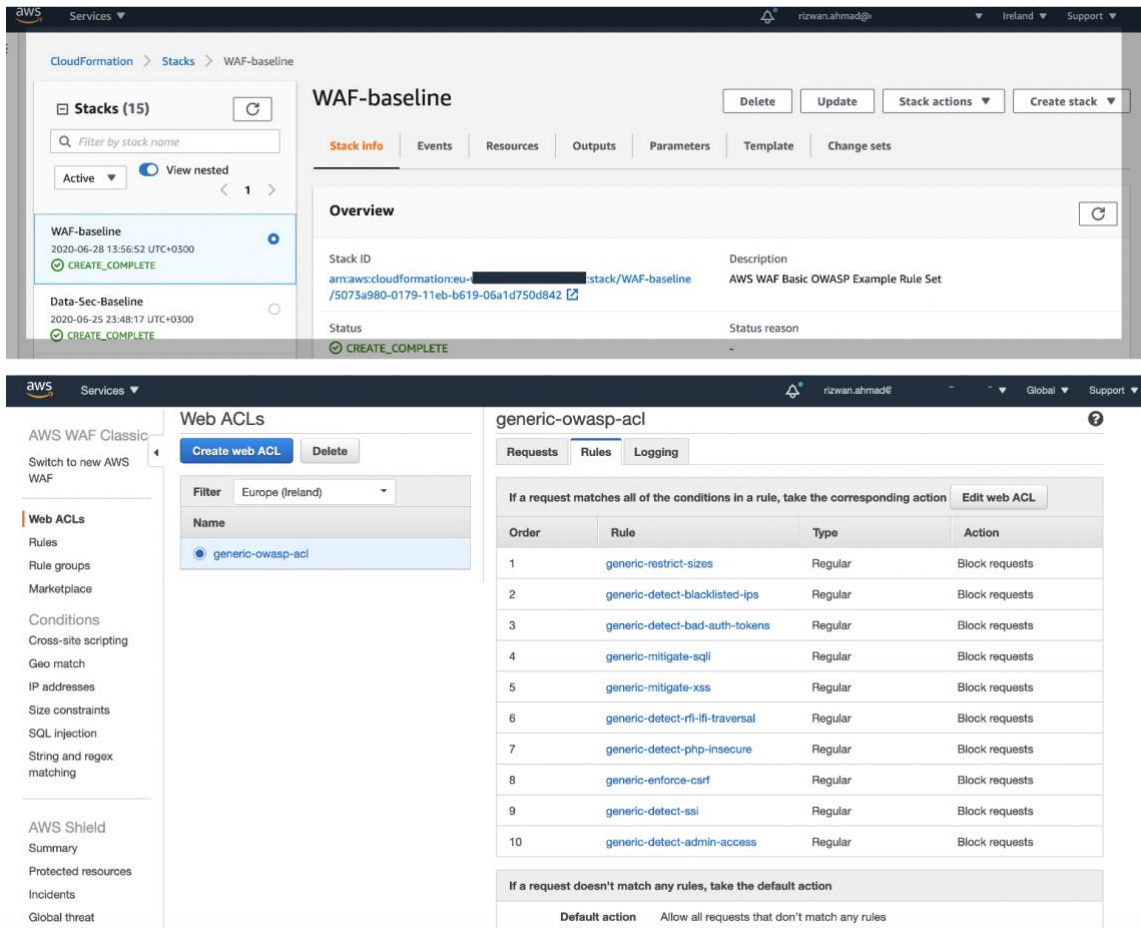


Figure 23. Web Application Firewall Dashboard

The following figure shows that how AWS WAF works, as one can see in the figure when a request was received from user it is evaluated through all rules defined in a WebACL in ascending order. If these rules allow the request than it was forward to the attached resources.

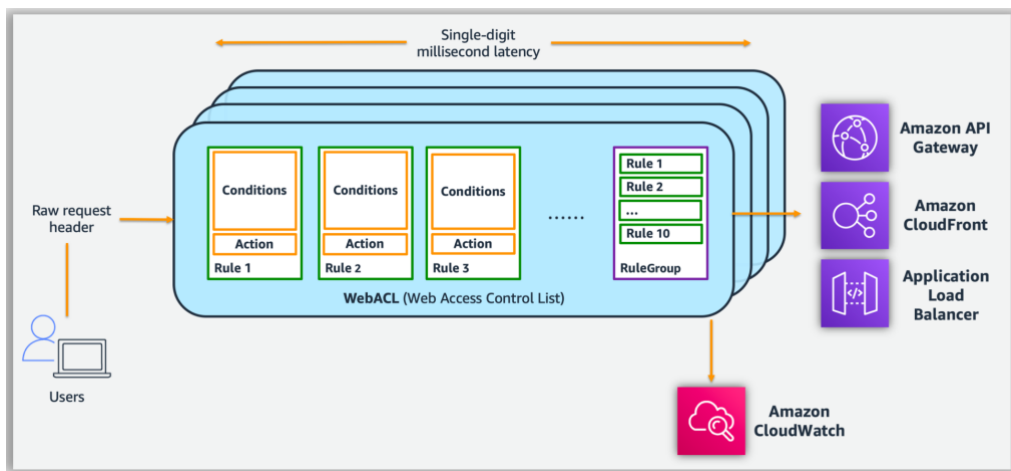


Figure 24. Web Application Firewall Workflow

AWS inspector was also configured to automatically assesses vulnerabilities, applications for exposure, and deviations from best practices. The following commands were used to download and install the agent.

```
[ec2-user@ip-10-10-10-79 ~]$ wget https://inspector-agent.amazonaws.com/linux/latest/install
--2020-09-25 13:49:17-- https://inspector-agent.amazonaws.com/linux/latest/install
Resolving inspector-agent.amazonaws.com (inspector-agent.amazonaws.com)... 99.86.126.99, 2600:9000:21ca:c200:4:6195:f54e:b641,
2600:9000:21ca:8e00:4:6195:f54e:b641, ...
Connecting to inspector-agent.amazonaws.com (inspector-agent.amazonaws.com)[99.86.126.99]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 31083 (30K)
Saving to: 'install'

100%[=====>] 31,083  --.-K/s  in 0s

2020-09-25 13:49:17 (415 MB/s) - 'install' saved [31083/31083]

[ec2-user@ip-10-10-10-79 ~]$ curl -O https://inspector-agent.amazonaws.com/linux/latest/install.sig
% Total % Received % Xferd Average Speed Time Time Current
Dload Upload Total Spent Left Speed
100 542 100 542 0 0 28526 0 --:--:-- --:--:-- --:--:-- 30111

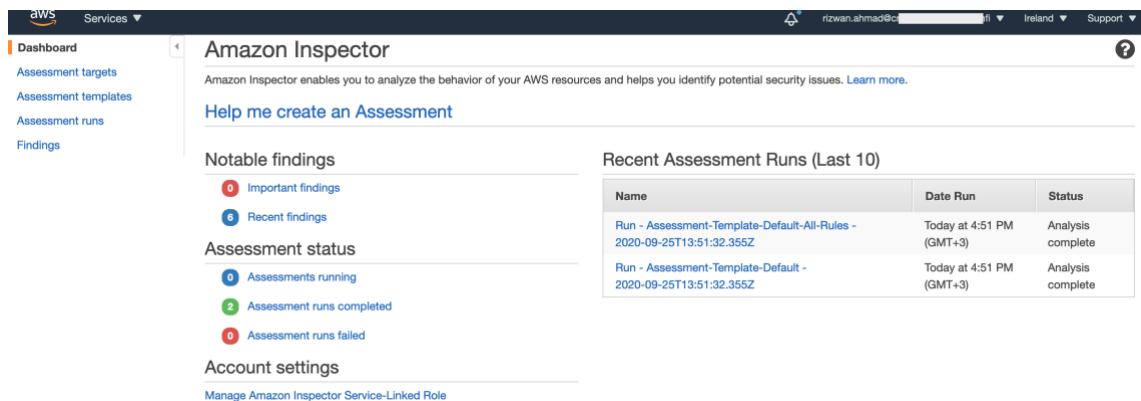
[ec2-user@ip-10-10-10-79 ~]$ sudo bash install
amzn.2.2
Distribution of the machine is Amazon Linux.
Distribution type of the machine is amzn.
Revision of the distro is 2.
Kernel version of the machine is 4.14.193-149.317.amzn2.x86_64.
Running transaction
  Installing : AwsAgent-1.1.1649.0-102649.x86_64 1/1
Redirecting to /bin/systemctl reload crond.service
  Verifying : AwsAgent-1.1.1649.0-102649.x86_64 1/1

Installed:
  AwsAgent.x86_64 0:1.1.1649.0-102649

Complete!
```

Figure 25. AWS Inspector Installation

After the successful installation of inspector agent, AWS inspector assessment has been done and the findings are shown below.



The screenshot shows the AWS Inspector dashboard. The left sidebar contains navigation links: Dashboard, Assessment targets, Assessment templates, Assessment runs, and Findings. The main content area is titled 'Amazon Inspector' and includes a 'Help me create an Assessment' button. Below this, there are sections for 'Notable findings' (with links for Important and Recent findings), 'Assessment status' (with indicators for Assessments running, Assessment runs completed, and Assessment runs failed), and 'Account settings' (with a link to Manage Amazon Inspector Service-Linked Role). On the right, the 'Recent Assessment Runs (Last 10)' table is displayed.

Name	Date Run	Status
Run - Assessment-Template-Default-All-Rules - 2020-09-25T13:51:32.355Z	Today at 4:51 PM (GMT+3)	Analysis complete
Run - Assessment-Template-Default - 2020-09-25T13:51:32.355Z	Today at 4:51 PM (GMT+3)	Analysis complete

Figure 26. Inspector dashboard

Amazon Inspector - Findings

Findings are potential security issues discovered after Amazon Inspector runs an assessment against a specified assessment target. [Learn more.](#)

✖ Filters: [{"assessmentRunArns": [{"arn": "aws:inspector:eu-west-1:618832500520:target/0-RwzLzYn3/template/0-27o8zvrn/run/0-MlBQeYp"}, {"arn": "aws:inspector:eu-west-1:618832500520:target/0-tLheyTKT/template/0-tRyL4MnI/run/0-5LU6oF7"}]}

Add/Edit attributes Last updated on September 25, 2020 5:21:14 PM (3m ago)

Filter Viewing 1-6 of 6

<input type="checkbox"/>	Severity	Date	Finding	Target	Template	Rules Package
<input type="checkbox"/>	Medium	Today at 4:5...	On instance i-0508f25bc98073aa, TCP port 22 w...	Assessment-Targe...	Assessment-Temp...	Network Reachability-
<input type="checkbox"/>	Medium	Today at 4:5...	On instance i-0508f25bc98073aa, TCP port 22 w...	Assessment-Targe...	Assessment-Temp...	Network Reachability-
<input type="checkbox"/>	Low	Today at 4:5...	On instance i-0508f25bc98073aa, TCP port 443 ...	Assessment-Targe...	Assessment-Temp...	Network Reachability-
<input type="checkbox"/>	Low	Today at 4:5...	On instance i-0508f25bc98073aa, TCP port 443 ...	Assessment-Targe...	Assessment-Temp...	Network Reachability-
<input type="checkbox"/>	Informational	Today at 4:5...	Aggregate network exposure: On instance i-0508f...	Assessment-Targe...	Assessment-Temp...	Network Reachability-
<input type="checkbox"/>	Informational	Today at 4:5...	Aggregate network exposure: On instance i-0508f...	Assessment-Targe...	Assessment-Temp...	Network Reachability-

Max records per page: 25 *refresh browser to reflect change

Figure 27. Inspector findings

As one can see in the above figures, AWS inspector finding dashboard is showing the result of its finding with severity level, ports and with additional details.

6.3.4 Data Layer Security Automation

AWS EBS, RDS and S3-SSE encryption was implemented to secure the data at rest and AWS ACM (Amazon Certificate Manager) service was used to protect data in transit through SSL certificates. A specific SCP (Service Control Policy) was created to enforce encryption on EBS, RDS and AWS lambda function was created to encrypt all S3 buckets automatically. AWS config service was also configured to send automated alerts for non-encrypted EBS, RDS and S3 resources to security notification channel. Data Layer security SCP policy snippet is below, full code can be found in appendix.

```
{
  "Sid": "EBSEncryption",
  "Effect": "deny",
  "Action": "ec2:CreateVolume",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "ec2:Encrypted": "false"
    }
  }
},
{
  "Sid": "RestrictEc2MountUnencryptedVolume",
  "Effect": "Deny",
  "Action": "ec2:RunInstances",
  "Resource": "arn:aws:ec2:*:*:volume/*",
  "Condition": {
    "Bool": {
      "ec2:Encrypted": "false"
    }
  }
}
```

Figure 28. Data-layer security

6.3.5 Incident Response Automation

The following incident response automation was configured to send alerts to the relevant team according to the activity. Incident response automation code snippet is below and full code can be found in appendix.

- Root user login activity
- Unauthorized login attempts
- SSH rejected attempts
- Audit trail changes
- VPN state monitoring
- Billing alerts

```

RejectedSSHAlarm:
  Type: 'AWS::CloudWatch::Alarm'
  Properties:
    AlarmName: cwalarm_rejected_ssh
    AlarmDescription: >-
      A CloudWatch Alarm that triggers when there are rejected SSH connections
      in a VPC.
    MetricName: RejectedSSHCount
    Namespace: VPCFlowLogsMetrics
    Statistic: Sum
    Period: '3600'
    EvaluationPeriods: '1'
    Threshold: '10'
    ComparisonOperator: GreaterThanOrEqualToThreshold
    AlarmActions:
      - Ref: SnsTopic
    TreatMissingData: notBreaching
  MetricFilter:
    Type: 'AWS::Logs::MetricFilter'
    Properties:
      LogGroupName: ''
      FilterPattern: >-
        [version, account, eni, source, destination, srcport, destport="22",
        protocol="6", packets, bytes, windowstart, windowend, action="REJECT",
        FlowLogstatus]
    MetricTransformations:
      - MetricValue: '1'
        MetricNamespace: VPCFlowLogsMetrics
        MetricName: RejectedSSHCount
  
```

Figure 29. Incident response

6.3.6 IAM Automation

The following IAM automation was configured to enforce the IAM policies according to the organization requirements through CloudFormation scripts.

- Username must be the user corporate email address.
- Users must get the permissions through groups and no permission policies should be attach directly to the user.
- Password policies (minimum length, rotation, complexity) are similar to corporate AD.
- MFA (multi-factor authentication) is enforced to get access to account resources.
- IAM Roles are configured to access cross account resources.

```

{
  "Sid": "DenyAllExceptListedIfNoMFA",
  "Effect": "Deny",
  "NotAction": [
    "iam:CreateVirtualMFADevice",
    "iam:EnableMFADevice",
    "iam:GetUser",
    "iam:ListMFADevices",
    "iam:ListVirtualMFADevices",
    "iam:ResyncMFADevice",
    "sts:GetSessionToken",
    "iam:ListUsers",
    "iam:CreateLoginProfile",
    "iam:ChangePassword"
  ],
  "Resource": "*",
  "Condition": {
    "BoolIfExists": {
      "aws:MultiFactorAuthPresent": "false"
    }
  }
}

```

Figure 29. IAM enforcement

6.3.7 Logging and Monitoring Automation

AWS CloudWatch, CloudTrail, Config and AWS Security Hub were configured to monitor real time events, logging, auditing and alerting. To automate these logging and monitoring for new and existing accounts CloudFormation scripts were created.

A central logging account was created to archive logs from all AWS accounts. AWS CloudTrail and AWS Config services were used to send logs to central logging account. A dedicated security account was created for security and governance monitoring. AWS config service in this account was showing all aggregated configuration of all accounts in a single dashboard and compliance status. A snippet from CloudFormation script is below and full script can be found from appendix.

```

AuthorizerDub:
  Type: "AWS::Config::AggregationAuthorization"
  Properties:
    AuthorizedAccountId: !Ref SecurityAccountId
    AuthorizedAwsRegion: eu-west-1

AuthorizerFra:
  Type: "AWS::Config::AggregationAuthorization"
  Properties:
    AuthorizedAccountId: !Ref SecurityAccountId
    AuthorizedAwsRegion: eu-central-1

```

Figure 30. Logging and monitoring

The following Figure 31 is showing that how AWS Config is getting configuration data from other AWS accounts and regions. Once aggregator was configured and authorized to get Config data from an account, it can be visible in aggregated view dashboard.

AWS Config

- Dashboard
- Conformance packs
- Rules
- Resources
- Advanced query
- Settings
- Authorizations

Aggregated view

- Rules
- Resources
- Aggregators**

What's new [↗](#)

Learn More

Documentation [↗](#)

Partners [↗](#)


FADs [↗](#)

Pricing [↗](#)

Aggregators

An aggregator is an AWS Config resource type that collects AWS Config data from multiple accounts and regions data recorded in AWS Config for multiple accounts and regions.

[Hide diagram](#)



Accounts and regions

Select the source accounts and regions from where you want to collect AWS Config data.

AWS Config data

Collection of AWS Config data from multiple source accounts and regions.

Aggregator

Contains the resource configuration information and the compliance data recorded in AWS Config.

Aggregated view

View all compliant and non-compliant data and resources for each aggregator.

[Add aggregator](#) Actions

Aggregator name	Source accounts
SecurityAccountAggregator	41 account(s)

AWS Config

- Dashboard
- Conformance packs
- Rules
- Resources
- Advanced query
- Settings
- Authorizations

Aggregated view

- Rules
- Resources
- Aggregators

What's new [↗](#)

Resources

Total resource count 3081

Top 10 resource types	Total
IAM Role	2660
CloudFormation Stack	122
SNS Topic	45
KMS Key	37
RDS DBSecurityGroup	35
Lambda Function	35

Figure 31. Monitoring aggregated view

7 Results and Analysis

This chapter describes the tests, results and analysis of the tested methods in detail. AWS organization service control policies are implemented at top level of different OU's. The following Figure 32 is showing the test results of Dev-OU applied SCP policy. This policy is only allowing provisioning of "micro" and "nano" EC2 (Virtual machine) instances. As you can see in the results that when a user is provisioning "t2.micro" instance it was successfully launched but when same user is trying to provision t2.small instance, it was unsuccessful and access was denied because Dev-OU SCP was configured to limit provisioning of bigger instances.

```

R00T $aws ec2 run-instances --image-id ami-0bb3fad3c0286ebd5 --count 1 --instance-type t2.micro --key-name DevKP --security-group-ids sg-067bcd3ddf64a4199
--subnet-id subnet-09de7ba6e68169d85 --region eu-west-1 --profile Dev
{
  "Groups": [],
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-0bb3fad3c0286ebd5",
      "InstanceId": "i-0375d9e8be5070108",
      "InstanceType": "t2.micro",
      "KeyName": "DevKP",
      "LaunchTime": "2020-11-02T16:04:36.000Z",
      "Monitoring": {
        "State": "disabled"
      },
      "Placement": {
        "AvailabilityZone": "eu-west-1a",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-10-0-0-133.eu-west-1.compute.internal",
      "PrivateIpAddress": "10.0.0.133",
      "ProductCodes": [],
    }
  ]
}

R00T $aws ec2 describe-instance-status --profile Dev --instance-ids i-0375d9e8be5070108
{
  "InstanceStatuses": [
    {
      "AvailabilityZone": "eu-west-1a",
      "InstanceId": "i-0375d9e8be5070108",
      "InstanceState": {
        "Code": 16,
        "Name": "running"
      },
      "InstanceStatus": {
        "Details": [
          {
            "Name": "reachability",
            "Status": "passed"
          }
        ],
        "Status": "ok"
      },
      "SystemStatus": {
        "Details": [
          {
            "Name": "reachability",
            "Status": "passed"
          }
        ],
        "Status": "ok"
      }
    }
  ]
}

R00T $aws ec2 run-instances --image-id ami-0bb3fad3c0286ebd5 --count 1 --instance-type t2.small --key-name DevKP --security-group-ids sg-067bcd3ddf64a4199 |
--subnet-id subnet-09de7ba6e68169d85 --region eu-west-1 --profile Dev
An error occurred (UnauthorizedOperation) when calling the RunInstances operation: You are not authorized to perform this operation. Encoded authorization
failure message: 3360dM0awzWOK5bpmpl5ZszRshQOPztApe8st_Td1xDOSxk9tbgeatU1bJoaGuWQCuin8_pun8fouZwbbjFVWnm53vjp9aHYJ8rGFTp8P5jPpqhWrfpzEFRD023CvSdI1QFNvvo19
JIHuym5kMe-eT0k9wP1q8iQ3LS5A9-AVpDr8tXMYu6696I-PaIQvxfDmp9bKQZfzqSkRq1111kYejEuicDQ001p9yQmXypkNFZudhBfBhV6yQU7pdCZv9Mhax1p6rpg0tvaGfnf-zJIdfAayrQhK3xG0r
i_bUHAIVfK21j29oKHQCFXZ0z7b1z8NpMvnu0GMQ7vZ0eqTUGh6u0jL24bjxJM5Yp1wqN6y7hrp80zQmNkaKxkZQ1M9g-INfEsT4xnzhtxInlg8glnq918s1P_r9mkILUE4tV7gr0_10W2IGp1pHX9DstRa
AM13jZKMytgyaA_93mXb1LOYeex1d05ZZ0NnQZMhXhXiaKU_potaiGuenzrj7_3a1_R0mDnu-nUUVKEZFs74Ej14189ad3e0A3YU0GKF_CXjVOZMAHLjKEVZ5-jWu5t0CSmdljVatjprRHE1zeB1l
cjU3PQ06MHWXpNtCjI_lYas3tRame-03NABBEI5AL3CviOW1_06-ISaw-bEd0Cvwm1p3jUfV1LwolxdhRGu7Wm19WP9cVB6zSE9c_4nv6et0_8Nd2qV5gH60fp0t0LLpk8X6D6hnc72VxMEN7QuUsq9
3JxYvX_3oDNfy1Y0jGrDV_v2RZfOVFPx5SDhsA

```

Figure 32. Dev-OU results

Prod-OU service control policy is not restricting the instance provisioning based on size and the Figure 33 is showing the results that bigger instances can be provisioned in Prod-OU.

```
[R@@T $aws ec2 run-instances --image-id ami-0bb3fad3c0286ebd5 --count 1 --instance-type t2.xlarge --key-name ProdKP --security-group-ids sg-0f3bd87bf6d00677b]
--subnet-id subnet-00198e42100010bd4 --region eu-west-1 --profile Prod
{
  "Groups": [],
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-0bb3fad3c0286ebd5",
      "InstanceId": "i-012f310d173a283a0",
      "InstanceType": "t2.xlarge",
      "KeyName": "ProdKP",
      "LaunchTime": "2020-11-02T19:36:59.000Z",
      "Monitoring": {
        "State": "disabled"
      },
      "Placement": {
        "AvailabilityZone": "eu-west-1b",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-10-0-2-206.eu-west-1.compute.internal",
      "PrivateIpAddress": "10.0.2.206",
      "ProductCodes": [],
      "PublicDnsName": "",
      "State": {

```

Figure 33. Prod-OU results

Core-OU service control policy is configured to secure the log account and audit account resources and restricting access to resources only by assuming a specific account role `AwsOpsExecution`. The following Figure 34 is showing that Test user has administrator access to Log_Account but SCP is protecting the log bucket and Test user is unable to change the bucket encryption.

```
[R@@T $aws sts get-caller-identity --profile LogAcc
{
  "UserId": "AIDA5PXCSVR3V33W3YOFD",
  "Account": "927",
  "Arn": "arn:aws:iam::927:user/test"
}
R@@T $aws iam list-attached-user-policies --user-name test --profile LogAcc
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    }
  ]
}
[R@@T $aws s3api get-bucket-encryption --bucket $LogAccBucket --profile LogAcc
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
[R@@T $aws s3api delete-bucket-encryption --bucket $LogAccBucket --profile LogAcc
{
  "An error occurred (AccessDenied) when calling the DeleteBucketEncryption operation: Access Denied"
}
R@@T $
```

Figure 34. Core-OU results

Let's try change the bucket encryption policy by using that specific `AwsOpsExecution` account role. The Figure 35 is showing that by using the specific role, bucket encryption was disabled and enabled successfully.

```

R@@T $aws s3api delete-bucket-encryption --bucket $LogAccBucket --profile LogAccRole Bucket encryption is deleted by using Role
R@@T $aws s3api get-bucket-encryption --bucket $LogAccBucket --profile LogAccRole
An error occurred (ServerSideEncryptionConfigurationNotFoundError) when calling the GetBucketEncryption operation: The server side encryption configuration was not found
R@@T $aws s3api put-bucket-encryption --bucket $LogAccBucket --server-side-encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}' --profile LogAccRole Bucket encryption is enabled by using Role
R@@T $aws s3api get-bucket-encryption --bucket $LogAccBucket --profile LogAccRole
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
LogAccRole profile configuration
[LogAccRole]
role_arn = arn:aws:iam::927:role/AwsOpsExecution
source_profile = default
region = eu-west-1

```

Figure 35. Core-OU OPS-Role results

After testing the OU's policies, Let's test the account level security baseline configuration. To protect the network layer, customized NACL and security groups are created and configured. NACL are created to implement another layer of security to infrastructure to filter the traffic before it reaches to security groups. The test is conducted between two different VPC's. Test-VPC has two different subnets and IP-ranges. Prod-VPC is attached to a NACL that is allowing only SSH from 10.0.0.0/24 which is one of Test-VPC subnet. Prod-VPC instance security group is allowing SSH traffic from any IP which is configured temporarily to test NACL.

The Figure 36 is showing the results that Prod-VPC instance 10.10.11.177 is only accessible from Test-VPC instance whose IP is in the NACL allowed range.

```

R@@T $aws ec2 describe-instances \
> --query 'Reservations[*].Instances[*].{Instance:InstanceId,PrivateIP:PrivateIpAddress,Subnet:SubnetId,Vpc:VpcId}' \
> --output table --profile Prod
-----
|                                     DescribeInstances                                     |
|-----|-----|-----|-----|-----|
| Instance | PrivateIP | Subnet | Vpc | |
|---|---|---|---|---|
| i-0da67be435cb62297 | 10.0.0.97 | subnet-0b5649bd6ccb9f3d9 | vpc-0dabc099f731e3503 | Test-VPC |
| i-00e6a7c9970cc2761 | 10.10.11.177 | subnet-0e24a588b24728b49 | vpc-0109d1eb19b1a4321 | Prod-VPC |
| i-0be0b1a766561fd26 | 10.0.2.71 | subnet-00198e42100010bd4 | vpc-0dabc099f731e3503 | Test-VPC |
|-----|-----|-----|-----|-----|

R@@T $aws ec2 describe-security-groups \
> --filters Name=tag:Name,Values=NetworkBaseline \
> --query "SecurityGroups[*].{SGname:GroupName,SGid:GroupId}" \
> --output table --profile Prod
-----
|                                     DescribeSecurityGroups                                     |
|-----|-----|
| SGid | SGname |
|-----|-----|
| sg-0333f7c8365567d9f | Web-DMZ |
| sg-0ab69673f31090486 | SSH-Test-SG |
|-----|-----|

R@@T $aws ec2 describe-security-groups \
> --filter Name=ip-permission.from-port,Values=22 \
> Name=ip-permission.to-port,Values=22 Name=ip-permission.cidr,Values='0.0.0.0/0' \
> --query "SecurityGroups[*].{SGname:GroupName,SGid:GroupId}" \
> --output table --profile Prod
-----
|                                     DescribeSecurityGroups                                     |
|-----|-----|
| SGid | SGname |
|-----|-----|
| sg-0333f7c8365567d9f | Web-DMZ |
| sg-03748d5c81e6892ee | launch-wizard-1 |
| sg-09d08345f48c955bb | ELK Certified by Bitnami-7-6-2-6-r01-AutogenByAWSMP- |
| sg-0ab69673f31090486 | SSH-Test-SG |
| sg-0bb3b73b755d392db | OpenVPN- - Open Source VPN solution powered by TurnKey GNU-Linux -HVM--15-1-AutogenByAWSMP- |
| sg-0f0cc2bdc04b3609e | Cisco Cloud Services Router -CSR- 1000V - Security Pkg- Max Performance-16-12-1a-AutogenByAWSMP- |
| sg-0f3bd87bf6d00677b | OpenVPN- - Open Source VPN solution powered by TurnKey GNU-Linux -HVM--15-1-AutogenByAWSMP-1 |
|-----|-----|
Securitygroups that allows ssh from any IP

```

Figure 36. Network security

The Figure 37 is showing that Prod instance can be accessed from Test-VPC instance whose IP was 10.0.0.97 because NACL is not blocking access from this IP range. Test-VPC second instance whose IP was 10.0.2.71 cannot access Prod-VPC instance because NACL is blocking it as VPC-flow log is showing in the figure.

```
[ec2-user@ip-10-0-2-71 ~]$ ssh -i Prod-KP.pem ec2-user@10.10.11.177
ssh: connect to host 10.10.11.177 port 22: Connection timed out
[ec2-user@ip-10-0-2-71 ~]$ exit
```

Timestamp	Message
2020-11-05T12:09:04.000+02:00	2 64:.....53 eni-0191438ff19e086b6 10.0.2.71 10.10.11.177 56010 22 6 4 240 1604570944 1604570990 REJECT OK

```
[ec2-user@ip-10-0-0-97 ~]$ ssh -i Prod-KP.pem ec2-user@10.10.11.177
The authenticity of host '10.10.11.177 (10.10.11.177)' can't be established.
ECDSA key fingerprint is SHA256:ahx1i7LDLUN6foZegPI3I5FqAlQwEmRTxML/8cNxEOE.
ECDSA key fingerprint is MD5:5f:00:20:70:23:fd:74:71:64:71:89:22:20:e4:18:6d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.11.177' (ECDSA) to the list of known hosts.
```

```
  __|  __|_ )
  _| (  /   Amazon Linux 2 AMI
 ---|\---|---
```

```
https://aws.amazon.com/amazon-linux-2/
25 package(s) needed for security, out of 39 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-10-11-177 ~]$
```

Timestamp	Message
2020-11-05T12:11:52.000+02:00	2 645..... eni-0191438ff19e086b6 10.10.11.177 10.0.0.97 22 55148 6 24 4421 1604571112 1604571170 ACCEPT OK
2020-11-05T12:11:52.000+02:00	2 645..... eni-0191438ff19e086b6 10.0.0.97 10.10.11.177 55148 22 6 22 4129 1604571112 1604571170 ACCEPT OK

Figure 37. Network flow log

Application layer is secured by AWS WAF and AWS inspector. The following figure is showing AWS WAF and AWS inspector test results. The result is showing that AWS WAF is mitigating most common application like attacks SQL-injection and XSS.

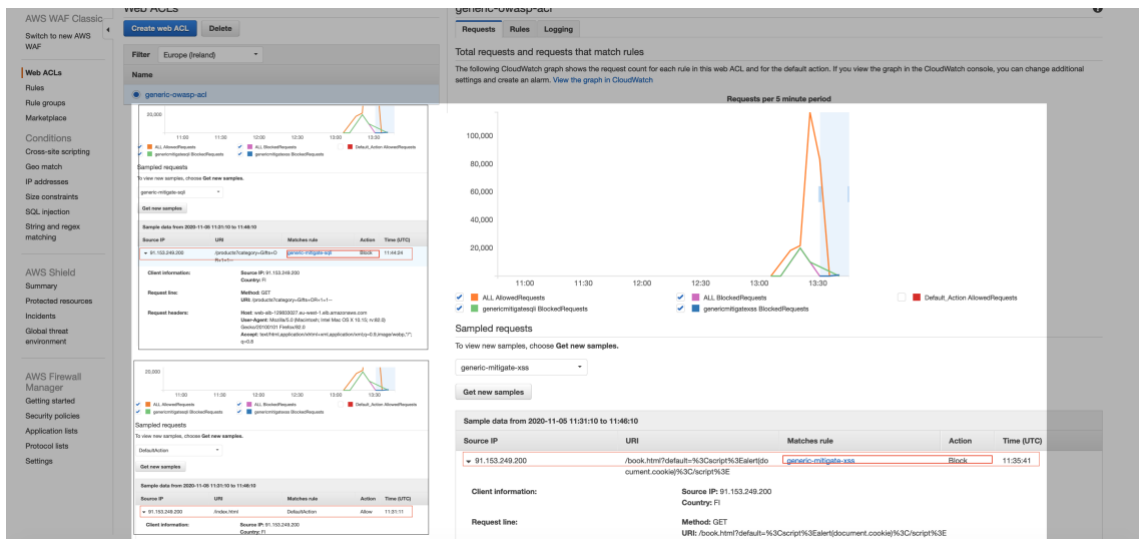


Figure 38. Web Application Firewall results

The following figure is showing AWS inspector assessment results for instance network exposure on port 22 and 443.

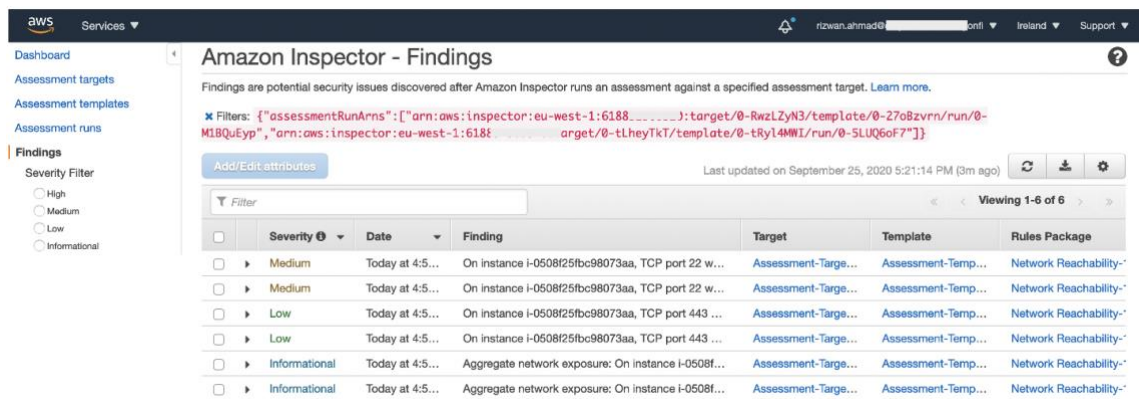


Figure 39. AWS inspector results

Data layer is secured by SCP policies for S3, EBS and RDS. The following figure is showing the S3 encryption enforcement. As the result is showing, Put-object operation was denied if bucket is not encrypted. After enabling the encryption on bucket, Put-object operation was successful.


```

R00T $aws s3api create-bucket --bucket test-riz-data --p Shape Style | --region eu-west-1 --create-bucket-configuration LocationConstraint=eu-west-1
{
  "Location": "http://test-riz-data.s3.amazonaws.com/"
}
R00T $aws s3api get-bucket-encryption --bucket test-riz-data --profile Prod Bucket encryption checking
An error occurred (ServerSideEncryptionConfigurationNotFound) when calling the GetBucketEncryption operation: The server side encryption configuration was not found
R00T $aws s3api put-object --bucket test-riz-data --key testfiles/test-file-upload --body ./S3upload.txt --profile Prod Uploading objects while bucket encryption is not enabled
An error occurred (AccessDenied) when calling the PutObject operation: Access Denied
R00T $aws s3api put-bucket-encryption --bucket test-riz-data --server-side-encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}' --profile Pr
R00T $aws s3api get-bucket-encryption --bucket deepsecurity-av-data --profile Prod Enabling Bucket encryption
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
R00T $aws s3api put-object --bucket test-riz-data --key testfiles/test-file-upload --body ./S3upload.txt --profile Prod
{"ETag": "\"c21a12b78867677b477f00362726679\"", "ServerSideEncryption": "AES256"}
R00T $aws s3api list-objects --bucket test-riz-data --profile Prod
{
  "Contents": [
    {
      "Key": "testfiles/test-file-upload",
      "LastModified": "2020-11-06T13:44:36.000Z",
      "ETag": "\"c21a12b78867677b477f00362726679\"",
      "Size": 27,
      "StorageClass": "STANDARD",
      "Owner": {
        "ID": "ea415c27fd23b1e5d28e2342dd5981735a1fcd1e993889cce7833aa12949e275"
      }
    }
  ]
}

```

Figure 40. Data security S3-results

The following figure is showing the test result of enforcing EBS encryption. EC2 instance creation was unsuccessful when attached EBS volume encryption was disable. Once it was specified that EBS volume encryption is enable, the instance provision successfully.

```

R00T $aws ec2 run-instances --image-id ami-0bb3fad3c0286ebd5 --count 1 --instance-type t2.micro --region eu-west-1 \
> --key-name Prod-KP --subnet-id subnet-0e24a588b24728b49 --security-group-ids sg-0ab69673f3109048e \
> --output table --profile Prod Without encrypted volume
An error occurred (UnauthorizedOperation) when calling the RunInstances operation: You are not authorized to perform this operation. Encoded authorization failure message: Whj5fH89h1tdgFUIUy9f2mepDvI
Yk2a7FncZrooedwFAf1Ri2ZrTgl-m0D1Dn_YWm8n783RA5dga4W0nRkCIdIwrrvYpZj2oz28d8BseZl0zAe0In9MtoYBkCH9RkYkUkU_mJk0yRoa4k865Gz2Fp0TvU0U24xv0dy0e3J0mJ3Pw8BY10GZIF900rY8l1N2YfSnl10C719r1aGf.
p1bKrsVurTC1owJ5wR1-up1z18220h2Q2a8f9YKN_Op6Shk8NwW8P6pM5yWqZ1Z5Nj0yYtg1WzEwIehbeCmkxg8K3PpBac2szk0gC_xhmmA0dWg0Jgnt0DPX24_up0rFXE_u660FNASE99k3RzB2evnCRKYsF0MfP5C421090F8_B7y0p1Hw5-jETr7tFNCI
o-yd_YKZV7MNNeqJhC87XRPVlJ7eFa0J4n14wVIIur0TvesJMsE6JFN7Kb1Whzj-0Druz1eGxFF0Q0InBAlPouxXMI0u0eTbX37nGCKF2AEALMht-jf5GRClio50wFwiohMw19h_hx1Lr5yWpm23_vA3NJaji0KnlhGLDwTMSvXSSD_AyuaUKBoSc9A700KS:
kgzza_R0K0QwP_C1830UcP5yIayZbn08TQ56s10btz-8xz3gknj5Tm1zha3UnsDANvNMe00ma1MP6K0Gj3B0_S0ZGTq1vekS0xeze_qiYbxrEb1FK5G080hrobyANC_81YfydEt-DrYECIYot69NL5XN_Ege7YdayuCb6Sed.I
R00T $
R00T $aws ec2 run-instances --image-id ami-0bb3fad3c0286ebd5 --count 1 --instance-type t2.micro --region eu-west-1 \
> --key-name Prod-KP --subnet-id subnet-0e24a588b24728b49 --security-group-ids sg-0ab69673f3109048e \
> --block-device-mappings file://mapping.json --output table --profile Prod With encrypted volume
R00T $cat mapping.json mapping.json file config
{
  "DeviceName": "/dev/xvda",
  "Ebs": {
    "Encrypted": true
  }
}

```

RunInstances	
OwnerId	64514717643
ReservationId	r-8dd563a8799795b71

Instances	
AmiLaunchIndex	0
Architecture	x86_64
ClientToken	
EbsOptimized	False
EnaSupport	True
Hypervisor	xen
ImageId	ami-0bb3fad3c0286ebd5
InstanceId	i-0c0214466196ce99
InstanceType	t2.micro
KeyName	Prod-KP
LaunchTime	2020-11-06T13:45:13.000Z
PrivateDnsName	ip-10-10-11-201.eu-west-1.compute.internal
PrivateIpAddress	10.10.11.201
PublicDnsName	
RootDeviceName	/dev/xvda
RootDeviceType	ebs
SourceDestCheck	True
StateTransitionReason	
SubnetId	subnet-0e24a588b24728b49
VirtulizationType	hvm
VpcId	vpc-01899deb1961a4321

Figure 41. Data security EBS-results

The following Figure 42 is showing incident response for different incidents as mentioned in the previous chapter such as unauthorized login attempts, root login activity and audit trail changes. To test that these incidents alerts were configured properly, unauthorized login attempts, root login and audit trail changes had been done and based on the metrics of these activity responses alerts were sent to the specified security notification group.

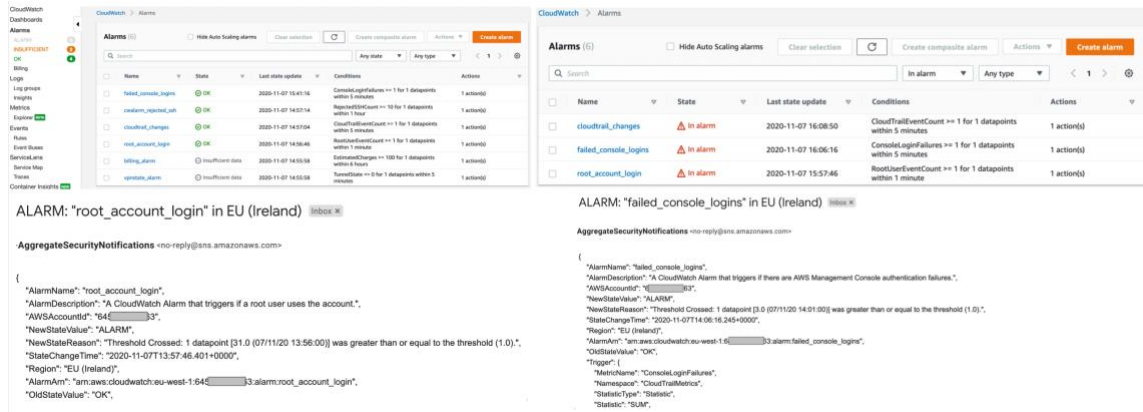


Figure 42. Incident response results

The following Figures 43-44 are showing IAM policies enforcement. As the result is showing that IAM user had not attached any inline policies directly. User had assigned all the permissions through Admin group and Admin policy was attached to the group. The result is also showing that user was able to access account resources successfully but once MFA enforcement policy was attached to Admin group, user could not access account resources because user had not configured MFA on this account.

MFA policy did not allow user to access account resources. User need to enable MFA authentication to access account resources. Once the user register MFA device and logged in with MFA, results is showing that user was able to access account resources. User generated temporary session token with MFA code to access the account resources from CLI.

```

R@@T $aws iam list-user-policies --user-name test@org.com --profile IAM-Admin
{
  "PolicyNames": []
}
R@@T $aws iam list-groups-for-user --user-name test@org.com --output table --profile IAM-Admin List user groups
-----
|                                     ListGroupsForUser                                     |
|-----|-----|-----|-----|-----|
|                                     Groups                                     |
|-----|-----|-----|-----|-----|
| Arn                               | CreateDate | GroupId   | GroupName | Path  |
|-----|-----|-----|-----|-----|
| arn:aws:iam::[redacted]:group/Admins | 2020-05-09T17:21:35Z | AGPA6DWLATUKCBLN6SS4Y | Admins   | /    |
|-----|-----|-----|-----|-----|
R@@T $aws iam list-attached-group-policies --group-name Admins --output table --profile IAM-Admin List user group policies
-----
|                                     ListAttachedGroupPolicies                                     |
|-----|-----|-----|-----|-----|
|                                     AttachedPolicies                                     |
|-----|-----|-----|-----|-----|
| PolicyArn                         | PolicyName  | | | |
|---|---|---|---|---|
| arn:aws:iam::aws:policy/AdministratorAccess | AdministratorAccess |
|-----|-----|-----|-----|-----|

```

Figure 43. IAM results

```

R00T $aws s3api list-objects --bucket test-bucket-for-testuser-iam --output table --profile IAM-TestUser
-----
|                               ListObjects                               |
|-----|-----|-----|-----|-----|
|                               Contents                               |
|-----|-----|-----|-----|-----|
| ETag                               | Key                               | LastModified                | Size | StorageClass |
|-----|-----|-----|-----|-----|
| "d12c4cdeeee484b3a9e8e4df60eaa5d7" | testfile                          | 2020-11-07T15:35:11.000Z    | 25  | STANDARD      |
|-----|-----|-----|-----|-----|

R00T $aws iam attach-group-policy --policy-arn arn:aws:iam::9761:policy/Enforce_MFA --group-name Admins --profile IAM-TestUser MFA enforcement policy attached to admin group
R00T $aws s3api list-objects --bucket test-bucket-for-testuser-iam --profile IAM-TestUser --output table
Access denied because MFA is not enabled for user.
An error occurred (AccessDenied) when calling the ListObjects operation: Access Denied
R00T $aws iam enable-mfa-device --user-name test@org.com --serial-number arn:aws:iam::9761:mfa/test@org.com --authentication-code1 984659 --authentication-code2 825918 --profile IAM-Admin
Attaching MFA device to user

R00T $aws sts get-session-token --serial-number arn:aws:iam::9761:mfa/test@org.com --token-code 998298 --profile IAM-TestUser
Authorizing user with MFA code to get temporary CLI credentials
{
  "Credentials": {
    "AccessKeyId": "ASIA6DWLATAKATETTKOH",
    "SecretAccessKey": "Za5mYTI14381Cq+8WveV19BUrth4lea5RQuyN1",
    "SessionToken": "Fw0ZXiVYXdzENK//////////EaDE2110Z83NrzW1iWCKGAZ52bJzJWXXH711+hnpV6ExnNq12En9gRmrIru5NcbYtXb1W995T6resVj1Mtuc3yDgpxZc/yM71f6EZESXHPsiXtKIeJf0xdgps50r1Z7vVgXadr0MPdmlLubia05vw183Z4H8dukZjZz3NpPa4MvabmAAw9NT1+S2S480YsEkFiyKLKlM/0FMIjHurZZj0Bg3Q+Ta4+bsu09Y31Y080HwqTrq923a4Fy88nTGoLBAhL",
    "Expiration": "2020-11-08T04:05:06Z"
  }
}
Attaching MFA device to user

R00T $aws s3api list-objects --bucket test-bucket-for-testuser-iam --output table --profile mfa
-----
|                               ListObjects                               |
|-----|-----|-----|-----|-----|
|                               Contents                               |
|-----|-----|-----|-----|-----|
| ETag                               | Key                               | LastModified                | Size | StorageClass |
|-----|-----|-----|-----|-----|
| "d12c4cdeeee484b3a9e8e4df60eaa5d7" | testfile                          | 2020-11-07T15:35:11.000Z    | 25  | STANDARD      |
|-----|-----|-----|-----|-----|

Configuration of mfa profile with temp credentials
R00T $aws sts get-caller-identity --profile mfa
{
  "UserId": "AIDA6DWLATAKOSQWR6RU",
  "Account": "976148",
  "Arn": "arn:aws:iam::9761:user/test@org.com"
}

```

Figure 44. IAM results

The Figure 45 is showing the results for logging and monitoring. A central logging account was created for ease of archiving, integrity and limited access to logs. All logs were saved to S3 bucket, As the figure is showing the format of archiving logs:

S3-Bucket-name -> Organization-Id -> AWSLogs -> AWS-account-Id -> CloudTrail -> AWS-region -> Year -> Month -> Day -> Filename

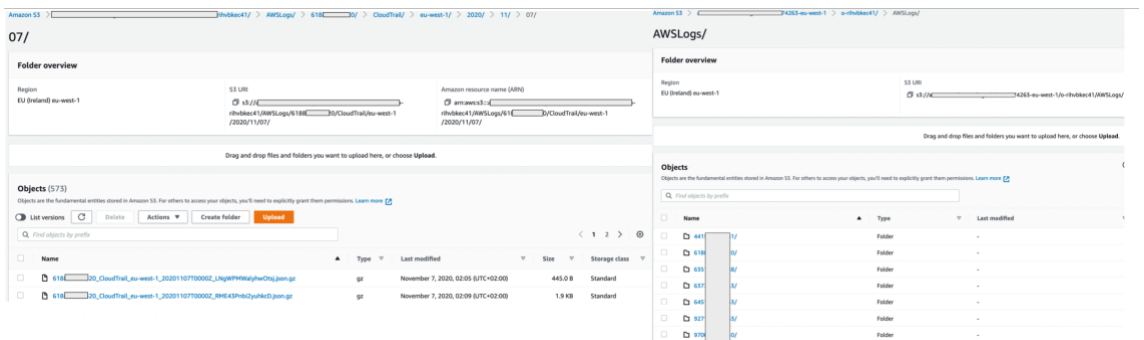


Figure 45. Log account results

The following Figure 46 is showing the results of security and monitoring compliance dashboard. A dedicated security account was created for security team to monitor AWS accounts compliance status and access to any AWS account from this account in case of emergency. As the one can see, aggregated dashboard is showing the EBS volumes and S3 resources were noncompliant and a notification email was sent to security team as well which is also showing in the figure. All the details regarding noncompliant resources were mentioned in the email such as resource-id, account-id, aws-region,

config-rule and resource-type. These compliance statuses were collected by AWS Config aggregator from different accounts.

The screenshot displays the AWS Config console interface. On the left is a navigation menu with options like Dashboard, Conformance packs, Rules, Resources, and Aggregators. The main area is titled 'Aggregated view' and shows a summary of resources (1052 total) and a 'Config rule compliance status' section. This status indicates 2 Noncompliant rules and 2 Compliant rules. A table lists noncompliant rules, including 'EBS_VOL_ENCR_CHK' in the eu-west-1 region across 64 accounts, with 4 noncompliant resources, and 'S3_ENCR_CHK' in the eu-west-1 region across 64 accounts, with 3 noncompliant resources. On the right, a 'Config Rules Compliance Change' notification is shown as a JSON object, detailing the rule name, region, account, and resource information.

Aggregated view

Region: All regions | Account: All accounts

Data collection from all source accounts and regions is incomplete. [View details.](#)

Note: Data displayed in the dashboard is received from multiple aggregation sources and is refreshed at different intervals. Data might be delayed by a few minutes.

Resources

Resource	Total
IAM Role	630
CloudFormation Stack	104
IAM User	57
SNS Topic	30
KMS Key	29

Config rule compliance status

- 2 Noncompliant rules
- 2 Compliant rules

Noncompliant rules

Rule name	Region	Account	Compliance
EBS_VOL_ENCR_CHK	eu-west-1	64	4 Noncompliant resource(s)
S3_ENCR_CHK	eu-west-1	64	3 Noncompliant resource(s)

Config Rules Compliance Change

```

AggregateSecurityNotifications <no-reply@sns.amazonaws.com>
{
  "version": "0",
  "id": "d7a586d4-3126-3bec-e826-081b5280ad62",
  "detail-type": "Config Rules Compliance Change",
  "source": "aws.config",
  "account": "645-363",
  "time": "2020-11-07T22:22:39Z",
  "region": "eu-west-1",
  "resources": [],
  "detail": {
    "resourceId": "vol-0ee4a3dfe1db5ad04",
    "awsRegion": "eu-west-1",
    "awsAccountId": "645-363",
    "configRuleName": "EBS_VOL_ENCR_CHK",
    "recordVersion": "1.0",
    "configRuleARN": "arn:aws:config:eu-west-1:645-363:config-rule/config-rule-y4fm9o",
    "messageType": "ComplianceChangeNotification",
    "newEvaluationResult": {
      "evaluationResultIdentifier": {
        "evaluationResultQualifier": {
          "configRuleName": "EBS_VOL_ENCR_CHK",
          "resourceType": "AWS::EC2::Volume",

```

Figure 46. Security account results

8 Conclusion

This research demonstrates that security automation, governance automation policies enforcement and self-development are key components of effective cloud security in enterprise cloud environments.

Cloud security assessment and responses are required to be agile as enterprises have embraced agile methodology. It is even much clearer to us now that security and governance play a decisive role as one of the most significant consideration affecting adoption of cloud computing. Based on the lack of information security experts in enterprises, it is also important to train different teams to provision secure cloud environments and implement automation as much as possible.

In this research, AWS security and governance at scale structure was implemented. With this implementation organization has achieved centrally managed scattered cloud resources, oversight of cloud implementations, policies implemented at account, organization unit and even organization level, security, governance and compliance automation with best practices.

The results of this study are indicating that automation should be undertaken by identifying security and governance tasks which can be handled by project teams themselves. Cloud security individuals must be in contact with project teams to configure baseline automation.

A routine assessment of all security tasks under an implemented security governance model is strongly recommended. The objective of the routine assessment is to identify:

- Security automation enhancements
- Security processes modifications for cloud agility
- Security tasks for individual project teams
- Trainings needed for individual project teams
- Refine security and governance policies

For future works, current organization IdP solution can be integrated with AWS and AWS logs can be send to organization SIEM system for deep inspection.

References

- [1] Darren Death, "Information Security Handbook"
Packt Publishing, 2017
- [2] Zeal Vora, "Enterprise Cloud Security and Governance"
Packt Publishing, December 2017
- [3] Shanti Bhushan, "Handbook of Research Methodology" August 2017 [Online]
Available:
https://www.researchgate.net/publication/319207471_HANDBOOK_OF_RESEARCH_METHODODOLOGY
- [4] Peter Mell, "NIST Special Publication 800-145" [Online]
Available: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>
- [5] Derrick Rountree; Ileana Castrillo, "The Basics of Cloud Computing"
Syngress, 2013
- [6] R.L. Krutz and R.D. Vines, "Cloud security : a comprehensive guide to secure cloud computing" Wiley, 2010
- [7] Michael Minelli, Michele Chambers, Ambiga Dhiraj, "Architecting the Cloud"
Wiley, January 2014
- [8] AWS Official Documentation "AWS Identity and Access Management"
Available: <https://docs.aws.amazon.com/IAM/latest/UserGuide/iam-ug.pdf>
- [9] AWS Official Documentation Security Pillar "AWS Well-Architected Framework"
Available: <https://d1.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf?ref=wellarchitected-ws>
- [10] S. Aljawarneh, "Advanced research on cloud computing design and applications"
IGI-Global, September 2015
- [11] C. Eckert, S.K. Katsikas, and G. Pernul, "Trust, privacy, and security in digital business" 11th Intl. Conf. on Security and Cryptology, Munich, Germany, Sept. 2014
- [12] Z. Wang, K. Sun, S. Jajodia, and J. Jing, "Proof of Isolation for Cloud Storage"
Secure Cloud Computing, Springer, 2014
- [13] C. Wang, Q. Wang, K. Ren, W. Lou "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing" IEEE INFOCOM, 2010
- [14] S. Pearson and G. Yee, "Privacy and security for cloud computing"
Springer, 2013.
- [15] M.G. Jaatun, G. Zhao, and C. Rong, "Cloud computing"
1st Intl. Conf. on Cloud Computing, Springer, December, 2009

- [16] Chris Dotson, "Practical Cloud Security" O'Reilly Media, Inc. March, 2019
- [17] Corey Schou, Steven Hernandez, "Information Assurance Handbook: Effective Computer Security and Risk Management Strategies" September, 2014
- [18] D.H. Sharma, C.A. Dhote, M. M. Potey, "Identity and Access Management as Security-as-a- Service from Clouds" Procedia Comput. Sci., 2016
- [19] Dottie Schindlinger and Brian Stafford "Governance in the Digital Age" Wiley, April 2019
- [20] AWS Official Documentation, "Shared Responsibility Model"
Available: <https://aws.amazon.com/compliance/shared-responsibility-model/>
- [21] AWS Multi-Account Architecture and Best Practices
Available: <https://www.slideshare.net/AmazonWebServices/aws-multiaccount-architecture-and-best-practices>

Appendices

Appendix 1: Network Baseline

AWSTemplateFormatVersion: "2010-09-09"

Description: This template deploys a VPC, two public and private subnets spread across two Availability Zones. It deploys an internet gateway, with a default route on the public subnets. It deploys a pair of NAT gateways (one in each AZ), and default routes for them in the private subnets.

Parameters:

EnvironmentName:

Description: An environment name that is prefixed to resource names

Type: String

VpcCIDR:

Description: Please enter the IP range (CIDR notation) for this VPC

Type: String

Default: 10.10.0.0/16

PublicSubnet1CIDR:

Description: Please enter the IP range (CIDR notation) for the public subnet in the first Availability Zone

Type: String

Default: 10.10.10.0/24

PublicSubnet2CIDR:

Description: Please enter the IP range (CIDR notation) for the public subnet in the second Availability Zone

Type: String

Default: 10.10.11.0/24

PrivateSubnet1CIDR:

Description: Please enter the IP range (CIDR notation) for the private subnet in the first Availability Zone

Type: String

Default: 10.10.20.0/24

PrivateSubnet2CIDR:

Description: Please enter the IP range (CIDR notation) for the private subnet in the second Availability Zone

Type: String

Default: 10.10.21.0/24

VPNIP:

Description: "VPN IP address for SSH"

Type: String

Default: 192.168.0.254/32

Resources:

VPC:

Type: AWS::EC2::VPC

Properties:

CidrBlock: !Ref VpcCIDR

EnableDnsSupport: true

EnableDnsHostnames: true

Tags:

- Key: Name

Value: !Ref EnvironmentName

InternetGateway:

Type: AWS::EC2::InternetGateway

Properties:

Tags:

- Key: Name

Value: !Ref EnvironmentName

InternetGatewayAttachment:

Type: AWS::EC2::VPCElasticNetworkInterface

Properties:

InternetGatewayId: !Ref InternetGateway

VpcId: !Ref VPC

PublicSubnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

AvailabilityZone: !Select [0, !GetAZs "]

CidrBlock: !Ref PublicSubnet1CIDR

MapPublicIpOnLaunch: true

Tags:

- Key: Name

Value: !Sub \${EnvironmentName} Public Subnet (AZ1)

PublicSubnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

```

AvailabilityZone: !Select [ 1, !GetAZs " ]
CidrBlock: !Ref PublicSubnet2CIDR
MapPublicIpOnLaunch: true
Tags:
  - Key: Name
    Value: !Sub ${EnvironmentName} Public Subnet (AZ2)
PrivateSubnet1:
Type: AWS::EC2::Subnet
Properties:
  VpcId: !Ref VPC
  AvailabilityZone: !Select [ 0, !GetAZs " ]
  CidrBlock: !Ref PrivateSubnet1CIDR
  MapPublicIpOnLaunch: false
  Tags:
    - Key: Name
      Value: !Sub ${EnvironmentName} Private Subnet (AZ1)
PrivateSubnet2:
Type: AWS::EC2::Subnet
Properties:
  VpcId: !Ref VPC
  AvailabilityZone: !Select [ 1, !GetAZs " ]
  CidrBlock: !Ref PrivateSubnet2CIDR
  MapPublicIpOnLaunch: false
  Tags:
    - Key: Name
      Value: !Sub ${EnvironmentName} Private Subnet (AZ2)
NatGateway1EIP:
Type: AWS::EC2::EIP
DependsOn: InternetGatewayAttachment
Properties:
  Domain: vpc
NatGateway2EIP:
Type: AWS::EC2::EIP
DependsOn: InternetGatewayAttachment
Properties:
  Domain: vpc
NatGateway1:
Type: AWS::EC2::NatGateway
Properties:
  AllocationId: !GetAtt NatGateway1EIP.AllocationId
  SubnetId: !Ref PublicSubnet1
NatGateway2:
Type: AWS::EC2::NatGateway
Properties:
  AllocationId: !GetAtt NatGateway2EIP.AllocationId
  SubnetId: !Ref PublicSubnet2
PublicRouteTable:
Type: AWS::EC2::RouteTable
Properties:
  VpcId: !Ref VPC
  Tags:
    - Key: Name
      Value: !Sub ${EnvironmentName} Public Routes
DefaultPublicRoute:
Type: AWS::EC2::Route
DependsOn: InternetGatewayAttachment
Properties:
  RouteTableId: !Ref PublicRouteTable
  DestinationCidrBlock: 0.0.0.0/0
  GatewayId: !Ref InternetGateway
PublicSubnet1RouteTableAssociation:
Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
  RouteTableId: !Ref PublicRouteTable
  SubnetId: !Ref PublicSubnet1
PublicSubnet2RouteTableAssociation:
Type: AWS::EC2::SubnetRouteTableAssociation
Properties:
  RouteTableId: !Ref PublicRouteTable
  SubnetId: !Ref PublicSubnet2
PrivateRouteTable1:
Type: AWS::EC2::RouteTable

```

```

Properties:
  Vpclid: !Ref VPC
  Tags:
    - Key: Name
      Value: !Sub ${EnvironmentName} Private Routes (AZ1)
DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1
PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1
PrivateRouteTable2:
  Type: AWS::EC2::RouteTable
  Properties:
    Vpclid: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ2)
DefaultPrivateRoute2:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway2
PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable2
    SubnetId: !Ref PrivateSubnet2
NoIngressSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: "Sg-no-ingress"
    GroupDescription: "Security group with no ingress rule"
    Vpclid: !Ref VPC
SecurityGroupDB:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupDescription: Allow traffic from Webserver to Database.
    Vpclid: !Ref VPC
    SecurityGroupEgress: []
    SecurityGroupIngress:
      - FromPort: 3306
        ToPort: 3306
        IpProtocol: tcp
        Description: Allow traffic from Webserver to Database
        SourceSecurityGroupId: !Ref SecurityGroupWebServer
    GroupName: Sg-Prod-Db
SecurityGroupLB:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupDescription: A security group that allows inbound web traffic (TCP ports 80 and 443).
    Vpclid: !Ref VPC
    SecurityGroupEgress:
      - FromPort: -1
        ToPort: -1
        IpProtocol: '-1'
        Description: ""
        CidrIp: 0.0.0.0/0
    SecurityGroupIngress:
      - FromPort: 80
        ToPort: 80
        IpProtocol: tcp
        Description: Allow HTTP traffic
        CidrIp: 0.0.0.0/0
      - FromPort: 443
        ToPort: 443

```

```

    IpProtocol: tcp
    Description: Allow HTTPS traffic
    CidrIp: 0.0.0.0/0
    GroupName: Sg-Prod-LB
SecurityGroupSSH:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupDescription: A security group that allows inbound SSH traffic (TCP port 22).
    VpcId: !Ref VPC
    SecurityGroupEgress:
      - FromPort: -1
        ToPort: -1
        IpProtocol: '-1'
        Description: ""
        CidrIp: 0.0.0.0/0
    SecurityGroupIngress:
      - FromPort: 22
        ToPort: 22
        IpProtocol: tcp
        Description: Allow SSH traffic
        CidrIp: 192.168.0.254/32
    GroupName: Sg-Prod-SSH
SecurityGroupWebServer:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupDescription: Build a custom security group.
    VpcId: !Ref VPC
    SecurityGroupEgress: []
    SecurityGroupIngress:
      - FromPort: 443
        ToPort: 443
        IpProtocol: tcp
        Description: Allow traffic from load balancer to web server
        SourceSecurityGroupId: !Ref SecurityGroupLB
    GroupName: Sg-Prod-Webserver
FlowLogs:
  Type: 'AWS::EC2::FlowLog'
  Properties:
    ResourceType: VPC
    ResourceId:
      Ref: VPC
    TrafficType: ALL
    LogDestinationType: cloud-watch-logs
    LogGroupName: FlowLogs
    DeliverLogsPermissionArn:
      'Fn::GetAtt':
        - iamRoleForFlowLogs
        - Arn
FlowLogsGroup:
  Type: 'AWS::Logs::LogGroup'
  Properties:
    LogGroupName: FlowLogs
iamRoleForFlowLogs:
  Type: 'AWS::IAM::Role'
  Properties:
    RoleName: iamRoleFlowLogsToCloudWatchLogs
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Sid: ""
          Effect: Allow
          Principal:
            Service: vpc-flow-logs.amazonaws.com
          Action: 'sts:AssumeRole'
  Policies:
    - PolicyName: allow-access-to-cw-logs
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action:
              - 'logs:CreateLogGroup'

```



```

    - 'logs:CreateLogStream'
    - 'logs:PutLogEvents'
    - 'logs:DescribeLogGroups'
    - 'logs:DescribeLogStreams'
    Resource: '*'
# Public Network ACL
PMPublicNACL:
  Type: AWS::EC2::NetworkAcl
  DependsOn: VPC
  Properties:
    Vpclid: !Ref VPC
    Tags:
      - Key: Name
        Value: !Join [ "", [ !Ref "AWS::StackName", "-public-acl" ] ]
# Private Network ACL
PMPrivateNACL:
  Type: AWS::EC2::NetworkAcl
  DependsOn: VPC
  Properties:
    Vpclid: !Ref VPC
    Tags:
      - Key: Name
        Value: !Join [ "", [ !Ref "AWS::StackName", "-private-acl" ] ]
#####
# Public Network ACL Firewall Protection (inbound and outbound traffic at the subnet level)
### INBOUND HTTP Network ACL RULES #####
InboundHTTPNACL:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PMPublicNACL
    RuleNumber: '100'
    Protocol: "6"
    RuleAction: "allow"
    Egress: "false"
    CidrBlock: "0.0.0.0/0"
    PortRange:
      From: '80'
      To: '80'
### INBOUND HTTPS Network ACL RULES ###
InboundHTTPSNACL:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PMPublicNACL
    RuleNumber: '200'
    Protocol: "6"
    RuleAction: "allow"
    Egress: "false"
    CidrBlock: "0.0.0.0/0"
    PortRange:
      From: '443'
      To: '443'
### INBOUND SSH Network ACL RULES ###
InboundSSHNACL:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PMPublicNACL
    RuleNumber: '300'
    Protocol: "6"
    RuleAction: "allow"
    Egress: "false"
    CidrBlock: !Ref VPNIP
    PortRange:
      From: '22'
      To: '22'
### INBOUND Ephemeral Ports Network ACL RULES ###
InboundEPHNACL2:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PMPublicNACL
    RuleNumber: '400'
    Protocol: "6"
    RuleAction: "allow"

```

```

Egress: "false"
CidrBlock: "0.0.0.0/0"
PortRange:
  From: '1024'
  To: '65535'
### INBOUND Ephemeral Ports Network ACL RULES (UDP) ###
InboundEPHNACL2UDP:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPublicNACL
  RuleNumber: '410'
  Protocol: "17"
  RuleAction: "allow"
  Egress: "false"
  CidrBlock: "0.0.0.0/0"
  PortRange:
    From: '1024'
    To: '65535'
### INBOUND ICMP Network ACL RULES ###
InboundICMPNACL:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPublicNACL
  RuleNumber: '500'
  Protocol: "1"
  RuleAction: "allow"
  Egress: "false"
  CidrBlock: "192.168.0.0/16"
  Icmp:
    Code: "-1"
    Type: "-1"
### OUTBOUND HTTP Network ACL RULES ###
OutboundHTTPNACL:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPublicNACL
  RuleNumber: '100'
  Protocol: "6"
  RuleAction: "allow"
  Egress: "true"
  CidrBlock: "0.0.0.0/0"
  PortRange:
    From: '80'
    To: '80'
### OUTBOUND HTTPS Network ACL RULES ###
OutboundHTTPSACL:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPublicNACL
  RuleNumber: '200'
  Protocol: "6"
  RuleAction: "allow"
  Egress: "true"
  CidrBlock: "0.0.0.0/0"
  PortRange:
    From: '443'
    To: '443'
### OUTBOUND SSH Network ACL RULES ###
OutboundSSHACL:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPublicNACL
  RuleNumber: '300'
  Protocol: "6"
  RuleAction: "allow"
  Egress: "true"
  CidrBlock: !Ref VPNIP
  PortRange:
    From: '22'
    To: '22'
### OUTBOUND Ephemeral Ports Network ACL RULES ###
OutboundEPHNACL:

```

```

Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPublicNACL
  RuleNumber: '400'
  Protocol: "6"
  RuleAction: "allow"
  Egress: "true"
  CidrBlock: "0.0.0.0/0"
  PortRange:
    From: '1024'
    To: '65535'
#### OUTBOUND Ephemeral Ports Network ACL RULES (UDP) ###
OutboundEPHNACLUDP:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPublicNACL
  RuleNumber: '410'
  Protocol: "17"
  RuleAction: "allow"
  Egress: "true"
  CidrBlock: "0.0.0.0/0"
  PortRange:
    From: '1024'
    To: '65535'
### OUTBOUND ICMP Network ACL RULES ###
OutboundICMPNACL:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPublicNACL
  RuleNumber: '500'
  Protocol: "1"
  RuleAction: "allow"
  Egress: "true"
  CidrBlock: "192.168.0.0/16"
  Icmp:
    Code: "-1"
    Type: "-1"
### INBOUND HTTP Network ACL RULES #####
InboundHTTPNACL:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPrivateNACL
  RuleNumber: '100'
  Protocol: "6"
  RuleAction: "allow"
  Egress: "false"
  CidrBlock: "0.0.0.0/0"
  PortRange:
    From: '80'
    To: '80'
### INBOUND HTTPS Network ACL RULES ###
InboundHTTPSACL:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPrivateNACL
  RuleNumber: '200'
  Protocol: "6"
  RuleAction: "allow"
  Egress: "false"
  CidrBlock: "0.0.0.0/0"
  PortRange:
    From: '443'
    To: '443'
### INBOUND SSH Network ACL RULES ###
InboundSSHACL:
Type: "AWS::EC2::NetworkAclEntry"
Properties:
  NetworkAclId: !Ref PMPrivateNACL
  RuleNumber: '300'
  Protocol: "6"
  RuleAction: "allow"
  Egress: "false"

```

```

CidrBlock: !Ref VpcCIDR
PortRange:
  From: '22'
  To: '22'
### INBOUND MYSQL Network ACL RULES ###
InboundMYSQLNACL:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PMPrivateNACL
    RuleNumber: '330'
    Protocol: "6"
    RuleAction: "allow"
    Egress: "false"
    CidrBlock: !Ref VpcCIDR
    PortRange:
      From: '3306'
      To: '3306'
### INBOUND Ephemeral Ports Network ACL RULES ###
InboundEPHNACL:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PMPrivateNACL
    RuleNumber: '400'
    Protocol: "6"
    RuleAction: "allow"
    Egress: "false"
    CidrBlock: "0.0.0.0/0"
    PortRange:
      From: '1024'
      To: '65535'
### INBOUND Ephemeral Ports Network ACL RULES (UDP) ###
InboundEPHNACLPU:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PMPrivateNACL
    RuleNumber: '410'
    Protocol: "17"
    RuleAction: "allow"
    Egress: "false"
    CidrBlock: "0.0.0.0/0"
    PortRange:
      From: '1024'
      To: '65535'
### INBOUND ICMP Network ACL RULES ###
InboundICMPNACL:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PMPrivateNACL
    RuleNumber: '500'
    Protocol: "1"
    RuleAction: "allow"
    Egress: "false"
    CidrBlock: "0.0.0.0/0"
    Icmp:
      Code: "-1"
      Type: "-1"
### OUTBOUND HTTP Network ACL RULES ###
OutboundHTTPNACL:
  Type: "AWS::EC2::NetworkAclEntry"
  Properties:
    NetworkAclId: !Ref PMPrivateNACL
    RuleNumber: '100'
    Protocol: "6"
    RuleAction: "allow"
    Egress: "true"
    CidrBlock: "0.0.0.0/0"
    PortRange:
      From: '80'
      To: '80'
### OUTBOUND HTTPS Network ACL RULES ###
OutboundHTTPSACL:
  Type: "AWS::EC2::NetworkAclEntry"

```

```

Properties:
  NetworkAcId: !Ref PMPrivateNACL
  RuleNumber: '200'
  Protocol: "6"
  RuleAction: "allow"
  Egress: "true"
  CidrBlock: "0.0.0.0/0"
  PortRange:
    From: '443'
    To: '443'
### OUTBOUND SSH Network ACL RULES ###
OutboundSSHACL:
  Type: "AWS::EC2::NetworkACLEntry"
  Properties:
    NetworkAcId: !Ref PMPrivateNACL
    RuleNumber: '300'
    Protocol: "6"
    RuleAction: "allow"
    Egress: "true"
    CidrBlock: !Ref VpcCIDR
    PortRange:
      From: '22'
      To: '22'
### OUTBOUND MYSQL Network ACL RULES ###
OutboundMYSQLACL:
  Type: "AWS::EC2::NetworkACLEntry"
  Properties:
    NetworkAcId: !Ref PMPrivateNACL
    RuleNumber: '330'
    Protocol: "6"
    RuleAction: "allow"
    Egress: "true"
    CidrBlock: !Ref VpcCIDR
    PortRange:
      From: '3306'
      To: '3306'
### OUTBOUND Ephemeral Ports Network ACL RULES ###
OutboundEPHACL:
  Type: "AWS::EC2::NetworkACLEntry"
  Properties:
    NetworkAcId: !Ref PMPrivateNACL
    RuleNumber: '400'
    Protocol: "6"
    RuleAction: "allow"
    Egress: "true"
    CidrBlock: "0.0.0.0/0"
    PortRange:
      From: '1024'
      To: '65535'
### OUTBOUND Ephemeral Ports Network ACL RULES (UDP) ###
OutboundEPHACLPU:
  Type: "AWS::EC2::NetworkACLEntry"
  Properties:
    NetworkAcId: !Ref PMPrivateNACL
    RuleNumber: '410'
    Protocol: "17"
    RuleAction: "allow"
    Egress: "true"
    CidrBlock: "0.0.0.0/0"
    PortRange:
      From: '1024'
      To: '65535'
### OUTBOUND ICMP Network ACL RULES ###
OutboundICMPACL:
  Type: "AWS::EC2::NetworkACLEntry"
  Properties:
    NetworkAcId: !Ref PMPrivateNACL
    RuleNumber: '500'
    Protocol: "1"
    RuleAction: "allow"
    Egress: "true"
    CidrBlock: "0.0.0.0/0"

```

```

Icmp:
  Code: "-1"
  Type: "-1"
PublicSubnet1AssociationNacl:
  Type: AWS::EC2::SubnetNetworkAclAssociation
  Properties:
    SubnetId: !Ref PublicSubnet1
    NetworkAclId:
      Ref: PMPublicNACL
PublicSubnet2AssociationNacl:
  Type: AWS::EC2::SubnetNetworkAclAssociation
  Properties:
    SubnetId: !Ref PublicSubnet2
    NetworkAclId:
      Ref: PMPublicNACL
PrivateSubnet1AssociationNacl:
  Type: AWS::EC2::SubnetNetworkAclAssociation
  Properties:
    SubnetId: !Ref PrivateSubnet1
    NetworkAclId:
      Ref: PMPrivateNACL
PrivateSubnet2AssociationNacl:
  Type: AWS::EC2::SubnetNetworkAclAssociation
  Properties:
    SubnetId: !Ref PrivateSubnet2
    NetworkAclId:
      Ref: PMPrivateNACL
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC
  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ] ]
  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ] ]
  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1
  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2
  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1
  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2
  NoIngressSecurityGroup:
    Description: Security group with no ingress rule
    Value: !Ref NoIngressSecurityGroup
  SecurityGroupDB:
    Description: Security group for db
    Value: !Ref SecurityGroupDB
  SecurityGroupWebserver:
    Description: Security group for Webserver
    Value: !Ref SecurityGroupWebServer
  SecurityGroupSSH:
    Description: Security group for SSH traffic
    Value: !Ref SecurityGroupSSH
  SecurityGroupLB:
    Description: Security group for public load balancer
    Value: !Ref SecurityGroupLB

```

Appendix 2: Enforcing MFA Authentication

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewAccountInfo",

```

```

"Effect": "Allow",
"Action": [
  "iam:GetAccountPasswordPolicy",
  "iam:GetAccountSummary",
  "iam:ListVirtualMFADevices",
  "iam:ListUsers"
],
"Resource": "*"
},
{
  "Sid": "AllowManageOwnPasswords",
  "Effect": "Allow",
  "Action": [
    "iam:ChangePassword",
    "iam:GetUser",
    "iam:CreateLoginProfile",
    "iam>DeleteLoginProfile",
    "iam:GetLoginProfile",
    "iam:UpdateLoginProfile"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
  "Sid": "AllowManageOwnAccessKeys",
  "Effect": "Allow",
  "Action": [
    "iam:CreateAccessKey",
    "iam>DeleteAccessKey",
    "iam:ListAccessKeys",
    "iam:UpdateAccessKey"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
  "Sid": "AllowManageOwnSigningCertificates",
  "Effect": "Allow",
  "Action": [
    "iam>DeleteSigningCertificate",
    "iam:ListSigningCertificates",
    "iam:UpdateSigningCertificate",
    "iam:UploadSigningCertificate"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
  "Sid": "AllowManageOwnSSHPublicKeys",
  "Effect": "Allow",
  "Action": [
    "iam>DeleteSSHPublicKey",
    "iam:GetSSHPublicKey",
    "iam:ListSSHPublicKeys",
    "iam:UpdateSSHPublicKey",
    "iam:UploadSSHPublicKey"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
  "Sid": "AllowManageOwnGitCredentials",
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceSpecificCredential",
    "iam>DeleteServiceSpecificCredential",
    "iam:ListServiceSpecificCredentials",
    "iam:ResetServiceSpecificCredential",
    "iam:UpdateServiceSpecificCredential"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
  "Sid": "AllowManageOwnVirtualMFADevice",
  "Effect": "Allow",
  "Action": [

```

```

        "iam:CreateVirtualMFADevice",
        "iam>DeleteVirtualMFADevice"
    ],
    "Resource": "arn:aws:iam::*:mfa/${aws:username}"
},
{
    "Sid": "AllowManageOwnUserMFA",
    "Effect": "Allow",
    "Action": [
        "iam:DeactivateMFADevice",
        "iam:EnableMFADevice",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
    "Sid": "DenyAllExceptListedIfNoMFA",
    "Effect": "Deny",
    "NotAction": [
        "iam:CreateVirtualMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ListVirtualMFADevices",
        "iam:ResyncMFADevice",
        "sts:GetSessionToken",
        "iam:ListUsers",
        "iam:CreateLoginProfile",
        "iam:ChangePassword"
    ],
    "Resource": "*",
    "Condition": {
        "BoolIfExists": {
            "aws:MultiFactorAuthPresent": "false"
        }
    }
}
]
}

```

Appendix 3: Enforcing Encryption

AWSTemplateFormatVersion: '2010-09-09'

Description: 'Enforcing encryption for S3 data objects'

Resources:

ScpPolicy:

Type: 'Custom::ServiceControlPolicy'

Properties:

PolicyName: scp_s3_encryption

PolicyDescription: >-

This SCP requires that all Amazon S3 buckets use AES256 encryption in an AWS Account.

PolicyContents: >-

```
{
  "Version": "2012-10-
```

```
17",
  "Statement": [
    {
      "Action": ["s3:PutObject"],
      "Resource": "*",
      "Effect": "Deny",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      }
    },
    {
      "Action": ["s3:PutObject"],
      "Resource": "*",
      "Effect": "Deny",
      "Condition": {
        "Bool": {
          "s3:x-amz-server-side-encryption": false
        }
      }
    }
  ]
}
```

ServiceToken:

'Fn::GetAtt':

- ScpResourceLambda

- Arn

ScpResourceLambdaRole:

Type: 'AWS::IAM::Role'

Properties:

AssumeRolePolicyDocument:

Version: '2012-10-17'

Statement:

- Effect: Allow

Principal:

Service: lambda.amazonaws.com

Action:

- 'sts:AssumeRole'


```

Path: /
ManagedPolicyArns:
- 'arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
Policies:
- PolicyName: scp-access
  PolicyDocument:
    Statement:
    - Effect: Allow
      Action:
      - 'organizations:UpdatePolicy'
      - 'organizations:DeletePolicy'
      - 'organizations:CreatePolicy'
      - 'organizations:ListPolicies'
      Resource: '*'
ScpResourceLambda:
Type: 'AWS::Lambda::Function'
Properties:
Code:
ZipFile: |-

'use strict';
const AWS = require('aws-sdk');
const response = require('cfn-response');
const organizations = new AWS.Organizations({region: 'us-east-1'});

exports.handler = (event, context, cb) => {
  console.log('Invoke:', JSON.stringify(event));
  const done = (err, data) => {
    if (err) {
      console.log('Error: ', err);
      response.send(event, context, response.FAILED, {}, 'CustomResourcePhysicalID');
    } else {
      response.send(event, context, response.SUCCESS, {}, 'CustomResourcePhysicalID');
    }
  };

  const updatePolicies = (policyName, policyAction) => {
    organizations.listPolicies({
      Filter: "SERVICE_CONTROL_POLICY"
    }, function(err, data){
      if (err) done(err);
      else {
        const policy = data.Policies.filter((policy) => (policy.Name === policyName))
        let policyId = ""
        if (policy.length > 0)
          policyId = policy[0].Id
        else
          done('policy not found')
        if (policyAction === 'Update'){
          organizations.updatePolicy({
            Content: event.ResourceProperties.PolicyContents,
            PolicyId: policyId
          }, done)
        }
        else {
          organizations.deletePolicy({
            PolicyId: policyId
          }, done)
        }
      }
    })
  }

  if (event.RequestType === 'Update' || event.RequestType === 'Delete') {
    updatePolicies(event.ResourceProperties.PolicyName, event.RequestType)
  }
  else if (event.RequestType === 'Create') {
    organizations.createPolicy({
      Content: event.ResourceProperties.PolicyContents,
      Description: event.ResourceProperties.PolicyDescription,
      Name: event.ResourceProperties.PolicyName,
      Type: "SERVICE_CONTROL_POLICY"
    })
  }
}

```

```

    }, done);
  } else {
    cb(new Error('unsupported RequestType: ', event.RequestType));
  }
};
Handler: index.handler
MemorySize: 128
Role:
  Fn::GetAtt:
  - ScpResourceLambdaRole
  - Arn
Runtime: nodejs12.x
Timeout: 120

```

Appendix 4: Governing Data Encryption

AWSTemplateFormatVersion: 2010-09-09

Description: Configure automatic governance for data encryption

Parameters:

ConfigRuleName:

Type: 'String'

Description: 'Checks whether EBS volumes that are in an attached state are encrypted.'

Default: EBS_VOL_ENCR_CHK

ConfigRuleName1:

Type: 'String'

Description: 'Checks whether storage encryption is enabled for your RDS DB instances.'

Default: RDS_DB_ENCR_CHK

ConfigRuleName2:

Type: 'String'

Description: 'Checks whether S3 storage encryption is enabled.'

Default: S3_ENCR_CHK

ManagedResourcePrefix:

Type: 'String'

Description: 'Prefix for the managed resources'

AllSupported:

Type: String

Default: 'true'

Description: Indicates whether to record all supported resource types.

AllowedValues:

- 'true'

- 'false'

IncludeGlobalResourceTypes:

Type: String

Default: 'true'

Description: Indicates whether AWS Config records all supported global resource types.

AllowedValues:

- 'true'

- 'false'

ResourceTypes:

Type: CommaDelimitedList

Description: A list of valid AWS resource types to include in this recording group.

Frequency:

Type: String

Default: 1hour

Description: The frequency with which AWS Config delivers configuration snapshots.

AllConfigTopicName:

Type: String

Default: ''

Description: All Configuration Notification SNS Topic.

SecurityAccountId:

Type: 'String'

Description: AWS Account Id of the Security account.

AuditBucketName:

Type: String

Default: ''

Description: Audit Bucket name from the Log Archive Account

AWSLogsS3KeyPrefix:

Type: 'String'

Description: 'Organization ID to use as the S3 Key prefix for storing the audit logs'

Conditions:

IsAllSupported: !Equals
 - !Ref AllSupported
 - 'true'

Mappings:

Settings:

FrequencyMap:
 1hour : One_Hour
 3hours : Three_Hours
 6hours : Six_Hours
 12hours : Twelve_Hours
 24hours : TwentyFour_Hours

Resources:

CheckForEncryptedVolumes:

Type: AWS::Config::ConfigRule
 Properties:
 ConfigRuleName: !Sub \${ConfigRuleName}
 Description: Checks whether EBS volumes that are in an attached state are encrypted.
 Source:
 Owner: AWS
 SourceIdentifier: ENCRYPTED_VOLUMES
 Scope:
 ComplianceResourceTypes:
 - AWS::EC2::Volume

CheckForRdsStorageEncryption:

Type: AWS::Config::ConfigRule
 Properties:
 ConfigRuleName: !Sub \${ConfigRuleName1}
 Description: Checks whether storage encryption is enabled for your RDS DB instances.
 Source:
 Owner: AWS
 SourceIdentifier: RDS_STORAGE_ENCRYPTED
 Scope:
 ComplianceResourceTypes:
 - AWS::RDS::DBInstance

CheckForS3StorageEncryption:

Type: AWS::Config::ConfigRule
 Properties:
 ConfigRuleName: !Sub \${ConfigRuleName2}
 Description: Checks whether S3 storage encryption is enabled.
 Source:
 Owner: AWS
 SourceIdentifier: S3_BUCKET_SERVER_SIDE_ENCRYPTION_ENABLED
 Scope:
 ComplianceResourceTypes:
 - AWS::S3::Bucket

ConfigRecorder:

Type: AWS::Config::ConfigurationRecorder
 Properties:
 Name: !Sub \${ManagedResourcePrefix}-BaselineConfigRecorder
 RoleARN: !Sub arn:aws:iam::\${AWS::AccountId}:role/\${ManagedResourcePrefix}-ConfigRecorderRole
 RecordingGroup:
 AllSupported: !Ref AllSupported
 IncludeGlobalResourceTypes: !Ref IncludeGlobalResourceTypes
 ResourceTypes: !If
 - IsAllSupported
 - !Ref AWS::NoValue
 - !Ref ResourceTypes

ConfigDeliveryChannel:

Type: AWS::Config::DeliveryChannel
 Properties:
 Name: !Sub \${ManagedResourcePrefix}-BaselineConfigDeliveryChannel
 ConfigSnapshotDeliveryProperties:
 DeliveryFrequency: !FindInMap
 - Settings
 - FrequencyMap
 - !Ref Frequency
 S3BucketName: !Ref AuditBucketName
 S3KeyPrefix: !Ref AWSLogsS3KeyPrefix
 SnsTopicARN: !Sub arn:aws:sns:\${AWS::Region}:\${SecurityAccountId}:\${AllConfigTopicName}

AuthorizerDub:
 Type: "AWS::Config::AggregationAuthorization"
 Properties:
 AuthorizedAccountId: !Ref SecurityAccountId
 AuthorizedAwsRegion: eu-west-1

Outputs:
 BaselineConfigRecorder:
 Description: Baseline Config Recorder
 Value: !Ref ConfigRecorder
 BaselineConfigDeliveryChannel:
 Description: Baseline Config Delivery Channel
 Value: !Ref ConfigDeliveryChannel

Appendix 5: Incident Response

AWSTemplateFormatVersion: 2010-09-09

Description: Incident response stack

Resources:

CloudWatchAlarmSSH:
 Type: 'AWS::CloudWatch::Alarm'
 Properties:
 AlarmName: cwalarm_rejected_ssh
 AlarmDescription: >-
 A CloudWatch Alarm that triggers when there are rejected SSH connections in a VPC (Default: 10 connections per hour). Requires VPC flow logs to be enabled.
 MetricName: RejectedSSHCount
 Namespace: VPCFlowLogsMetrics
 Statistic: Sum
 Period: '3600'
 EvaluationPeriods: '1'
 Threshold: '10'
 ComparisonOperator: GreaterThanOrEqualToThreshold
 AlarmActions:
 - 'arn:aws:sns:eu-west-1:645147117463:aws-controltower-SecurityNotifications'
 TreatMissingData: notBreaching

MetricFilterSSH:
 Type: 'AWS::Logs::MetricFilter'
 Properties:
 LogGroupName: aws-controltower/CloudTrailLogs
 FilterPattern: >-
 [version, account, eni, source, destination, srcport, destport="22", protocol="6", packets, bytes, windowstart, windowend, action="REJECT", flowlogstatus]
 MetricTransformations:
 - MetricValue: '1'
 MetricNamespace: VPCFlowLogsMetrics
 MetricName: RejectedSSHCount

CloudWatchAlarmVPN:
 Type: 'AWS::CloudWatch::Alarm'
 Properties:
 AlarmName: vpnstate_alarm
 AlarmDescription: >-
 A CloudWatch Alarm that triggers when the state of both VPN tunnels in an AWS VPN connection are down.
 MetricName: TunnelState
 Namespace: AWS/VPN
 Statistic: Maximum
 Period: '300'
 EvaluationPeriods: '1'
 Threshold: '0'
 ComparisonOperator: LessThanOrEqualToThreshold
 AlarmActions:
 - 'arn:aws:sns:eu-west-1:645147117463:aws-controltower-SecurityNotifications'
 Dimensions:
 - Name: VpnId
 Value: vpn-1122334455aabbccd

CloudWatchAlarmBilling:
 Type: 'AWS::CloudWatch::Alarm'
 Properties:
 AlarmName: billing_alarm
 AlarmDescription: >-
 A CloudWatch Alarm that triggers the AWS bill reaches the specified threshold (default: 100 USD).

MetricName: EstimatedCharges
 Namespace: AWS/Billing
 Statistic: Maximum
 Period: '21600'
 EvaluationPeriods: '1'
 Threshold: '100'
 ComparisonOperator: GreaterThanOrEqualToThreshold
 AlarmActions:
 - 'arn:aws:sns:eu-west-1:645147117463:aws-controltower-SecurityNotifications'
 Dimensions:
 - Name: Currency
 Value: USD

CloudWatchAlarmLogins:
 Type: 'AWS::CloudWatch::Alarm'
 Properties:
 AlarmName: failed_console_logins
 AlarmDescription: >-
 A CloudWatch Alarm that triggers if there are AWS Management Console authentication failures.
 MetricName: ConsoleLoginFailures
 Namespace: CloudTrailMetrics
 Statistic: Sum
 Period: '300'
 EvaluationPeriods: '1'
 Threshold: '1'
 ComparisonOperator: GreaterThanOrEqualToThreshold
 AlarmActions:
 - 'arn:aws:sns:eu-west-1:645147117463:aws-controltower-SecurityNotifications'
 TreatMissingData: notBreaching

MetricFilterLogins:
 Type: 'AWS::Logs::MetricFilter'
 Properties:
 LogGroupName: aws-controltower/CloudTrailLogs
 FilterPattern: >-
 { (\$.eventName = ConsoleLogin) && (\$.errorMessage = "Failed authentication") }
 MetricTransformations:
 - MetricValue: '1'
 MetricNamespace: CloudTrailMetrics
 MetricName: ConsoleLoginFailures

CloudWatchAlarmRootlogin:
 Type: 'AWS::CloudWatch::Alarm'
 Properties:
 AlarmName: root_account_login
 AlarmDescription: A CloudWatch Alarm that triggers if a root user uses the account.
 MetricName: RootUserEventCount
 Namespace: CloudTrailMetrics
 Statistic: Sum
 Period: '60'
 EvaluationPeriods: '1'
 Threshold: '1'
 ComparisonOperator: GreaterThanOrEqualToThreshold
 AlarmActions:
 - 'arn:aws:sns:eu-west-1:645147117463:aws-controltower-SecurityNotifications'
 TreatMissingData: notBreaching

MetricFilterRootlogin:
 Type: 'AWS::Logs::MetricFilter'
 Properties:
 LogGroupName: aws-controltower/CloudTrailLogs
 FilterPattern: >-
 { (\$.userIdentity.type = "Root") && (\$.userIdentity.invokedBy NOT EXISTS) && (\$.eventType != "AwsServiceEvent") }
 MetricTransformations:
 - MetricValue: '1'
 MetricNamespace: CloudTrailMetrics
 MetricName: RootUserEventCount

CloudWatchAlarmAudittrail:
 Type: 'AWS::CloudWatch::Alarm'
 Properties:
 AlarmName: cloudtrail_changes
 AlarmDescription: A CloudWatch Alarm that triggers when changes are made to CloudTrail.
 MetricName: CloudTrailEventCount

Namespace: CloudTrailMetrics
Statistic: Sum
Period: '300'
EvaluationPeriods: '1'
Threshold: '1'
ComparisonOperator: GreaterThanOrEqualToThreshold
AlarmActions:
- 'arn:aws:sns:eu-west-1:645147117463:aws-controltower-SecurityNotifications'
TreatMissingData: notBreaching
MetricFilterAudittrail:
Type: 'AWS::Logs::MetricFilter'
Properties:
LogGroupName: aws-controltower/CloudTrailLogs
FilterPattern: >-
{ (\$.eventName = CreateTrail) || (\$.eventName = UpdateTrail) ||
(\$.eventName = DeleteTrail) || (\$.eventName = StartLogging) ||
(\$.eventName = StopLogging) }
MetricTransformations:
- MetricValue: '1'
MetricNamespace: CloudTrailMetrics
MetricName: CloudTrailEventCount