Continuous Compliance Automation in AWS cloud environment

Isak Jansson 42605 / 1800439

Master of Science (Technology), Master's thesis in computer engineering

Faculty of Science and Engineering, Information Technologies

Åbo Akademi

21.05.2021

Supervisor: Marina Waldén

Abstract

Compliance is increasingly growing as an area of importance in the Information Technology sector to compete and deliver applications that follow regulatory requirements and standards. The scrutiny of auditors and regulators in some parts of the Information Technology sector has increased significantly during the last years due to the increasingly hostile environment. Compliance has become required by customers and needed if engaging in specific activities in the Information Technology sector, especially when working in the financial industry.

This thesis investigates compliance for Information Technology systems in the financial sector. More specifically, the thesis examines how to remain in compliance using the methodology of continuous compliance. Achieving compliance can be challenging, especially when moving Information Technology infrastructure to a public cloud service provider from a strictly on-premises solution. However, by introducing automation into the process, this thesis tries to show how compliance work can be decreased with the possibilities of automation in the cloud. Both to improve the compliance posture as well as security and minimize the involvement of human beings in the ever-changing compliance process. Automating the compliance process is done using different services provided by Amazon Web Services and introducing other tools even to remediate compliance problems wherever possible automatically. This thesis shows that it may be beneficial to introduce an automated continuous compliance process when working with strict regulation to help with reoccurring issues. The implemented solution focuses on compliance frameworks like CIS benchmark and PCI-DSS requirements in relation to Information Technology infrastructure. The proof of concept implemented focuses on the advantage and possibilities to automate compliance work in Amazon Web Services cloud environment as well as investigates the possible use of the idea in a full-scale solution.

Keywords: Continuous Compliance, Compliance, Cloud, AWS, Automation

Preface and Acknowledgements

This Master's thesis was written for Crosskey Banking Solution. The work has been fascinating and inspiring. I especially want to thank my supervisor, Jonathan Klingberg at Crosskey, for making this thesis possible and for helping me in the process. I also want to thank the AWS team for the help and guidance provided. Thanks to my team at Capital Markets for the support and for making this possible by letting me shortly leave the team to complete my studies. I also would like to thank the Crosskey Security Department for the guidance regarding compliance processes and security. I also would like to thank my supervisor Marina Waldén at Åbo Akademi University for helping me and providing me with valuable ideas for improving the thesis as well as good guidance through the process. Last but certainly not least, I would like to thank everyone at Datateknologerna vid Åbo Akademi r.f for the support and the good times during my Master's studies.

Mariehamn 21.05.2021

Isak Jansson

Acronyms

IT - Information Technology

CSE - Continuous Software Engineering

CI - Continuous Integration

CD - Continuous Delivery

AWS - Amazon Web Services

CIS - Center for Internet Security

PCI DSS - Payment Card Industry Data Security Standard

SDK - Software Development Kit

Table of Contents

List of Figures

List of Tables

1. Introduction

Following and implementing compliance frameworks has become an everyday task for many companies in many sectors of Information Technology (IT), especially in the financial sector. The significance of following these compliance frameworks is not only to follow regulations but also to be able to engage in specific business sectors that require compliance with international standards and regulations. The need to enforce compliance requirements and the need to prove that these requirements are fulfilled is a constant task for IT service providers affected by any compliance framework. [1]

1.1. Purpose of the thesis

Customers of IT service providers are increasingly demanding better quality and safety from the purchased applications. Requirements of following specific compliance frameworks are often wished for and, in some industries, mandatory for conducting operations in the market. One such industry is the financial sector, with tight regulations and many compliance frameworks required for operation. The tight regulations are not only to make the applications more secure but also to ensure the quality of the services provided to the end customer. To fulfil customer expectations, more compliance frameworks that try to promote security are introduced and enforced, often leading to increased overhead for proving that the systems and the company itself are compliant with the customer's compliance requirements.

In some cases, introducing manual verification in different processes to ensure compliance of the systems. This especially if new systems or a new compliance framework needs to be implemented. Costs can often be minimized by moving applications hosted previously on on-premises servers to a public cloud provider. But this introduces both new processes and new systems as well as the complexity of proving compliance. The previously set routines and processes for proving compliance are impacted and the workload often increases. However, by introducing automated continuous compliance tools to do some of the workloads, the impact on the compliance work can be reduced and only the required manual tasks increase. The thesis aims to introduce automated continuous compliance tools and processes for remediation in AWS to reduce the compliance work of moving some of the

infrastructure to a public cloud provider. The thesis focuses on compliance and security from an infrastructure point of view and only includes minor insights into application compliance. Introducing cloud solutions to an already existing compliance process often introduces new complexity, but by using automation, some of the complexity can be reduced over time. The constant changes to infrastructure in the public cloud environment will complicate the implementation of previously used compliance processes. Therefore, it is crucial to focus on implementing automatic processes for handling infrastructure changes that lead to compliance changes. The application compliance processes do not suffer to the same degree from the move to the cloud and are only touched upon in this thesis.

1.2. Thesis structure

The thesis is divided into chapters that range from theory to practical work. Chapter 2 describes the background and theory behind compliance, security, and continuous compliance as well as Continuous Software Engineering. Furthermore, a brief description of what cloud solutions are and how compliance is connected to the financial sector are presented. In Chapter 3, different compliance frameworks, services and tools used are shortly explained to give an understanding of what they are. The next Chapter 4 introduces the already existing research in cloud solutions and compliance as well as a couple of previous continuous compliance solutions available. Chapter 5 discusses and presents the available solutions for compliance in AWS (Amazon Web Services) cloud environments and how different compliance frameworks are considered in the different AWS services. Here also mapping between different compliance frameworks is presented. Chapter 6 the proof of concept that continuous automated compliance is presented, starting with a brief description of the architecture followed by infrastructure implementations. Compliance automation in code is then presented and remediation of different compliance events is explained. Different alternatives to design and possible improvements are also discussed. Chapter 7 gives a short evaluation of the proof of concept, discussing problems and possibilities for use in the future. The final Chapter 8 presents some concluding remarks about the work presented.

2. Background

2.1. Compliance and Security Compliance

In today's IT world, where the increasing complexity is becoming the norm, securing the systems is becoming increasingly difficult. Therefore, different customers and vendors have mandated the suppliers to follow a set of rules that often make the IT systems more likely to be safe, often called Security Compliance. "Security is the state of being safe from threats. By contrast, *security compliance* means conformance with a given set of security requirements" [1]. The security requirements can be set by any given entity like regulators, governments, or groups of experts. Compliance can also be defined as complying with a wish or a demand from, for example, a customer. To fulfil the delivery requirements, a company needs to comply with the regulatory requirements and comply with customer requirements.

2.2. Momentary Assessment to Continuous Compliance

In compliance work, the concept of Momentary Assessment is often the method of choice. The system or application assessed is observed at a specific time. A user often requests a report or compiles a report manually from data available for that exact moment. In this case, the compliance evaluation is based on manual labor or manual triggering a system to produce a compliance report that can be shown to the assessor. The process is illustrated in Figure 1. We can also call this Momentary Compliance because of the nature of the reported data. The report only contains the compliance state of the system at a specific moment. [2]

Figure 1 Momentary Assessment

If the Momentary Assessments are in some ways automated to the degree of being able to assess the Compliance state of the system at least periodically, we can call it Continuous Assessment. The user does not request an assessment of the system state, and instead the system tries to show that it is compliant at least with periodical intervals, the process shown in Figure 2. In the perfect world, the checks should not only be periodical, but event driven for continuously detecting change in the system and compiling an always up to date picture of the compliance state of the system. This can be called Continuous Compliance and the process is shown in Figure 3. But in some cases, Continuous Compliance is not feasible and periodical checks are the only solution. With large systems the process of compiling a report or gathering data for all compliance requirements can be hard, this is due to high workloads and the frequency of change. If the resources monitored are small in numbers, a periodical report can be sufficient for getting an overview of the compliance state of the system. Therefore, when the resources monitored are large in numbers and change occurs often the preferred way to achieve compliance is event driven compliance monitoring or, even better, implementing some kind of Continuous Compliance process. [2] [3] [4]

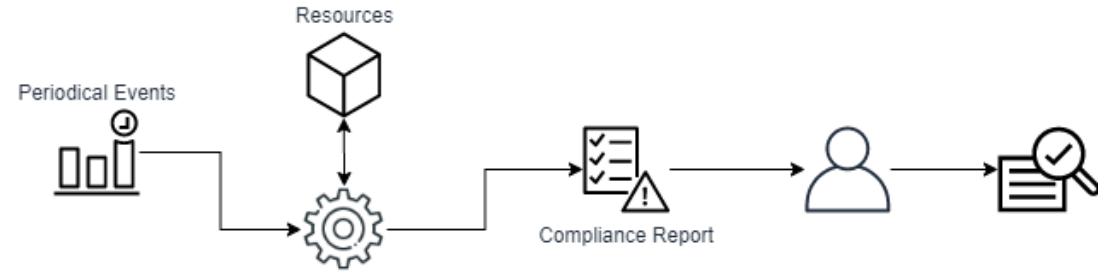


Figure 2 Continuous Assessment

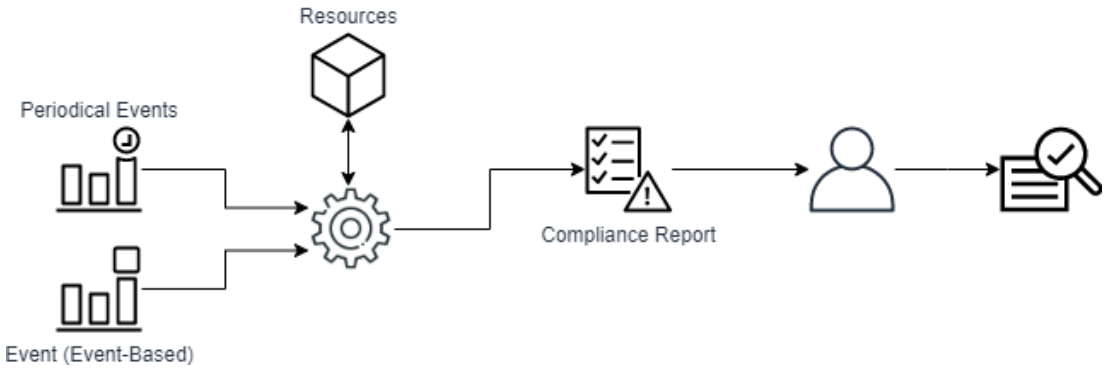


Figure 3 Continuous Compliance

2.3. Continuous Software Engineering CSE

Continuous Software Engineering (CSE) is a debated topic in academic work and the definition of continuous software has many different approaches. However, the importance of solutions for rapid software development strategies is becoming increasingly important. CSE contains many practices used to describe processes that increase automation and the continuous development of applications. Some of the principles build on the agile software development practices introduced in the agile manifesto, but many advocates that the tools used have greater importance than the actual individuals and interactions focused on in the agile manifesto [5].

CSE's most commonly used practices include Continuous Integration (CI) and Continuous Delivery (CD). The constant demand for decreasing the time between release cycles, fixing bugs faster, and increasing changes to the code base requires automation of the processes to deliver a functioning application. To get short release cycles and closer to achieving CD the concept of CI needs to be achieved. CI means that as soon as a developer introduces code and uploads it to the version control repositories, an automated process compiles the code, runs tests, and deploy applications to staging environments. CI does not necessarily mean that the code is deployed into production when the process is ready. In many cases, the process needs some kind of human intervention when necessary. However, the latest version of the application can be deployed to production environments automatically if the process does not run into any problems. A fully automated release pipeline that has no human intervention can also be defined as Continuous Deployment. [6] [7]

These CSE concepts together are often referenced as Code Automation in day-to-day conversations and often indicates that a CI process and some form of CD are used in the development process of the applications or infrastructure.

Releasing early and often will also decrease the effort of the teams needing to switch to operational tasks, leaving more time for development and improvements. To improve the development process, it is essential to detect problems early and correct them immediately. It is also essential to automate the correction if possible and if the problem ever occurs again the automation should take care of the task.

To get a completely automated process is almost unachievable in large companies with already established processes. But, automating as much as possible when it comes to development lifecycle and infrastructure maintenance is essential for delivering better-performing applications. Leading to more stable application updates and better performing operations of the applications. The goal is to minimize the downtime of the customer applications and minimize loss in revenue or customer satisfaction. But achieving full automation is not always possible. However, automating as much as possible is the ultimate goal. [8]

2.3.1. Infrastructure as Code and Automation

The power of using code for infrastructure is that the CSE concept can be applied even to the infrastructure, not only for the application layer. When moving towards cloud environments, the possibility to have infrastructure written in code is appealing. To be able to automatically deploy and version-control infrastructure code means that the infrastructure can also respond faster to the constant changes needed in delivering an application. Enabling automation for the infrastructure where the applications are running enables the application layer to be more flexible to change and customer needs. Therefore, automation is the key.

Infrastructure is not always easy to get into code and there will always be some manual work, often regarding static networking or hardware limitations. However, everything that can be changed using infrastructure as code should also be automated. To at least a degree, the whole infrastructure can be deployed again in case of misconfiguration. This is often achievable in cloud environments and can be implemented in some form in a hybrid cloud solution with consideration of some hardware limitations. [9]

2.4. CSE in relation to Continuous Compliance

When implementing CSE workflows, the security aspect of the development process is often forgotten. Before releasing an application to the public, the security team comes in and assesses the solution, demanding fixes for critical security vulnerabilities. The nature of only checking for vulnerabilities at the end of the development process often leads to missing some critical parts of the system, which then causes a serious security hazard. The same can be said about compliance. Often

the compliance checks are considered in the later part of the development process and fixed after the solution is ready. Therefore, the importance of automating the security controls as well as the compliance controls during development is vital for keeping the good health of the applications when they are released. [7]

Continuous Compliance Monitoring (CCM) can be directly correlated to Continuous Assessment and focuses on always having an overview of the compliance state of the system. Either with periodical checks or event driven monitoring of the compliance state of the system. With a CCM tool detecting compliance issues within your systems is easy and actions to solve these issues can be taken quickly as the development of the applications proceed. As well as be well informed about the risks the development introduces. It is not only important to have a CCM tool for detecting problems. However, the tool should also support the concept of CSE; the solution should be implemented using the same development approach as the rest of the applications. Always having infrastructure and code that supports finding compliance problems that will improve the state of the system. [10]

By implementing infrastructure that supports CCM, the system should now be able to detect compliance and security problems and by using CSE, the system should be ready for Continuous Deployment at any time. However, one automation step could still be introduced. The Continuous Assessments results of CCM findings in the systems still need human intervention to be resolved. Automating these kinds of findings is not easy in on-premises only solution and taking action on findings can be time consuming and repetitive. However, by moving to a cloud solution or a hybrid-cloud solution, the possibilities to automate the actions taken on the compliance or security findings become possible. The automated actions can, if possible, remediate the finding, mitigate the risk of becoming non-compliant or at least notify about the finding, so that correct actions can be taken if human intervention is needed. This allows the automation of repetitive tasks that can be remediated or mitigated through code. Only findings that need human intervention or findings that have not yet been automated are sent to manual processes for further investigation. This is what the thesis will call Continuous Compliance Automation.

2.5. Continuous Compliance Automation

Compliance in today's IT climate is a simple compliance report with checkboxes that indicates the compliance state of the system at a given time. However, this is not always enough to prove compliance. A Compliance Assessment is not considered complete when the assessment is done, and the systems are compliant until the next auditing. Compliance needs to be interpreted as a continuous process where the resources and conditions constantly change. [7]

To be continuously compliant with any framework, there must be some kind of automation. Both collecting data for the compliance assessment as well as continuously remediate the compliance problems. Using CCM tools to monitor your compliance state is the first step to getting an overview of the environment. However, the CCM tool should continuously check for changes at a periodical rate but preferably on an event basis. The events should be forwarded to an automation process that can Remediate the compliance problems and at least involve human beings if it is necessary. Remediating compliance problems can be a daunting task. However, by introducing automation to do so, the reoccurring problems can be fixed, and the time can be focused on the more demanding tasks that require human intervention and cannot be automated in any way.

Continuous Compliance Automation should be part of every step of delivering an application, not only looking at security when a product is ready for production but also during the whole development. This means that even in the development and staging process, compliance problems should be fixed and detected automatically, continuously improving the compliance state of the environment. If the development process is focused on CSE concepts also the security and compliance aspects should be automated as much as possible. Every finding that can improve the security or compliance state of the system should be automated if possible. Here Continuous Compliance Automation can be a way to minimize the need for human intervention and as long as the non-compliant or non-secure state can be fixed to some degree by code. The Continuous Compliance Automation solution should be implemented using the same approach as the rest of the applications, meaning using CSE development strategies. This can often be done quickly using different cloud solution services in a combination of already established techniques.

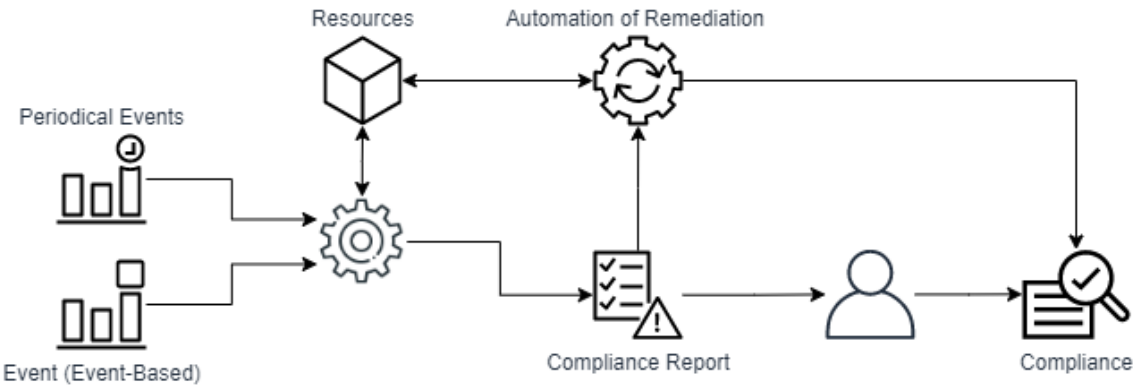


Figure 4 Continuous Compliance Automation

2.6. Cloud solutions and Hybrid Cloud

On demand delivery of IT resources over the internet that often functions as a pay-as-you-go service can be called a Cloud solution. There are many Cloud providers in today's IT market; however, all functions on the principle of providing computing power, storage and other technologies to customers at a pay-as-you-use service without an upfront cost. The ability to deploy an application without the hardware needed on hand is advantageous for start-ups or companies that want to test new solutions. Also, many companies are considering moving to the cloud for better scalability against fluctuating demand and faster delivery of solutions to the customers. [11]

The agility of using a cloud solution can be beneficial for applications that have high variability in usage. On an on-premises service an increase or decrease in server capacity is difficult. This often leads to either having too much or a lack of resources to meet the demand. In cloud environments can scale more efficiently by allocating more resources when the demand is high and release the resources when demand is low, leading to a more cost-effective model than always provision the hardware for the demand. The commercially available services are often called public cloud. The public cloud solutions on the market today all have their differences but follow the same rule of offering a more cost-effective solution to deal with fluctuating demand. There is a solution for on-premises sever configurations that handle the demand fluctuations in some ways using different virtualization solutions. However, the hardware needs to be maintained and exist when peak demand is present, this can be called private cloud.

There are many questions and concerns when moving from an on-premises solution to a cloud provider. The general opinion about moving to the cloud often concerns

security aspects, how can you verify that your data is secure and how can you trust the cloud providers to keep availability at the same level as in an on-premises solution. Therefore, a market for a hybrid cloud solution has emerged. Many cloud providers offer some solutions to having an on-premises datacentre that handles some of the operations.[12]

Hybrid cloud is when using services both from private cloud and public cloud solution in parallel, connecting the on premises serves to external located servers for infrastructure that is flexible both for applications and workloads. These solutions are often highly individual to companies and can be for both flexibility and cost optimization. Hybrid cloud solutions often use the public cloud to handle applications that need scalability and portability meanwhile using the on-premises existing solutions for security, latency and convenience of having the existing solutions in-house. Hybrid cloud solutions often lead to a better response to market change with the scalability and cost-optimization of the public cloud and the safety of on-premises solutions already existing. A hybrid cloud can also be a step for developing new services for existing applications to be more flexible in the marketplace by using the existing on-premises hardware and only paying for what the new services use in the public cloud. [13]

2.7. Compliance in On-Premises versus Cloud Solutions

On-Premises

Compliance is often on an infrastructure level, a very manual process in on-premises environments, hardware and servers are tracked using manual processes and notes about what servers are available at the moment. The developers often do not have the power to introduce new hardware; therefore, the inventory remains somewhat static. The inventory changes are always needed to be manually updated when something changes. Even though some processes are automated, they always require some manual processing. Managing compliance in an on-premises solution is traditionally handled by taking an inventory of the resources in use. The resources are then reviewed one by one by the requirements from the compliance regulation framework and mapped to the existing controls to prove compliance.

Cloud solutions

Manual compliance management falls short in a cloud environment because of the fast-changing infrastructure and the possibilities for developers to change the infrastructure by, for example setting up own “Hardware”. These infrastructure changes are not reviewed by the security or compliance teams, this increases the possibility of the environment not being compliant. The compliance status of the environment cannot be monitored at only one point in time as before in an on-premises solution. Because of the fast pace changes the compliance status of the system quickly becomes inaccurate. This increases the need for automatic compliance checks and checks on the infrastructure as the developers work not to overwhelm the security or compliance teams when deployment is scheduled to production.

Many of the existing cloud providers have taken into account the fast pace environment changes that can occur in a cloud environment. Therefore, many solutions are integrated into the core offering of these cloud providers and many third-party tools and solutions for tracking compliance in cloud environments exist on the market. The importance of providing solutions that ensure easy compliance in the cloud has become an important offering of the public cloud providers. As well as proving that the services already follow several compliance frameworks from the beginning are of great importance by the customers that may want to utilize the opportunities in the public cloud sector.

Amazon Web Services (AWS) that is used in this thesis, has a proven track record of both enforcing and achieving compliance related tasks. By having a large range of tools and solutions for building a scalable and secure infrastructure that facilitates the implementation of the followed compliance requirements. Many of the tools provided by AWS are already certified by many compliance frameworks as well as the clear responsibility structure that is used by the shared responsibility model created by AWS. More information about AWS and its services is taken up later in the thesis, for example in section 3.2. [12]

2.8. Compliance in the Financial sector

As discussed in the introduction, the demand for compliance to different compliance frameworks has increased, both for ensuring quality and safety. This can easily be observed in the financial sector, where demands for proving compliance constantly increase and, in many cases is mandatory for operations in the industry. This not only proving integrity, availability and confidentiality but also ensures customer satisfaction and transparency. [3]

The compliance of the financial sectors includes much more than only the IT solutions used and is often heavily independent on where operations reside. With international frameworks and regulations are shaping the more local regulations to a certain degree. The cost of maintaining compliance has increased significantly and with more compliance frameworks to follow the complexity also increases [14]. The process of analyzing the different compliance requirements and this concerning changes in the application or infrastructure in IT systems often imposes high costs and increased workloads. Often some of the requirements of the different compliance frameworks are overlapping and mapping the requirements to each other has become an important step of being able to prove compliance. [10]

With these things in mind, it becomes clear that a manual process of proving compliance may not be the best solution and hopefully, by introducing automation into the process, the cost and labor can be minimized.

3. Compliance frameworks, services and tools used

This chapter will briefly explain the different compliance frameworks, services and tools used in the thesis and the proof of concept. The thesis is focused on two frameworks that provide recommendations and requirements for improving security for an IT system. The Center for Internet Security Benchmark focuses on recommendations in security best practices and Payment Card Industry Data Security Standard focuses on providing requirements for a safe and secure environment for handling credit card data.

The main service used is Amazon Web Services (AWS) and its underlying services, which provides a cloud computing environment with many services focused on automation and on-demand computing power. For managing the AWS services and resources, the service AWS CloudFormation is used. CloudFormation uses a template format written in JSON or YAML to provision infrastructure and configure resources. The CloudFormation templates are then deployed to AWS using Ansible. Ansible is an infrastructure management and configuration management tool that can manage more than just AWS services and is used in the proof of concept as a continuous integration and continuous delivery tool for high level orchestration. Furthermore, to be able to perform Continuous Compliance Automations, Python is used and to get access to AWS resources and remediate compliance problems the AWS SDK Boto3 is used.

All these services and tools combined enable an entire infrastructure as code solution that supports a Continuous Compliance Automation Architecture and can help with compliance tasks imposed by the Compliance frameworks.

3.1. Compliance frameworks

3.1.1. Center for Internet Security Benchmark

Center for Internet Security (CIS) Benchmark is a global standard and best practices for securing IT systems and data. CIS is a non-profit community-based organization that aims to improve and promote best security practices for IT environments in cyberspace. The CIS controls are developed for best security practices and

configurations for a specific target system. These controls are then mapped to a benchmark for every target system called CIS Benchmarks. Today there are more than 140 different benchmarks for existing technologies. The benchmark is a community developed by achieving consensus between involved actors and experts on best practices to develop and configure different target IT systems securely. The proposed recommendations are published when every party of the development team has reached a consensus about best practices. The CIS benchmark framework is always improved and updated; however, updates depend on community activity for the specific technologies. [15]

CIS controls are divided into three distinct categories. Basic, Foundational and Organizational. These categories contain 20 different controls that help, prevent and detect security issues with a defence in depth model. These controls can be easily mapped to other compliance frameworks and by following the CIS controls or CIS Benchmarks for the system in question the work of implementing other compliance frameworks can be facilitated. [16]

3.1.2. Payment Card Industry Data Security Standard

Payment Card Industry Data Security Standard (PCI DSS) is an information security standard created by four credit card companies: Visa, MasterCard, Discovery and American Express. The PCI Security Standards Council (SSC) was formed in 2006 to act as a global forum for improving, maintaining and raising awareness of security in the payment card industry. The PCI standards are a set of rules that help global organizations improve, maintain, and evolve security measures to protect cardholder data. The standard is enforced by large payment card brands for ensuring safety of cardholder data and mitigating risk of credit card fraud as well as improving the safety of the financial institutions involved in the payment card industry. The standard cover areas are connected to cardholder data both from an operational and technical point of view. The areas are divided into six different categories and these categories define a goal for the requirements. These categories then contain twelve requirements that need to be followed to achieve compliance. The requirements can be found in the table below in Table 1. The key to achieving compliance is to define a scope for the assessment, the scope defines the flow and locations of card holder data as well as all

connected systems that can compromise the data. The scope is important for being able to describe the environment as well as reduces the assessor's ability to verify the compliance of the environment.

Table 1 PCI-DSS Requirements and their goals [17]

Goals	PCI DSS Requirements
Build and Maintain a Secure Network	1. Install and maintain a firewall configuration to protect cardholder data
	2. Do not use vendor-supplied defaults for system passwords and other security parameters
Protect Cardholder Data	3. Protect stored cardholder data
	4. Encrypt transmission of cardholder data across open, public networks
Maintain a Vulnerability Management Program	5. Use and regularly update anti-virus software or programs
	6. Develop and maintain secure systems and applications
Implement Strong Access Control Measures	7. Restrict access to cardholder data by business need-to-know
	8. Assign a unique ID to each person with computer access
	9. Restrict physical access to cardholder data
Regularly Monitor and Test Networks	10. Track and monitor all access to network resources and cardholder data
	11. Regularly test security systems and processes
Maintain an Information Security Policy	12. Maintain a policy that addresses information security for employees and contractors

Different vendors can have variations in how they enforce the PCI DSS; however, all requirements created by the PCI SCC are mandated by the card brands. To get certified a company needs to be assessed using a Qualified Security Assessor (QSA), Internal Security Assessor (ISA) and Approved Scanning Vendor (ASV) that is certified by the PCI SSC. However, if a merchant only handles a small number of card transactions, a

Self-Assessment Questionnaire (SAQ) can be sufficient to achieve compliance. There are four distinct levels of PCI compliance that directly correlate with the number of transactions the merchant has and assessed risk level. The assessments are done annually, but some parts of the process are done at least quarterly.[17] [18]

3.2. Amazon web services and tools

Amazon Web Services (AWS) is a public cloud provider owned by Amazon that provides an on-demand cloud computing platform that started operation in 2006. The cloud provider offers pay-as-you-use pricing and no upfront cost with long commitments. The solution offered to customers is scalability, high availability and low-cost infrastructure for operating in 190 countries globally. The customer can instantly deploy new applications on new servers as well as scale to the demand flexibly using the built-in scalability of the solution. AWS offers more than 200 services that help developers and businesses create applications for their own needs. These services include networking, computing, storage, management tools, developer tools and security solutions, among others.[19]

The cloud infrastructure is divided into different regions that is in a different geographical area. These regions are then divided into availability zones that are individual failure domains. The domains have their own local zones that can host services at replicated locations. Within a local zone the latency is low. Within an availability zone the infrastructure is isolated, and each region is completely independent of each other. A visualization of the different zones can be found in Figure 5. This infrastructure enables the customers to choose between high availability and low latency while being able to implement fail safe infrastructure by choosing the optimal deployment strategies. [20]

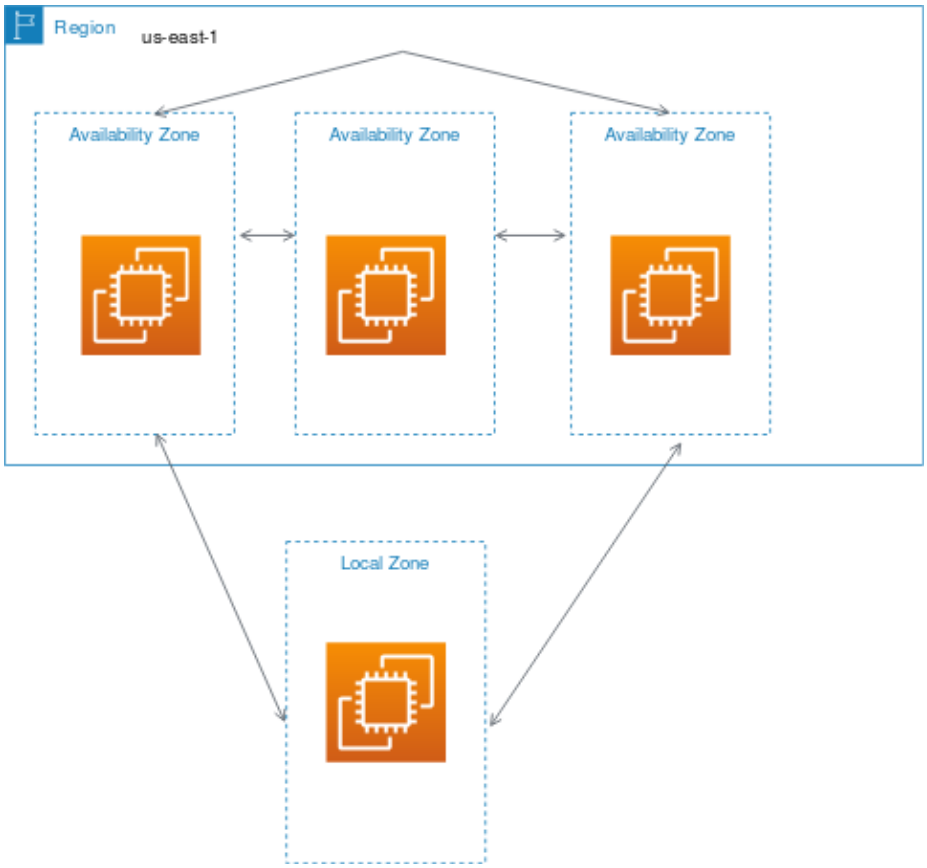


Figure 5 AWS Global Infrastructure, with Regions, Availability zones and local zones [20]

3.2.1. Shared responsibility model

AWS has created a concept to enforce the proper security and compliance model for customers using the service. The model is created to clearly define the responsibilities of the cloud provider and the customers. With using the model and the documented functionalities of the services the customer is using, the security implementations needed can vary. However, by defining two separate areas of responsibility, the customers can focus on the right security aspects and leave the other parts to AWS responsibility. The model is divided into AWS responsibility of "security of the cloud" and Customer responsibility of "security in the cloud". AWS is responsible for the overall infrastructure of the cloud service provided, with security of physical location, hardware, networking, and some software solutions. The customer is responsible for the security in the product they use and the implementation of applications in the cloud, including safe configurations, operating systems, applications, and data processing. AWS tries to visualize the difference between Customer responsibility and AWS responsibility using Figure 6.

AWS provides services in which the customer directly inherits the security responsibilities. Some services are shared responsibilities where AWS is responsible for some of the security controls. However, the customers are responsible for applying or enforcing the controls and configuring the services correctly as recommended by AWS. The security controls can also be completely customer specific and in complete responsibility of the customer. Based on the different levels of responsibilities, the customers can use the provided documentation and controls to evaluate compliance accordingly using the guidelines in the Shared Responsibility model. [21]

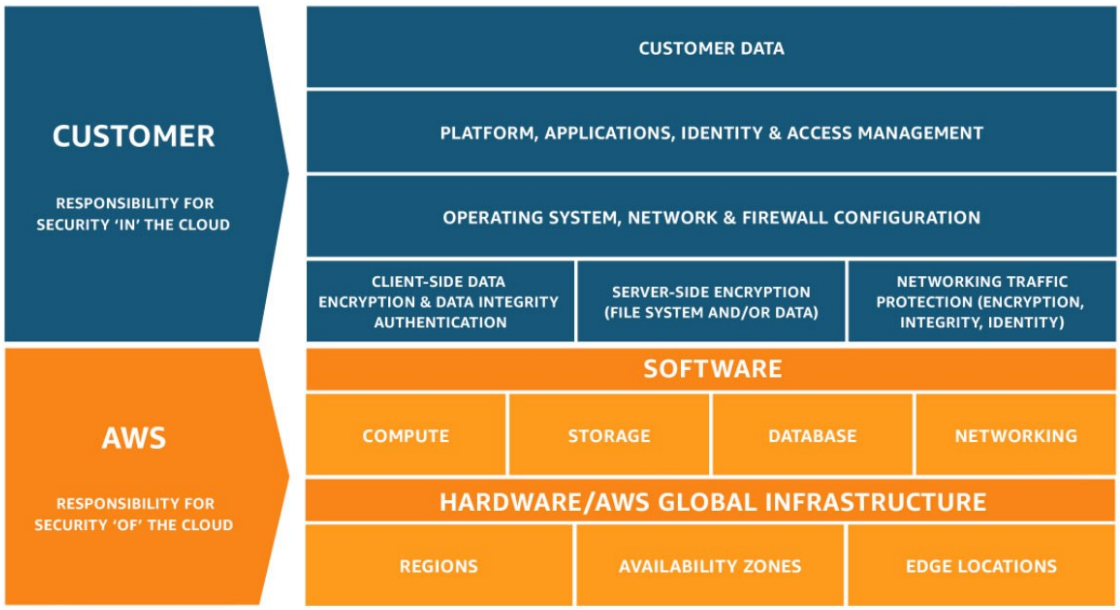


Figure 6 AWS Shared responsibility model [21]

3.2.2. A brief overview of AWS services used

As described in the chapter beginning, many AWS services are used for the proof of concept. Managing all the resources used, the service CloudFormation functions as an infrastructure and configuration manager and to manage access between the resources and users the Identity and Access Management service is used. For monitoring changes and events of the resources AWS Config is used. For finding compliance problems in conjunction with AWS Config, the CCM tool, AWS Security Hub, is used. Furthermore, CloudWatch is used to filter out events caused by non-compliant resources and forward the information to AWS Step function for orchestration of computing resources like AWS Lambdas and message resources like Simple

Notification Service. For creating remediations, the AWS SDK Boto3 is used, allowing manipulation of the resources and their configurations. All these services are stitched together and used. To give an understanding of what they are and how they function a brief overview is presented in this section.

AWS CloudFormation

One way of managing AWS resources is to use the service AWS CloudFormation, a template format that describes and configures AWS resources. The template code describes the infrastructure and configurations wanted in the AWS environment, specifying provisioning and much more. CloudFormation is often called infrastructure as code that can deploy application infrastructure and services directly in AWS. Using CloudFormation, the infrastructure can be versioned and deployed using pipeline solutions or manual deployment to automate and securely manage AWS resources. The templates are written in JSON or YAML format. CloudFormation creates a stack that constructs and configures the stack resources. The stack keeps track of changes and deconstructs the resources as well as builds them up again after code changes. Using CloudFormation automation of creating infrastructure can be achieved, creating secure configurations that can be repeated as well as controlling changes that can be reverted in a simple manner. [22]

Identity and Access Management IAM

To control access to AWS and AWS resources the Identity and Access Management (IAM) service is used. The IAM allows for controlling access across different accounts and manages access and permissions of the resources used. When starting to use AWS, a root account is the first setup that has permissions to the whole AWS infrastructure that is needed. However, it is recommended that an IAM account is setup even for managing administrative tasks as well as accounts for managing the services used on appropriate levels. With IAM, a user can specify what a certain account, group or role can do and cannot do by assigning granular permissions using permission policies. The permission policies are often defined on different levels in JSON format. Using identity-based policies, a user or group can be managed by defining rules that control what that identity can perform on a certain resource and under what conditions. If a policy only controls permissions for one user, the user can be described in the policy

as a Principal. There are also resource-based policies that are applied on a resource level. On this level the user can specify what actions and under what condition a certain resource or principal can perform on that resource. Service Control Policies are policies that manage all accounts across the organization and are the highest policy level. However, this feature is only enabled if the organization does not use the consolidated billing feature. The policies are handled on many levels and multiple policies can be applied to a certain resource. AWS evaluates all policies starting with the assumption that everything is denied and only allows access if a policy explicitly defines access to the resource or identity. [23]

AWS Config

AWS Config is an AWS service that tracks and keeps a detailed view of your AWS resources as well as their configurations. The service enables monitoring of the resource configurations as well as evaluating resources with rules that check the desired configurations. The monitoring can function as an audit of what has happened to the resources in the AWS account, acting as a change management tool as well as compliance auditing and security analysis using continuous assessment. The evaluation of configurations is done using an AWS Config Rule that defines desired configurations and checks them against resources and changes to resources in your AWS environment. The rules can be evaluated when resources are changed, triggering a Config Rule evaluation or the rule can be triggered to check resources at a selected time interval. Some resources can only be monitored through periodical checks because changes do not trigger Config events in the AWS environment. The evaluation result or auditing result is then sent to other AWS services like Amazon S3, Amazon CloudWatch and are displayed in the Config console. [24]

AWS Security Hub

AWS Security Hub is a service that aggregates findings of security concerns in the whole account, using different integrated AWS services or AWS partner integrations. The service is focused on collecting security and compliance events and prioritizing them for decision-making actions. Security events of large environments can easily become cluttered with false positives and Security Hub ties to prioritize the critical security compliance events of the environment. Security Hub uses a Finding Format

that is a standardized format of information about the findings Security Hub has detected, that then can be sent to different services for notifications or Security Orchestration Automation and Response tools as well as other services can send their result to Security Hub using the AWS Security Finding Format (ASFF). Security Hub can be enabled to run automated compliance checks on the AWS account for some industry standards and best practices, like CIS AWS Benchmark and PCI-DSS for improving compliance in the environment. The compliance checks are based on AWS Config rules that Security Hub sets up automatically when the standards are enabled. [25]

Amazon CloudWatch

For monitoring of the AWS services in use the service Amazon CloudWatch observes and collects information about resources, applications and even on-premises data if integrated. CloudWatch collects data in the form of logs, metrics and events from all services used in the environment. Using different features in CloudWatch a user can set alarms, visualize logs and metrics, and take action on events and trouble-shoot applications. This enables response actions on events in different ways and creates workflows based on event-driven architecture and notifying about anomalies happening in the environment. The service can also collect metrics about CPU usage from some services and can be configured to notify about the operational health of servers. [26]

AWS Lambda

One of AWS services for computing is AWS Lambda, a serverless computing unit running without provisioning or managing servers. The code is automatically executed in response to triggers or events and is scaled by automatic processes as more computing power is needed as well as being able to run parallel executions. AWS Lambda automatically manages all configurations and scaling, leaving only the responsibility of coding to the user. Lambda supports several languages, including Python, Java, and Node.js. [27]

Amazon Simple Notification Service

Amazon Simple Notification Service (SNS) is a messaging service that handles publish and subscription messaging between services in AWS. SNS is highly available and managed by AWS, giving a reliable push-based message service that can be used for serverless architecture and application integration. SNS can also publish messages on a topic with parallel subscriber endpoints for processing and sending SMS and email. The message published in the topic can be customized to fit any service as long as the message is published in JSON format. [28]

AWS Step Function

To coordinate many AWS services into workflows AWS Step Functions stitches the services together. By using Step Functions many services can be combined into an easily managed workflow that also can be visualized for added visibility of the workflow. This enables application-like structures for coordinating, for example, serverless computing units. The individual step in the function performs a discrete function, and the Step Function records each step and result, including errors, making debugging and problem solving easy. The Step Function is written in JSON-based Amazon States Language, resulting in a State Machine that can easily be visualized and changed. [29]

AWS SDK Boto3

To be able to change and manage AWS resources from code, the AWS software development kit Boto3 is used. Boto3 allows for creating, updating, and deleting certain resources and services in AWS with Python code. The SDK provides access to resources through the object-oriented API and allows for low level access to these resources. Boto3 supports a range of different AWS resources and accessing these is done by importing the library as well as specifying which client to use. The code can access the specified resource with functions as describe, update, or delete. A simple example of how to use Boto3 is shown in Figure 7. [30]

```python
import boto3

ec2 = boto3.client('ec2')
response = ec2.describe_instances()
print(response)
```

Figure 7 Example of using Boto3 in Python code [30]

3.3. Ansible and CSE

Ansible is an automation engine that automates configuration management, cloud provisioning, application deployment and infrastructure orchestration. Ansible uses an automation language that is written in playbooks that describe the infrastructure of an application or task. The tasks in a playbook can be called plays and are executed in sequence. The Ansible automation engine executes the playbooks on every node that is included in the inventory list. The playbooks are written in YAML, a human-readable automation language that is easy to understand. An example of the format is shown in Figure 8.

```yaml
- name: create a cloudformation stack
  cloudformation:
    stack_name: "ansible-cloudformation"
    state: "present"
    region: "us-east-1"
    disable_rollback: true
    template: "files/cloudformation-example.json"
    template_parameters:
      KeyName: "jmartin"
      DiskType: "ephemeral"
      InstanceType: "m1.small"
      ClusterSize: 3
    tags:
      Stack: "ansible-cloudformation"
```

Figure 8 Example playbook [31]

Ansible uses agentless deployment and connects to the nodes using SSH, pushing out small programs, and after the programmes are completed, the files are removed. The small programs are called Ansible modules, there are many modules to choose from and the modules can be directly incorporated into custom playbooks extending the capability of the playbooks as well as creating complex task executions and configurations. For more complex tasks, an Ansible role can be created, that is a playbook that is self-contained. The role has tasks like any other playbook, but it can also have specific variables, configuration templates and other files. The different levels of roles, tasks and files are shown in Figure 9 where best practices for organizing these are shown. [32]

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them here (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Figure 9 Best practices for Ansible playbooks [33]

In this thesis the ansible module “cloudformation” is used, that is a module for creating, updating, and deleting an AWS CloudFormation stack to and AWS environment. With this module it is easy to create custom CloudFormation templates that can provision and configure resources into the AWS cloud environment. CloudFormation can in this case, be used as an infrastructure management tool and Ansible is used as a configuration management tool. Combining both Ansible and CloudFormation gives more flexibility and possibilities for managing infrastructure. The entire process can be further automated for Continuous Integration and Continuous Delivery using higher-level orchestration systems. This can be done using, for example, Jenkins, other services like AWS CodeCommit in combination with AWS CodeBuild or Ansible Tower. [31] [34]

Ansible Tower is an automation hub for automating Ansible tasks to manage deployments and visualize them. Ansible Tower uses a Web-based console design for easily visualizing and management of the IT infrastructure. Ansible jobs can be scheduled to run full continuous delivery pipelines running automated Ansible tasks and in this way manage infrastructure by version-controlled code. The possibility to fully automate infrastructure by code and integrate this with a continuous delivery pipeline is therefore easy to achieve using a product like Ansible Tower. [35]

3.4. Programming Languages

The primary Programming language used in the proof of concept is Python. Python was chosen mainly used because of its readability and highly supported libraries for AWS resource manipulation like Boto3. And as discussed in previous sections the serializing languages JSON and YAML are heavily used by both AWS services and Ansible templates and therefore have a heavy presence in the proof of concept.

3.4.1. Python

Python is a high-level programming language that was created by Guido van Rossum. The language is focused on code readability and has an object-oriented approach. The biggest difference from other programming languages in the same category is that Python uses new lines for command completion instead of semicolons and other characters. The language also relies on indentations for defining scope, as can be seen in the example of a function in Figure 10. Python has a comprehensive library and is often used due to its versatility and modularity. As well as its readability because of its resemblance to the English language. It also does not require compilation and therefore, all code is executed at program start. Therefore, minimize the edit-test-debug cycle. [36][37]

```python
def say_hello():
    # block belonging to the function
    print('hello world')
# End of function

say_hello()  # call the function
```

Figure 10 Example of Python code [36]

3.4.2. YAML and JSON

YAML is a serializing language that is human readable and is often used for configuration files, but it is easily serializable for any data structure. The format is often used for its readability and broad support in many languages. YAML can easily be converted into JavaScript Object Notation (JSON), but they are not completely interchangeable. YAML cannot contain duplicated keys, but JSON allows duplicates, meaning a JSON-object cannot always be converted into YAML, but almost every time a JSON-object can be converted into YAML. JSON is often used for its simple form and ease of use, as well as faster processing. YAML is often used for its better readability and ease of serialization of arbitrary data structures. YAML can in some cases, be more complex and therefore harder to parse than JSON. This is due to relations between different sections in YAML. JSON uses bracket syntax but YAML uses indentations. A small example of the syntax can be found in Figure 11 and Figure 12 .[38][39]

```yaml
- name: Sammy Sosa
  hr:   63
  avg:  0.288
```

Figure 11 YAML example

```json
{
  "Sammy Sosa": {
      "hr": 63,
      "avg": 0.288
    }
}
```

Figure 12 JSON example

4. Related work and current solutions

There are many compliance tools on the market, with different philosophies and solutions to continuous compliance and security problems. Every solution has positive and negative sides, as well as supports cloud solutions or hybrid cloud to a different degree. Also, the discussion on security and compliance in cloud solutions is a topic of debate. How good are current cloud providers to ensure safety and compliance? How can a cloud provider be compliant? Therefore, many discussions on compliance and safety in cloud environments have taken place over the years.

In this chapter, several solutions that inspired the thesis are explained and investigated and the topic of cloud solutions in relation to the compliance framework.

4.1. Cloud solutions and Compliance frameworks

When public cloud providers started to emerge, there was a common consensus that the cloud was not as safe as on-premises solutions. However, there have been a number of improvements to how cloud providers handle security and compliance in this area as well as research on how cloud solutions can be safer or easier to maintain compliant than on-premises solutions.

In the master thesis, "Can PCI DSS compliance be achieved in a cloud environment" by Durkin, Patrick the topic of achieving compliance of PCI DSS in different cloud environments are discussed. The thesis investigates every part of the environment and the use of different cloud providers and how the cloud solutions are affected by the compliance requirements. The thesis concludes that with the right approach to implementing compliance requirements and the awareness of cloud architecture, PCI DSS compliance can be achieved. The implementation is heavily dependent on the ability of the security professionals and some increased workloads for logging network access can be expected. Therefore, a hybrid solution is proposed where payment processing is not done in the cloud. [40]

4.2. Solutions in Continuous Compliance

The concept of continuous compliance is a relatively new concept even for cloud providers. However, there are many different solutions on the market to support the compliance related work with a continuous process. The tools and companies that supply continuous compliance solutions all define the continuous compliance concept differently, resulting in different approaches in the end product. Many of the continuous compliance products are an addon to an existing product that companies may use, which is often the case when it comes to cloud services.

There are not that many research papers available on the subject. However, the paper "Continuous Compliance: Experiences, Challenges, and Opportunities" introduces the concept of Continuous Compliance and its use in real-time infrastructure compliance states. The paper provides an implementation using a tool called Chef for real time monitoring of infrastructure resources or endpoints and their configuration states, comparing them to a compliance policy using the Chef recipe language. The compliance checks are run automatically, and reports are compiled of the compliance state of the systems. Also, remediation monitoring of the fixed compliance issues was tracked. The solution is an automated process for both the developers and security or compliance personnel. However, the specifics of the solution are not discussed. [3]

In the Pluralsight educational video "Managing Inventory, Change, and Compliance with AWS Config" by Paul Kirby the Continuous Compliance concept is discussed in a solution based on an AWS service called AWS Config. The solution explained contains the explanation of why compliance is important to continuously monitor and track as well as the benefit of using automation in AWS for achieving the compliance goals. When achieving compliance, the different stages of assessments are presented and the possibilities to achieve Continuous Compliance with either a fully managed AWS solution or a customized solution using either third-party-developed functionality or self-developed solutions is presented. Also, a solution for implementing automation into the remediation process for non-compliant resources is presented using AWS Config, AWS SNS and AWS Lambda. The solution presented is sufficient for small applications and can be seen as a good starting point for any Continuous Compliance Automation solution. However, the solution has some

drawbacks in the design as well as it cannot be scaled to no more than a few automatic remediation tasks.

This was the initial idea for automating compliance tasks that inspired the thesis topic. The solution proposed in the course had to be improved, and no specific compliance requirements were targeted. The concept of automating the entire process from finding a compliance problem to fixing the problem inspired the thesis. [2]

In an AWS blog post by Jonathan Rau called "Continuous compliance monitoring with Chef InSpec and AWS Security Hub" the concept of using Chef InSpec as a Continuous Compliance Monitor in AWS is discussed. The solution uses Chef compliance checks and ingests the results of them to AWS Security Hub. This improves the customizability of compliance checks for the customer and with this solution an extension to Security Hub default compliance checks can be achieved. [41]

In other AWS educational materials, product slides at conferences and blog posts solutions variations on Continuous Compliance processes have been shown. The possibilities to use AWS services to detect, respond and remediate security and compliance problems are advertised in the documentation to some extent [42]. Some concepts of Continuous Compliance solutions have been shown. However, the code and implementation are often not given. Also, the lack of automation beyond the compliance solution itself is often missing. With the release of Security Hub in the summer of 2019, a complete offering supports Continuous Compliance Monitoring solutions. The possibilities of remediations are a hot topic for the time being and implementing an automatic remediation functionality in AWS using different services has been shown as possible during case studies. These have not yet been released as a service and only the possibility to use AWS services for the purpose of remediating compliance change has been discussed in the solutions [43].

An open-source tool was released during the thesis implementation and writing that is a Continuous Compliance Monitoring tool for AWS infrastructure as well as containing a module for automatic remediation actions in the AWS environment. The tool is called ElectricEye, released on16.03.2020 by Jonathan Rau. [44] The solution

presented in ElectricEye is a real contender to the Proof of concept in this thesis, with the multiple compliance checks as well as complete remediation of many more compliance requirements.

The solution lacks some of the functionalities implemented in the proof of concept, such as error handling and notifying the team of compliance problems. The solution was not possible to integrate into the proof of concept done for the company due to using an already existing method for CSE. The purpose of the proof of concept was to automate every part of the process using already available methods and infrastructure.

The possibilities for implementing Continuous Compliance Automation in AWS are advertised across the platform and other forums. However, a solution that is ready to be used directly does not exist yet. Also, a solution that uses infrastructure as code and concepts of CSE is not directly available on the market without purchasing a complete service. In addition, the topic is not discussed in academic work in general.

Implementing a Continuous Compliance Automation solution using AWS and automating the whole process is a relatively new approach and all the above examples inspired the idea to make such proof of concept.

5. Compliance and AWS

Compliance and security considerations are the highest priority of AWS solutions. The architecture of the solutions provided is built to comply with many Certifications, Laws, Regulation requirements and best practices using different compliance frameworks as a foundation for achieving that. AWS aims to enable the customer to comply with their required compliance programs with as little effort as possible by using the correct AWS service and recommended architecture for best practices. However, by simply using these services, the system is not automatically compliant, the architecture, configurations, applications, and data storage is up to the user/customer to manage for achieving compliance. AWS supplies tools and guidelines on managing and creating an environment that is compliant and by following these guidelines, the compliance process can be easier. [42]

In this section, the different security concepts of AWS are briefly discussed and explained concerning the Proof of Concept done in the thesis. AWS Security and Compliance is a vast area, and the information is constantly improved. Therefore, this section is limited to understanding the used services and tools in relation to the proof of concept and compliance frameworks.

5.1. AWS strategies for compliance

As discussed in section 3.2.1 the Shared Responsibility model is a tool for clearly defining boundaries where the customer has responsibilities and where AWS is responsible for the security implementation. The responsibility of AWS itself is to maintain and provide a secure platform for the customer to use as well as tools for the customer to improve their security posture. The platform needs to facilitate and promote secure operations so that the customer can easily configure and use the AWS services securely and according to the relevant compliance requirements they need to follow. Examples of tools to help customers with their security posture are AWS Identity and Access Management (IAM), Amazon Inspector, Amazon Macie, AWS Security Hub, AWS Config, AWS CloudTrail and AWS CloudWatch.

For securing the cloud platform itself, AWS uses commonly used practices such as Continuous monitoring, Penetration testing, Compliance certification and secure

coding practices. Automated Reasoning is used to secure the cloud platform itself and help secure the customer's implementations to further improve security. Automated reasoning can be described as mechanical reasoning in mathematical logic to provide additional assurance, in short, proving mathematically and logically that a resource fulfils a set of rules. This is a kind of formal verification that things work as expected. AWS uses these techniques to create tools for verifying internal code as well as customer related implementations for improving security.

One of the Automated Reasoning tools used is ZELKOVA, a policy analyzing engine that reason about AWS access control policies and their validity. The policies are converted into Satisfiability Modulo Theories (SMT) that can then be solved and verified for easy-to-understand answers about the correctness of different access policies. AWS uses IAM (Identity and Access Management) policies that consist of a Policy language that defines access to resources and interactions between resources as well as access for users. Policies in AWS define access control across various services and can become quite complex when combining these services. As a user, you often need to ask yourself questions about what the policy that was created means, what users can or cannot do with the policy restrictions. However, ZELKOVA tries to automatically reason over all possible contexts if the policy complies with common best practices. A policy evaluating engine handles all requests to an AWS service, comparing the context to existing policies, granting or denying access to the service. ZELKOVA reasons over all possible requests and can easily determine if a policy change can lead to more access to a resource than before and from this the user can be informed that a policy needs changing to increase security. This information facilitates security improvements that are easy to correct and can significantly improve the security and compliance of the service used by the customer.

Previously users and AWS have used different kinds of checks to detect dangerous patterns in the access policies using all kinds of pattern checking and static comparing the policies. But, these heuristic-based syntactic checks, pattern matching, or simulation checks are of no use when involving multiple services and multiple access control policies. They also become difficult to maintain, especially if we try to create patterns for all possible scenarios and resources change quickly.

An example of what kinds of checks ZELKOVA can do is whether you have the correct permission set on an S3 storage bucket. As a principle you should never have public read and write access to a S3 bucket, instead, you should define who and what has access to the bucket. This is also a compliance rule in PCI DSS 7.2.1 and CIS AWS Foundations 1.22. In Figure 13, a user has created an S3 bucket with the intent to restrict the read/write access to the bucket to only one specific principal. However, a human error has occurred, and the policy states the principal in the key value pair "NotPrincipal". This causes the policy to allow public read/write access to the bucket but not for the intended principal.

```
{
    "Effect": "Allow",
    "NotPrincipal": { "AWS": "111122223333" },
    "Action": "*",
    "Resource": "arn:aws:s3:::test-bucket"
}
```

Figure 13 IAM Policy for S3 bucket with public read/write

When the user saves the policy, ZELKOVA will process the policy and compare it to all known combinations of access requests, a known policy that uses public access. Using mathematics and SMT, ZELKOVA can determine that your policy is more permissive than allowed and therefore label the policy as Public, as can be seen in an example of the console in Figure 14. The evaluation is done using AWS Lambda and takes only a few milliseconds in most cases. Then a red label "Public accessible" is almost instantly displayed to the user to indicate that something is not correct.

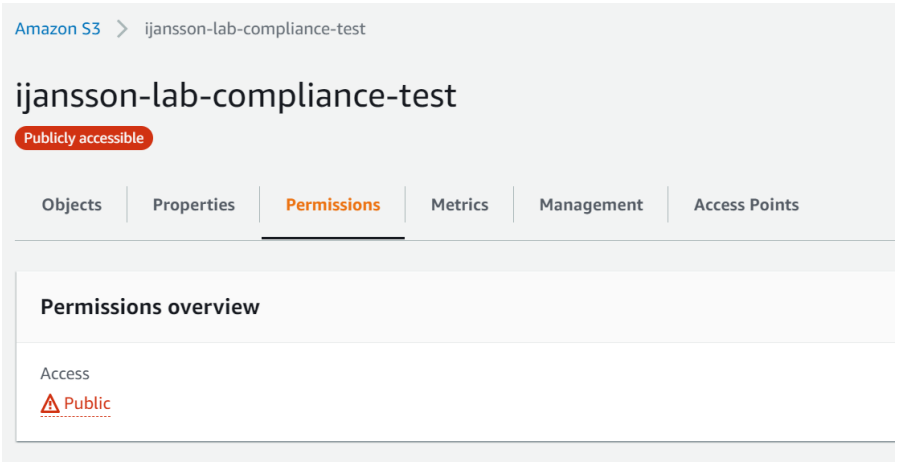


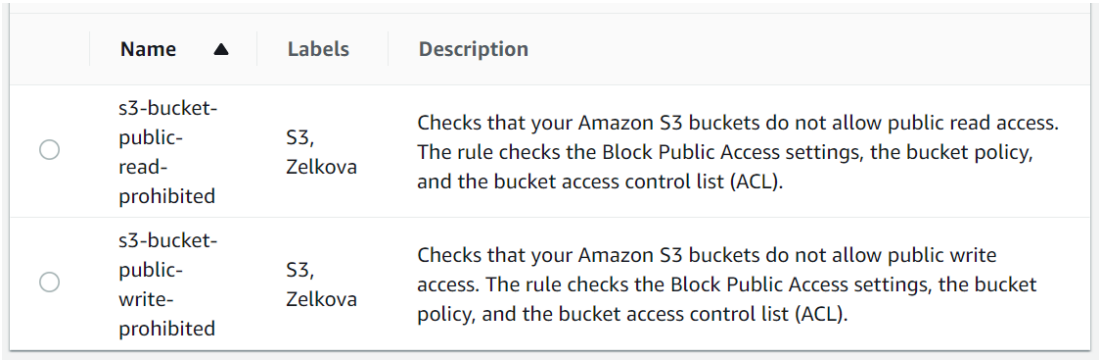
Figure 14 ZELKOVA warning for public policy on S3 bucket

```
{
    "Effect": "Deny",
    "Principal": { "AWS": "111122223333" },
    "Action": "*",
    "Resource": "arn:aws:s3:::test-bucket"
}
```

Figure 15 IAM Policy for S3 bucket with read/write to a specific principal

The recommendation is not to allow Action: ”*” with Effect: “Allow”. In Figure 15, we have fixed the human error and specifies that only this principal has no access to the bucket. As there can be many IAM policies to every resource on different levels, the complexity can be challenging for a human to understand and using ZELKOVA AWS can mathematically prove that some things can be a security hazard.

ZELKOVA is integrated into many AWS services, one of them AWS Config, where configuration changes trigger Config rules that uses AWS Lambda with ZELKOVA in the background to evaluate if a resource is compliant or not. In the above example the Config rules triggered where s3-bucket-public-write-prohibited and s3-bucket-public-read-prohibited for that specific resource, in the console they have the label Zelkova as can be seen in Figure 16. Config triggers the rule either on resource creation or on a configuration change, in this case, policy change. The other services like Amazon Macie and Trusted Advisor service are also using ZELKOVA in similar ways.



	Name ▲	Labels	Description
○	s3-bucket-public-read-prohibited	S3, Zelkova	Checks that your Amazon S3 buckets do not allow public read access. The rule checks the Block Public Access settings, the bucket policy, and the bucket access control list (ACL).
○	s3-bucket-public-write-prohibited	S3, Zelkova	Checks that your Amazon S3 buckets do not allow public write access. The rule checks the Block Public Access settings, the bucket policy, and the bucket access control list (ACL).

Figure 16 ZELKOVA as a Label for AWS Config Rules in the AWS Console

ZELKOVA is also used in many internal AWS security auditing tools to evaluate compliance of all internally made resources as well as misconfigured policies. Some of the evaluations are done with configuration change, but there are also periodical scans for compliance problems. ZELKOVA is not the only tool used to improve security across the platform but an essential improvement to the already extensive existing policy checks in place for both the infrastructure as well as for the customer used services. [45] [46]

5.2. AWS Services to help with Compliance for customers

As discussed in the previous chapter, AWS facilitates different tools and services to help a customer with compliance and security issues. However, compliance is often more advanced than only looking into security. There are other areas that may need documentation and manual assessment for reaching a compliant state. AWS Config records configuration changes and can be used as a change management service to view past configuration changes and evaluate them easily. AWS Config has config rules as discussed in earlier sections, that check compliance. The rules can be managed rules created by AWS, but it is also possible to create your own config rules. The managed rules are used by other services like AWS Security Hub that is a centralized hub for security and compliance events. Multiple AWS services findings, not only AWS Config, are ingested into AWS Security Hub and there are also possibilities for ingesting similar kinds of findings by third-party tools. In AWS Security Hub, events can be sorted, filtered, and prioritized to improve your environment's security posture. AWS Security Hub is discussed in more detail in the proof-of-concept section. AWS

Config can also be integrated with AWS CloudTrail for logging configuration changes done through API calls. AWS CloudTrail can log changes across deployments and accounts to get a better understanding of the user activity done through AWS management console, AWS SDKs and command line tools. This to increase visibility for possible security analysis and compliance auditing tasks. With the integration of CloudWatch events, monitoring of security events found in the CloudTrail logs can be handled. CloudWatch is a complete application and infrastructure monitoring solution that supports monitoring of most resources and logs and can notify on specific logs or events in the environment. CloudWatch Events can be forwarded to other AWS services for processing, creation of notification to different channels, auto-scaling, resource optimization and Remediations.

These are only some of the services to help customers with compliance and security when using AWS services. However, it is still the customer's responsibility to use these services to help with compliance. As well as using the available services in a way that compliance can be reached towards different frameworks according to the security responsibility model. [24][25][26]

5.2.1. Compliance Certified Services in AWS

As the shared security responsibility model implies, AWS has the responsibility to remain compliant and is responsible for the security of the cloud. That means that AWS also needs to comply with industry standards and laws and undergo assessments independently. To ensure that the customer can operate in an accredited environment. AWS computing environments are certified by accredited bodies and are continually audited to ensure that the customers can take advantage of not having to do the whole assessment themself. Therefore, the customers can reduce their scope as well as cost by using already accredited services. The services certified by a specific compliance framework can be different depending on the operational region. However, most regions are certified with SOC 1/SSAE 16/ISAE 3402(formerly SAS 70), SOC 2, SOC 3, ISO 9001 / ISO 27001, FedRAMP, DoD SRG, and PCI DSS Level 1. When using a service, AWS provides compliance certificates for the different services to the customer, enhancing transparency between the parties. The customer will find most of the certificates needed for compliance assessments in the AWS Artifacts self-service

portal. Therefore, the customer can inherit the controls needed to be compliant with a certain framework from AWS. Further, AWS also supplies guidelines, templates, and compliance mappings to facilitate the process of the customer becoming compliant to a framework when using different AWS services. [47]

One thing that is important to remember is that a customer can still choose to use the service in an uncompliant way. The customer is responsible for using and configure the service for the compliance requirements that they face. In other words, an AWS compliant service enables the customer to be compliant and by only using the service the customer is not automatically compliant.

5.3. CIS for AWS

CIS benchmarks areas discussed in chapter 3.1.1 a self-imposed guideline for safe operations of IT systems. AWS has put together a series of requirements that are based on the CIS controls for AWS services. The recommendation is for configuring best practices as well as hardening used services by the customer. This guide is called AWS CIS Foundational Benchmark and contains a series of recommendations to be followed to be compliant with the CIS recommendation. The guide is divided into two levels. Level 1 for necessary recommendations that provide clear security benefits and are the best practices. Level 2 for recommendations that are necessary if the security needs are essential to the environment and these recommendations are advanced in-depth measures. Furthermore, the guidelines are separated into different sections and areas of interest, like Identity and Access Management, Storage, Logging, Monitoring and Networking.

Recommendations in Identity and Access Management focus on IAM configurations, password management and management of roles and groups. In this section, there are 22 recommendations, for example, 1.5 to 1.10, that focus on password hardening. The recommendation “1.5 Ensure IAM password policy requires at least one uppercase letter” is recommended in combination with the other recommendation to ensure password policy complexity.

The three Storage recommendations are focused on the storage services in AWS like S3, and the importance of safe handling of data. Where recommendation “2.1.1 Ensure all S3 buckets employ encryption-at-rest” is for ensuring that data at rest is in an

encrypted state for safe long storage, ensuring that if data is exposed, the impact can be minimized.

The next section of recommendations is Logging configurations that will improve visibility of what is happening in the AWS accounts. For example, one of the eleven recommendations is “3.1 Ensure CloudTrail is enabled in all regions" which will ensure that all AWS API calls can be logged and monitored. This enables compliance and security work with analysis, compliance auditing and change tracking across the AWS accounts.

The Monitoring section contains 15 recommendations that will improve the visibility of what is happening in the environment and inform personnel of particular concerns. The recommendation is to setup different CloudWatch alarms with log metric filters to find different events that can be deemed dangerous or compliance breaching. For example, “4.3 Ensure a log metric filter and alarm exist for usage of "root" account” that introduces an alarm that notifies if the “root” account is used, highlighting the frequency or unwanted use of this powerful account.

The last section is Networking, with four very basic recommendations. For example,” 5.2 Ensure no security groups allow ingress from 0.0.0.0/0 to remote server administration ports” limiting public access to a remote server to minimize attack surface or reduce risk of compromise of a resource.

All these recommendations are defined as automatic or manual, depending on what the guidelines can offer for the process of implementing them. [48]

All the requirements are integrated into compliance checks in AWS Security Hub and the real checks can be found in the AWS Config module. Many of the recommendations explained can be automated using these services and the manual tasks are only setting up the different configurations and alarms. [25]

5.4. PCI DSS for AWS

AWS provides guidelines that inform about PCI DSS compliance state of AWS services. AWS also provides recommendations on architecture, services, and configurations for customers to follow. This is to be able to achieve compliance when following these guidelines. The core principles of compliance for PCI DSS in AWS are to provide an environment where compliance can be achieved. The certified services assume that cardholder data and sensitive data can end up in AWS services. Therefore, datacentres and services are assessed as if they would process, store, or transmit that kind of data. This is according to the PCI DSS scope on behalf of the customer. But the customer is still responsible for how they use the services and implement their applications according to the shared responsibility model. As discussed in chapter 3.1.2, the scope of the assessed environment is essential and AWS guidelines try to minimize the scope by providing standardized architecture to follow for reducing the scope of the cardholder data processing. There are also available CloudFormation templates with basic infrastructure and networking for a possible PCI compliant architecture that can be used as a starting point for any customer that needs a PCI DSS compliant environment.

The 12 different PCI DSS requirements all have a recommendation or guidelines on how configuration should be done to be compliant in AWS or what AWS service should be used for achieving compliance. One easy example of a requirement that AWS guidelines provide a solution for is “3. Protect stored cardholder data” and subcategory 3.4 where according to the PCI DSS, all data that might contain cardholder data needs to be encrypted to the degree that if the data is exposed that data cannot be decrypted, this both at rest and in transit. This can be achieved by using, for example the AWS S3 storage service that supports encryption for both storing the data as well as data in transit. Furthermore, the sub requirement 3.5 is to use secure key management for the encryption keys where AWS Key Management Services provides automatic key rotation and key policies. Also, with CloudTrail enabled on these services, the requirement “10. Track and monitor all access to network resources and cardholder data” can be satisfied in this example. The customer still needs to implement the compliance requirements on an application-level if they use another database service other than the certified AWS Data Base services. [49] [50]

As mentioned in the previous chapter about CIS benchmarks, AWS has integrated all automated checks for PCI DSS into AWS Security Hub using AWS Config for all automated compliance checks with guidelines on manual tasks for solving the compliancy problem. [25]

5.5. Mapping of Compliance frameworks

A central part of implementing a continuous compliance workflow and automation of the continuous compliance process is to map different compliance frameworks. This will decrease the duplicated work if many compliance frameworks need to be followed. The number of compliance frameworks that a company needs to comply with can wary, but often they are more than one. The requirements for the different frameworks can be completely different and some of the compliance requirements can exactly have the same goal. Therefore, it is essential to have a complete picture of which compliance frameworks are used and how they relate to each other. If a requirement can either be completely covered or partially covered by another compliance framework requirement, the workload can be minimized. [10]

There are many guides and mapping tools or mapping matrixes existing for many of the available compliance frameworks. However, it is always important to look at the specific usage of the compliance framework in a solution and how the mapping can be different with using specific services. The Center of Internet security has an easy tool where the CIS Controls are mapped to other compliance frameworks or regulations like PCI-DSS, ISO27001 and NIST. This mapping is done from a CIS perspective and only explains to which controls the requirement directly are related. Therefore, it can be beneficial to compare the mappings to other similar mappings and verify that the systems and services the company uses are covered by the mapping. [51]

One example of requirements or recommendations from both CIS and PCI DSS that are the same is encryption of data at rest. These requirements are as an example compared to each other in Table 2.

Table 2 Example of Compliance Mapping between CIS and PCI DSS [16], [18]

CIS	**PCI DSS**
CIS Control: 14.8	PCI: 3.4, 3.4.1
- 14.8 Encrypt Sensitive Information at Rest. Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.	- 3.4 Render PAN unreadable anywhere it is stored by using any of the following approaches: One-way hashes based on strong cryptography, Truncation, Index tokens and pads or Strong cryptography with associated key-management processes and procedures. - 3.4.1 If disk encryption is used, logical access must be managed separately and independently of native operating system authentication and access control mechanisms. Decryption keys must not be associated with user accounts.

5.5.1. Mapping CIS controls to PCI-DSS in AWS

As discussed in chapter 5.2.1 AWS tries to help the customers in many ways, this is also for compliance mappings. If the customer chooses to use AWS Security Hub the documentation and console interface help with mapping different compliance requirements to the compliance checks that the service use. In the console, the compliance requirements information is displayed for CIS recommendations in the title and for PCI DSS in the related requirements tab as seen in Figure 17 or in the findings tab when looking at a specific finding.

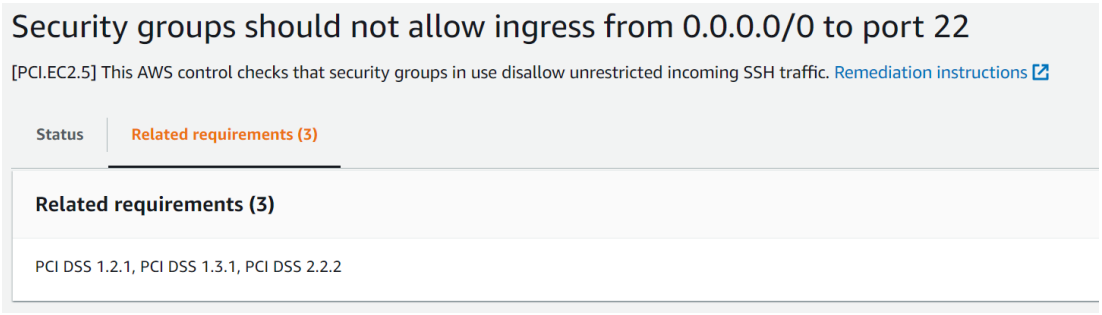


Figure 17 AWS Security Hub related requirements example

The documentation for the PCI DSS compliance checks also gives a bit more information about related requirements and the relevant services connected to these requirements. For developers, there are possibilities to analyze the ASFF information that AWS Security Hub creates. The *Compliance* tag the *RelatedRequirements* contain various requirements related to that specific compliance check, as seen in Figure 18.[25]

```
"Compliance": {
    "RelatedRequirements": ["Req1", "Req2"],
    "Status": "FAILED",
    "StatusReasons": [
        {
            "ReasonCode": "CLOUDWATCH_ALARMS_NOT_PRESENT",
            "Description": "CloudWatch alarms do not exist in the account"
        }
    ]
}
```

Figure 18 Example of ASFF for Compliance Finding [25]

These requirements are directly checked by the compliance checks done in AWS Security Hub. However, there can be indirect mappings between the requirements, especially more complex requirements that involve many services.

Mapping between different compliance frameworks or recommendations is not available in the console or in the developer resources directly. This leaves the customer to handle the actual mappings, but there are many tools and mappings available, as discussed in the previous sections. Also, a part of the compliance process in a company is to map the compliance requirements to internal processes and this mapping often can extend to map compliance requirements for other systems. In Table 3, an example of mapping the different compliance checks to different compliance frameworks is shown.

Table 3 Example of Mapping requirements between CIS and PCI DSS in AWS [25], [48]

CIS AWS Foundations Benchmark	**PCI DSS AWS**
[S3.4] S3 buckets should have server-side encryption enabled (2.1.1 Ensure all S3 buckets employ encryption-at-rest)	[PCI.S3.4] S3 buckets should have server-side encryption enabled
Related recommendation: CIS Control 14.8 Encrypt Sensitive Information at Rest	Related requirement: PCI DSS 3.4: Render Primary Account Numbers (PAN) unreadable anywhere it is stored

6. Proof of concept: Continuous Compliance Automation in AWS

The proof of concept for the Continuous Compliance Automation architecture implemented is influenced by currently used methods and techniques as well as the chosen cloud provider by the company this thesis was done for. As discussed previously in this thesis, AWS has many services that can help achieve specific goals in the AWS environment. The proof of concept tries to take advantage of the already existing solutions that AWS manages for minimizing the development needed for implementing such a solution. AWS managed services and solutions that are available directly when activating these services are maintained by AWS and can often be modified or customized to their own preferences with a small amount of work. Therefore, the standard solution is used as a base for implementing the proof of concept. The ability to use already existing solutions reduced the development scope drastically, but the possibility of creating similar solutions that can be customized further is still there and supported by the implemented architecture with only minor changes. The drawbacks are that there may be breaking changes and updates by AWS need to be monitored for these breaking changes.

6.1. Continuous Compliance Automation Architecture, a brief overview

The architecture that was chosen is a flexible solution that can be modified and customized as needed. However, the overall functionality is simple, with some more complex services to handle exceptions and exceptional cases. Some services can feel excessive, but the benefit of using them can be justified by the negative complexity or cost they create. In Figure 19 the architecture is shown in a diagram that explains the data flow as well as the different AWS services used. By having the diagram as support, this section tries to give a brief overview without going into more details. The details are discussed later in this chapter, with code snippets as support. [1]

[1] All code can be found at https://github.com/ijansson/masters-continuous-compliance-automation

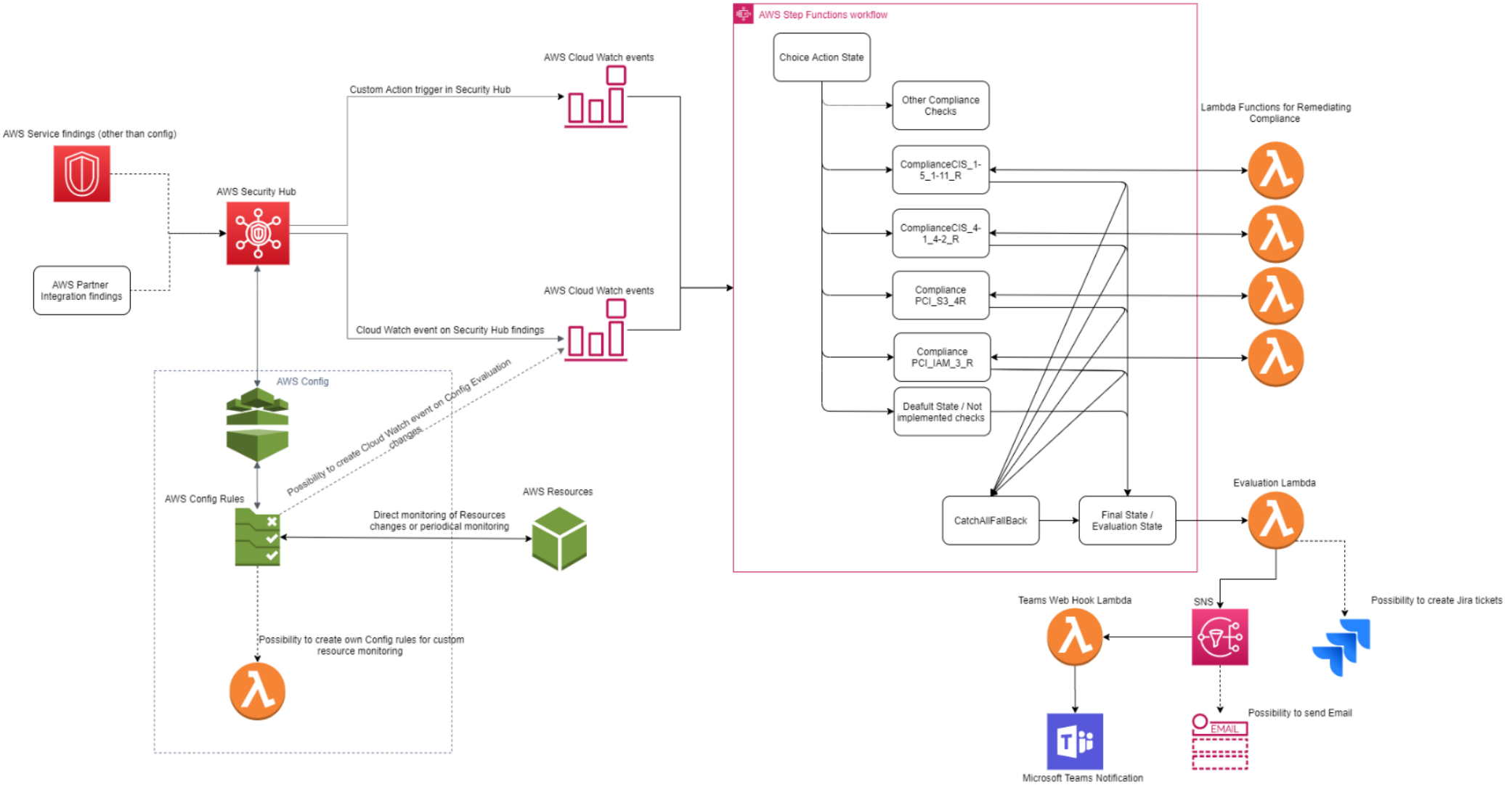


Figure 19 Continuous Compliance Automation Architecture

The solution starts with the service AWS Security Hub that is used as a CCM tool for the environment of the account. Security Hub uses the CIS benchmarks and the PCI DSS compliance checks that are enabled by the code as a base for the compliance checks that this solution monitors. Security Hub uses AWS Config for finding resource changes or Config rule changes, Security Hub sets up Config rules that correspond with the included checks for CIS and PCI. The rules monitor resource changes live on events or periodically and determine if they are compliant or non-compliant. The results of these rules are then ingested into Security Hub.

Security Hub displays and analyses the findings as well as marks priority of the findings to highlight what compliance problems that should be fixed first easily. Security Hub also displays other findings from other AWS services like GuardDuty or Inspector and integrated partner solutions that can be enabled if these services are already used. All results are then shown as Security Hub findings; the findings have labels for different levels of severity, resource type, and a specific resource.

An AWS CloudWatch Event Rule is then used to trigger on specific Security Hub findings, filtering out the FAILED compliance checks and the monitored rules that are defined in the event. In this case there is only one CloudWatch Event rule that contains all the compliance checks and one CloudWatch Event rule that handles unsupported events triggered by Custom Action in Security Hub. The Custom Action is activated manually by sending the finding to the event rule.

The CloudWatch Event rule sends the Security Hub finding to an AWS Step Function that contains a State Machine. The State Machine has different states and uses server-less functions AWS Lambdas to execute different task states. The Step Function State Machine is written in Amazon states language. The first state is a Choice state where the different monitored rules are separated into groups ending up in different states.

The chosen state based on the compared rule triggers a Task state that calls a Lambda function with the Security Hub finding as an event input. The Lambda function then tries to remediate and correct the compliance issue. If the function returns without errors or controlled errors, it enters an Evaluation or Final state. If the Lambda fails with unexpected errors, the *CatchAllFallBack* state handles the exception and sends the event with a controlled error message to the Evaluation state. If no matching state can be found in the Choice states, it sends the event to a default state where the event

is forwarded with a flag to the Evaluation state. The Evaluation state then decides what to do with the result, either error or corrected compliance checks. The corrected and failed corrections are sent to an SNS topic that delivers the message to a Lambda with a webhook into a Microsoft Teams channel of choice. This also happens to non-supported events with a different kind of message. There are endless possibilities for creating tickets into a ticketing system on unsupported events or failed remediation's or sending Emails to mail recipients.

6.2. Implementation of Continuous Software Engineering concept

The goal with the proof of concept is to use the CSE concept regarding code automation as much as possible. Whenever the solution is used, full code automation can be implemented. As soon as a version control system like Git is used in cooperation with a pipeline that supports Ansible, the environment can be provisioned and created with a CI/CD process. The architecture and code for the AWS environment implemented is then created by the CloudFormation templates. This leads to complete automation of code deployment in the different environments managed by the ansible framework and complete version control solution. Everything can be created and updated with code changes, which, in turn, will lead to a more stable environment that does not need human intervention when the environment architecture is built. The possibilities to monitor changes made to the environment and the possibility to roll back changes if they do not work as intended introduces a significant improvement over manual change and configurations.

6.2.1. Infrastructure implementations with Ansible and CloudFormation

To enable code automation, Ansible and the Ansible module "**cloudformation:"** are used. In this section, the structure of the project and the use of the Ansible framework for configuration management as well as the use of CloudFormation for infrastructure management, are shown. The Ansible playbooks and role structure enable complex configuration and management of multiple resources as well as multiple

configurations for multiple AWS accounts. As discussed earlier in chapter 3.3 and Figure 9, the roles often have the same file structure as the Ansible best practices example. The proof of concept tries to follow these recommendations. The *\site.yml* is the master playbook that includes other playbooks. The included playbooks are often more complex tasks that are divided into roles. The included playbooks in a role are often located in a folder with the name of the role, *roles\role_name* as seen in Figure 20.

```
- name: Configure the AWS securityhub and AWS config
  hosts: localhost
  connection: local
  gather_facts: false

  roles:
    - configure-securityhub
```

Figure 20 role_name the included task that is referenced from the master playbook

The *configure-securityhub* role is located at *roles\role_name\tasks\main.yml* seen in Figure 21 the code is executed with the variables stored in a file accessible for all roles, in this case, information about the different AWS *{{accounts}}* that are used. This information is sensitive information and is not shown in any figures. The *included_tasks* variable in Figure 21 is a list of included tasks that need to be executed for each account.

```
- name: Configure default security hub document properties
  include_tasks: configure_securityhub.yaml
  with_items: "{{ accounts }}"
  loop_control:
    loop_var: account_item
```

Figure 21 roles\role_name\tasks\main.yml

Additional variables that are specific to the role can be included in the additional file *\roles\role_name\vars\main.yml* as seen in Figure 22 and these are included in the *roles\role_name\tasks* namespace for easy use in that role. The passed variables from other files can also be used as long as they are located in the task's namespace and changed if needed.

```
region: "eu-west-1"
s3BucketName: "{{ account_item.account_name }}-{{ region }}-config"
kmsKeyId: "alias/{{ account_item.account_name }}-config-KMSKey"
SNSTopicName: "{{ account_item.account_name }}-config-sns-topic"
```

Figure 22 roles\role_name\vars\main.yml

The tasks themselves can include different ansible modules; in this case, the "**cloudformation:"** module is used for creating and updating an AWS CloudFormation stack. The stack information and all variables needed to create the CloudFormation files are passed in the tasks file *configure-securityhub.yaml*. In the task *roles\role_name\tasks\role_name*(*configure-securityhub.yaml*), which is the main task in the role as seen in Figure 23, a final file *configure_config.yaml* with a template for the CloudFormation code is included.

```
- name: Configure AWS Config CF-stack for all accounts
  cloudformation:
    profile: "{{ account_item.account_name }}"
    stack_name: "{{ account_item.account_name }}-config-cf"
    state: "present"
    region: "{{ account_item.default_region | default('eu-west-1') }}"
    disable_rollback: true
    template: "roles/configure-securityhub/files/configure_config.yaml"
    template_parameters:
      AccountName: "{{ account_item.account_name }}"
      EnvironmentPrefix: ""
      S3BucketName: "{{ s3BucketName }}"
      KmsKeyId: "{{ kmsKeyId }}"
      SNSTopicName: "{{ SNSTopicName }}"
    tags:
      Account: "{{ account_item.account_name }}"
      Stack: "{{ account_item.account_name }}-config-cf"
      Contact: "{{ account_item.owner_email }}"
      CostCenter: "{{ account_item.owner_costcenter }}"
  register: register_config
```

Figure 23 roles\role_name\tasks\role_name the *configure-securityhub.yaml* file

The file in *roles\role_name\files\template_name* takes the variables as previously defined as template parameters, enabling usage of the parameters in the CloudFormation code. When a playbook is executed, Ansible uploads the CloudFormation service through Boto3 to an AWS CloudFormation stack that is then

deployed in the AWS environment that configures the resources defined in the template.

All code and structure of the Ansible framework are then version controlled by using some form of version control system. When the code is committed, builds are triggered in a build system, such as Jenkins or Ansible Tower, which are automation servers that can handle CI processes or complete continuous delivery processes.

6.2.2. Enabling Continuous Compliance Monitoring

Several services need to be enabled and configured to implement the continuous compliance architecture in AWS, as has been discussed previously. Among those is Security Hub that also needs the service Config enabled and configured. To configure and enable these services an ansible role *configure_securityhub* was created as well as a subtask to enable Config using a CloudFormation stack. The Ansible playbooks for setting up the *configure_securityhub* role are shown in Figure 20, Figure 21, Figure 22 and Figure 23.

The prerequisite to enabling Security Hub is to have Config enabled and configured. Therefore the first task in the playbook is configuring AWS Config as shown in Figure 23. The CloudFormation template used *configure_config.yaml* uses variables previously defined in the playbooks as parameters and configures different AWS services to be able to activate the Config service. The components needed for enabling Config are an S3 bucket, SNS topic, Config Delivery Channel, Config Configuration Recorder, a KMS key for encryption, an IAM role and the required policies. More about how these services are configured can be found in the code.

After the configuration of the Config service, enabling Security Hub is done using shell command from the AWS CLI as seen in the Figure 24. When enabling Security Hub from the API, the security standards are automatically enabled by default.

```
- name: Enable securityhub
  shell:  aws securityhub enable-security-hub  --profile "{{ account_item.account_name  }}"
  register: securityhub_output
  ignore_errors: True
```

Figure 24 Shell Command for activating Security Hub

In the console, the results of the compliance evaluations can then be monitored and resources that fail compliance evaluations can be found using the Security Hub findings.

6.3. Compliance Automation and Remediation

To create an automated solution that does not require human intervention every time a compliance problem occurs is the key. AWS resources constantly change and the possibilities to create new resources, like servers in a couple of minutes, make the manual corrections of compliance and security problems impossible for a human being to comprehend. The fast-paced environment and constant change generate endless compliance and security problems that may be critical to fix for the environment to remain compliant.

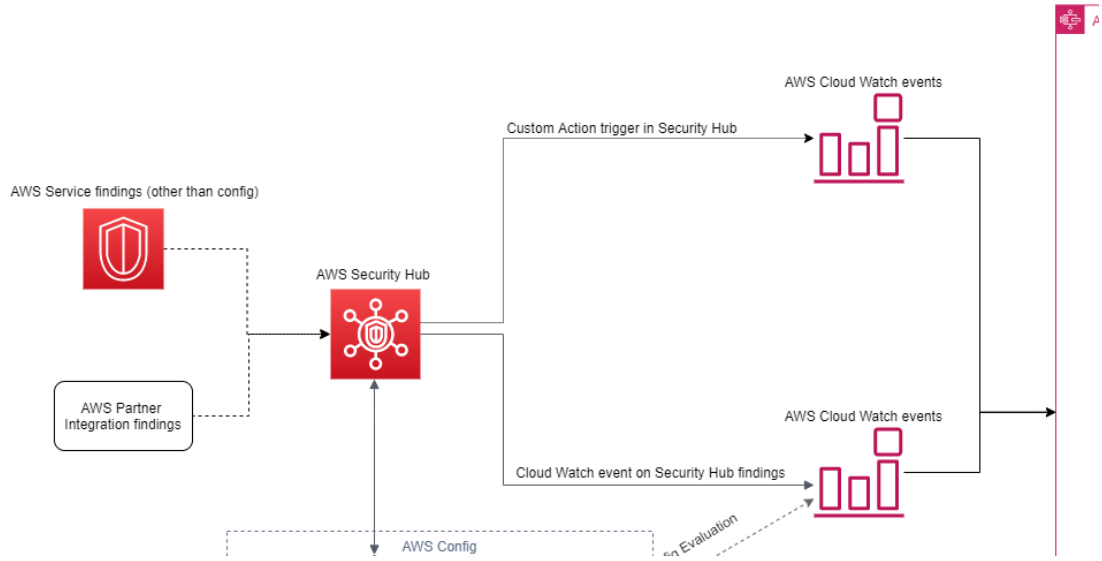


Figure 25 CloudWatch Events on Security Hub Findings

Using the CloudWatch service the Security Hub findings can be captured by creating a CloudWatch Event rule as seen in Figure 25 (the figure is a subset of Figure 19). The CloudWatch event rule defines an action to take automatically when a finding is imported or updated by Security Hub. The CloudWatch Event rule can be triggered by every finding or triggered by only specified event patterns in the findings. By specifying an event pattern, a filter can be created to catch only desired Security Hub findings by for example specific failed compliance requirements. The event pattern is written in JSON format, but CloudFormation takes care of the conversion from the Template code written in YAML format. The “GeneratorId” is unique for every rule

that Security Hub checks and can therefore be used to filter out the specific events that the solution can handle. The CloudFormation code for creating the CloudWatch event rule is shown in Figure 26.

```
ContinuousComplianceAutomationEventRule:
  Type: AWS::Events::Rule
  Properties:
    Name: ContinuousComplianceAutomation_CWER
    Description: "Continuous Compliance Automation for CIS and PCI events from Security hub to Stepfunction
    EventPattern:
      source:
        - aws.securityhub
      detail-type:
        - Security Hub Findings - Imported
      detail:
        findings:
          RecordState:
            - ACTIVE
          Compliance:
            Status:
              - FAILED
          GeneratorId:
              - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.5
              - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.6
              - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.7
              - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.8
              - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.9
              - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.10
              - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.11
              - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/4.1
              - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/4.2
              - pci-dss/v/3.2.1/PCI.S3.4
              - pci-dss/v/3.2.1/PCI.IAM.3
    State: "ENABLED"
    Targets:
      -
        Arn: !Ref ContinuousComplianceAutomationStateMachine
        RoleArn: !GetAtt
          - ContinuousComplianceAutomationExecuteStateMachineRole
          - Arn
        Id: "ContinuousComplianceAutomation_CWE"
```

Figure 26 CloudWatch Event Rule for catching specific Security Hub findings based on Compliance requirements

The CloudWatch event rule allows for forwarding the event, in this case the findings to an action target. The action target can be any resource invoked by events, like Lambdas, Step Functions, Batch Jobs and much more. The events in this solution are each forwarded to one AWS StepFunction State Machine *ContinuousComplianceAutomationStateMachine* that handles the event, as seen in the code for the CloudWatch event in Figure 26. The event JSON itself can also be modified in the CloudWatch event rule to filter out the desired information. However, in this solution the default Security Hub event is forwarded.

The Security Hub event JSON contains information about the breached compliance requirement and available information about the resource. Some resource information is limited, but for most events the resource information is enough, AWS constantly improves the resource information.

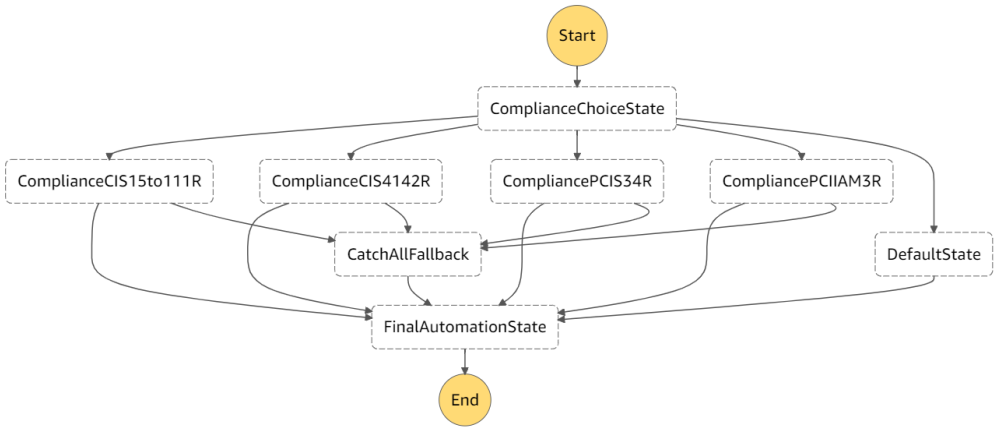


Figure 27 Compliance Automation Step Function State machine

The *ContinuousComplianceAutomationStateMachine* StepFunction is written in Amazone States Language. Still, as with other code it can be included in the CloudFormation template as inline code and Lambda functions created in the same CloudFormation stack can easily be referenced for the state machine to use. The purpose of the step function is to separate the different compliance requirements based on *GeneratorId* and forward the finding to a Lambda for remediation. A visualization of the StepFunction and its architecture is shown in Figure 27. The separation between different compliance events is done in a *Choices* step as seen in Figure 28 where choices are made on the *GeneratorId*. The choices then reference a next step to be executed with the key value *"Next"*: "*ComplianceCIS4142R"* that is called when a matching string is found, as seen in Figure 29. In this case the next step is a *"Type": "Task"* step and the task itself has a reference to the actual resource, in this case, a Lambda *Compliance_CIS_41_42_Remediation* as seen in Figure 30.

```
{
  "Comment": "Compliance Automation choosing between compliance remediations",
  "StartAt": "ComplianceChoiceState",
  "States": {
    "ComplianceChoiceState": {
      "Type": "Choice",
      "Choices": [
        {
          "Or": [
            { "Variable": "$.detail.findings[0].GeneratorId",
              "StringEquals": "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.5"
            },
            { "Variable": "$.detail.findings[0].GeneratorId",
              "StringEquals": "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.6"
            },
```

Figure 28 Compliance Choice State

```
{
  "Or": [
    {
      "Variable": "$.detail.findings[0].GeneratorId",
      "StringEquals": "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/4.1"
    },
    {
      "Variable": "$.detail.findings[0].GeneratorId",
      "StringEquals": "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/4.2"
    }
  ],
  "Next": "ComplianceCIS4142R"
},
```

Figure 29 Compliance Choice State with reference to Next task

```
  "ComplianceCIS4142R": {
    "Type": "Task",
    "Resource": "arn:aws:lambda:eu-north-1:[redacted]:function:Compliance_CIS_4-1_4-2_Remediation",
    "Next": "FinalAutomationState",
      "Catch": [{
        "ErrorEquals": ["States.ALL"],
          "Next": "CatchAllFallback",
          "ResultPath": "$.error"
      }]
  },
```

Figure 30 Task State with reference to Lambda resource in the console after creation

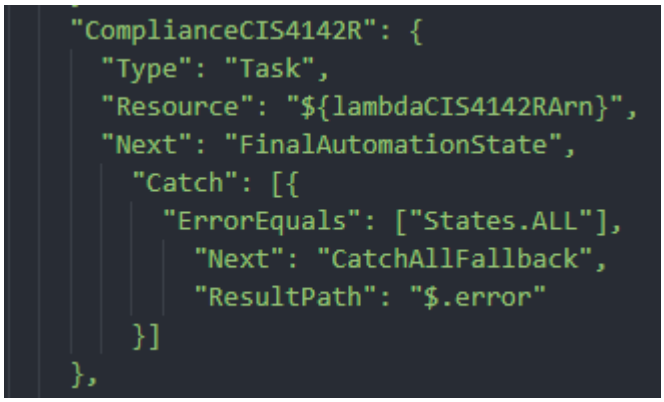


Figure 31 Task State with reference to Lambda resource in CloudFormation stack

The CloudFormation code shown in Figure 31 and be compared to the StepFunction code in the console seen in Figure 30. The *Task* itself in CloudFormation is referenced with *lambdaCIS4142RArn*, an alias for the Compliance_CIS_41_42_Remediation Lambda function executed when the step is triggered. The task also is configured to handle uncontained errors experienced in the Lambda and forwards the results of the Lambda execution to the next step that in this case is the "*FinalAutomationState*". The architecture of the StepFunction State Machine *ContinuousComplianceAutomationStateMachine* can be seen in Figure 32.

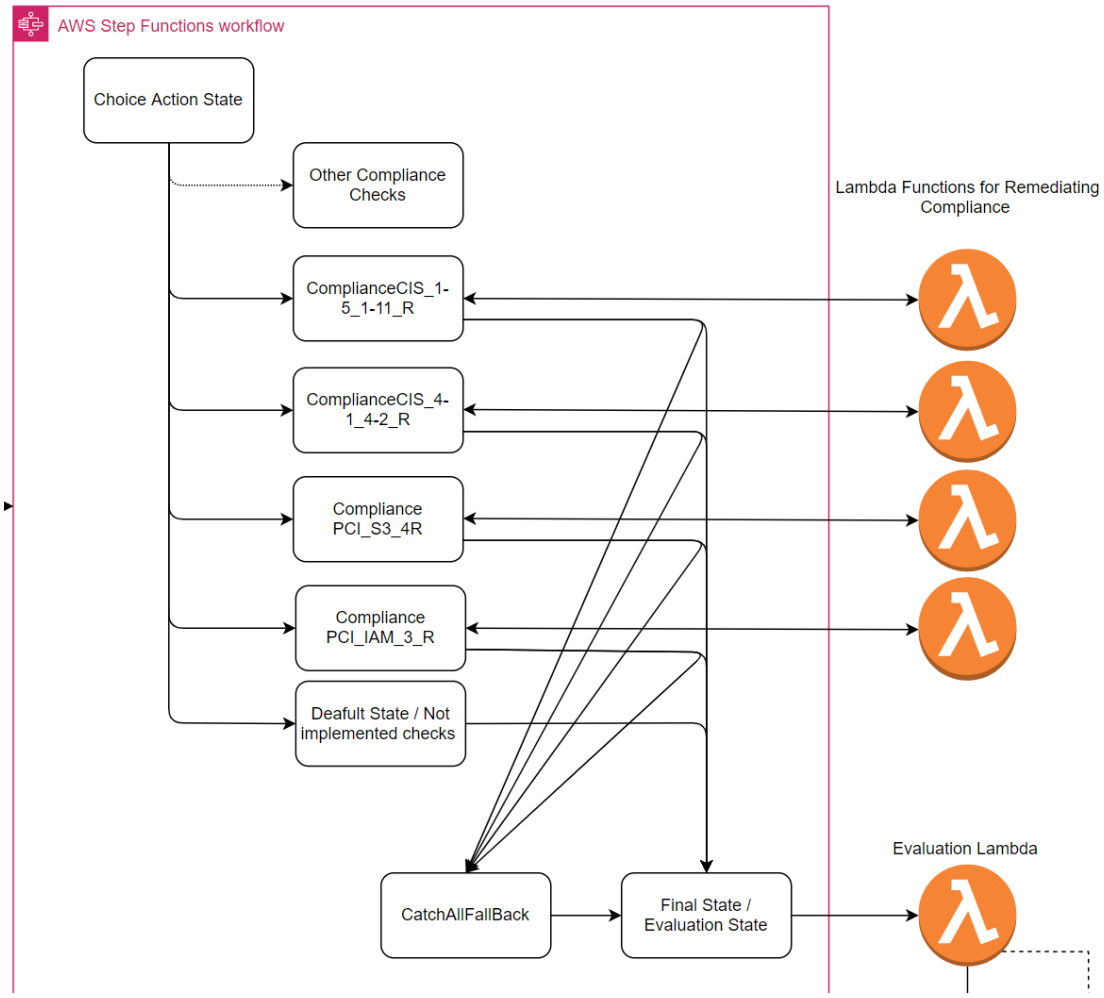


Figure 32 Architecture of the ContinuousComplianceAutomationStateMachine

An execution flow through the compliance architecture up to the execution of the remediation Lambda can be as follows. Security Hub has found a non-compliant resource that triggers the compliance rule cis-aws-foundations-benchmark/v/1.2.0/rule/4.1 that states “Security groups provide stateful filtering of ingress/egress network traffic to AWS resources. It is recommended that no security group allows unrestricted ingress access to port 22.”. The non-compliant resource is a Security Group for EC2 that allows unrestricted ingress to that instance. The CloudWatch event rule picks up the Security Hub finding and forwards the finding to the State Machine. The Choice State filters out the specific rule that was triggered, as shown in Figure 29 and calls the next task in Figure 30. The triggered tasks start executing the Compliance_CIS_4-1_4-2_Remediation Lambda that it has as a resource, as seen in Figure 30.

6.3.1. Remediation of Compliance events

The task of finding a non-compliant resource or configuration is handled automatically in this case by Security Hub. Often CCM solutions only inform about the compliance events. However, by also trying to remediate or fix the non-compliant resources, we can automate this process further. By using Lambdas that can execute code on demand and the extensively available AWS SDKs/APIs to access non-compliant resources.

Remediation

To continue from the example in the previous section the *Compliance_CIS_4-1_4-2_Remediation* Lambda handles remediation of CIS recommendations 4.1 and 4.2. The Lambda code starts automatically with the function *lambda_handler(event, context)* that gets the *event* from the Security Hub finding. The event is a JSON object that contains all information collected by Security Hub. The functions extract the needed information from the *event* into more manageable variables, as seen in Figure 33.

```
def lambda_handler(event, context):
    print('Event: {}'.format(event))
    if 'Details' in event['detail']['findings'][0]['Resources'][0].keys():
        det = event['detail']['findings'][0]['Resources'][0]['Details']
        nonCompliantSgId = str(det['AwsEc2SecurityGroup']['GroupId'])
        nonCompliantSgIdName = str(det['AwsEc2SecurityGroup']['GroupName'])
        findingId = str(event['detail']['findings'][0]['Id'])
        region = str(event['detail']['findings'][0]['Resources'][0]['Region'])
        lambdaFunctionName = os.environ['AWS_LAMBDA_FUNCTION_NAME']
        result = ComplianceResult.NOT_CORRECTABLE
        result,result_msg = fix_security_group(region, nonCompliantSgId,nonCompliantSgIdName)
```

Figure 33 Compliance_CIS_4-1_4-2_Remediation: Extracting Finding information and call fix_security_group() function

```
def fix_security_group(region, nonCompliantSgId,nonCompliantSgIdName):
    aws_session = boto3.Session(region_name=region)
    ec2 = aws_session.resource('ec2')

    print ('Attempting to create compliance...')
    sg = ec2.SecurityGroup(nonCompliantSgId)
    result = ComplianceResult.NOT_CORRECTABLE
    result_msg = 'No port found, ' + nonCompliantSgId
```

Figure 34 Compliance_CIS_4-1_4-2_Remediation: fix_security_group() AWS Boto3 SDK to access resources and configurations

```
port = rule['ToPort']
if port in FORBIDDEN_PORTS:
    sg.revoke_ingress(IpProtocol=rule['IpProtocol'], FromPort=port, ToPort=port, CidrIp='0.0.0.0/0')
    time.sleep(1)
    result = ComplianceResult.CORRECTED
    result_msg = 'Non compliant security group removed '+nonCompliantSgId +' '+nonCompliantSgIdName
```

Figure 35 Compliance_CIS_4-1_4-2_Remediation: fix_security_group() Revoke ingress for non-compliant resource

To remediate the compliance problem the needed information is forwarded to a *fix_security_group()*, as seen in Figure 33. The *fix_security_group()* function uses the Boto3 python AWS SDK to access resources and information about the resource, as seen in Figure 34. Using the acquired security group resource, we can try to revoke ingress for that specific inbound rule. This is done by calling the *revoke_ingress()* function in the Boto3 SDK with parameters about the specific rule to revoke, as seen in Figure 35. If the revoke is unsuccessful, an error is thrown and caught by the *fix_security_group()* error handling.

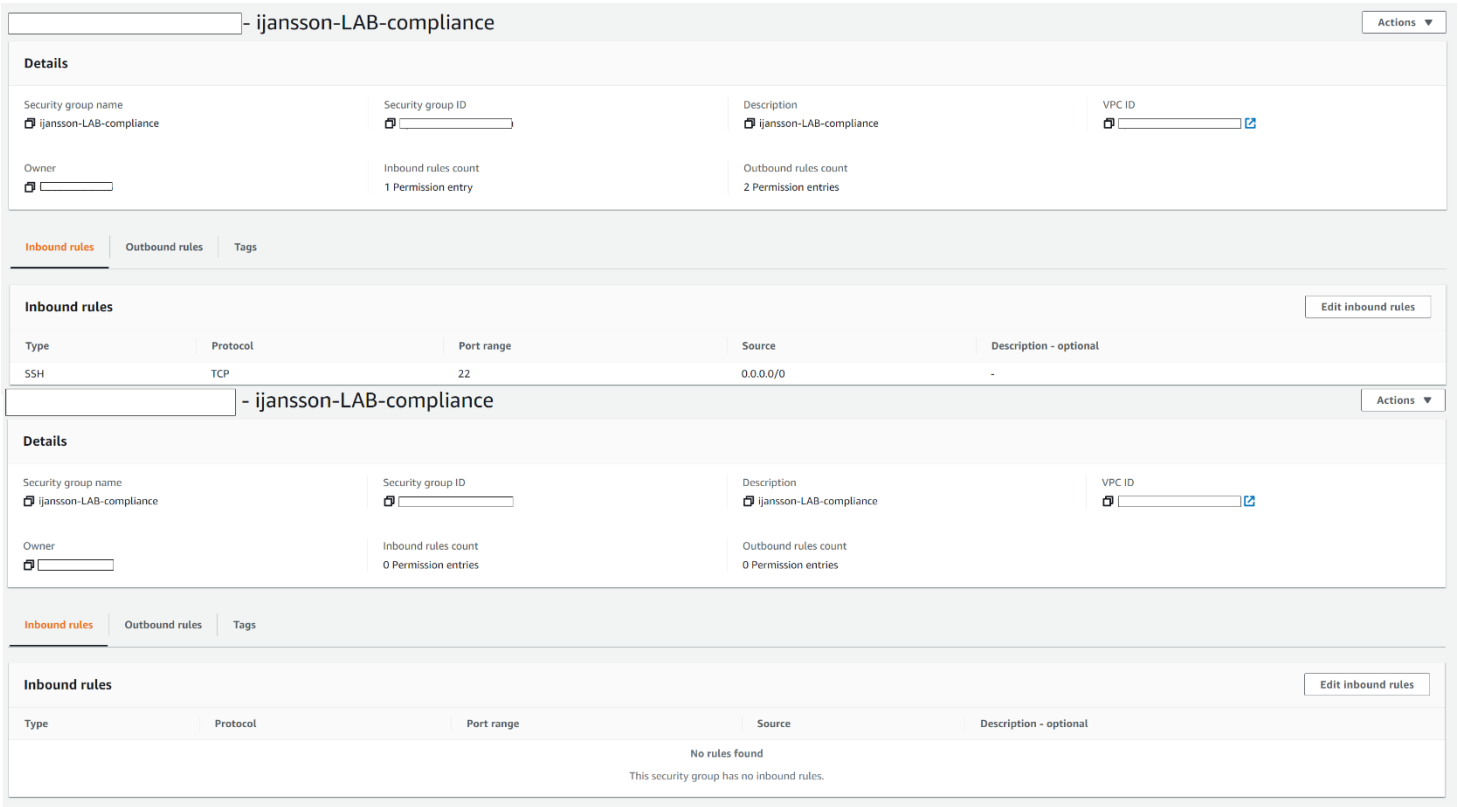


Figure 36 Security Group before and after revoked ingress in the console

For the user, the Security Group is removed from the list of inbound rules for the Security Group in the console as seen in Figure 30, where the Security Group on top contains an inbound rule and at the bottom, the remediation Lambda has removed the inbound rule after execution.

Remediation notification process

To be able to inform the user of the invoked remediation event and the result of the remediation the Security Hub finding sent to Lambda is updated with a note text that describes the event and the result of the event. This is done by using the *update_finding()* function in the Boto3 SDK as seen in Figure 37 and the note text in the Security Hub finding is updated as seen in Figure 38.

```
if result == ComplianceResult.CORRECTED:
    try:
        securityhub = boto3.client('securityhub')
        response = securityhub.update_findings(Filters={'Id': [{'Value': findingId,'Comparison': 'EQUALS'}]
            },Note={'Text': 'Automatic Remidiation was invoked. Refer to Automation results to determine efficacy:' + result_msg,
            'UpdatedBy': lambdaFunctionName},RecordState='ACTIVE')
```

Figure 37 Compliance_CIS_4-1_4-2_Remediation: Update Security Hub finding with note text

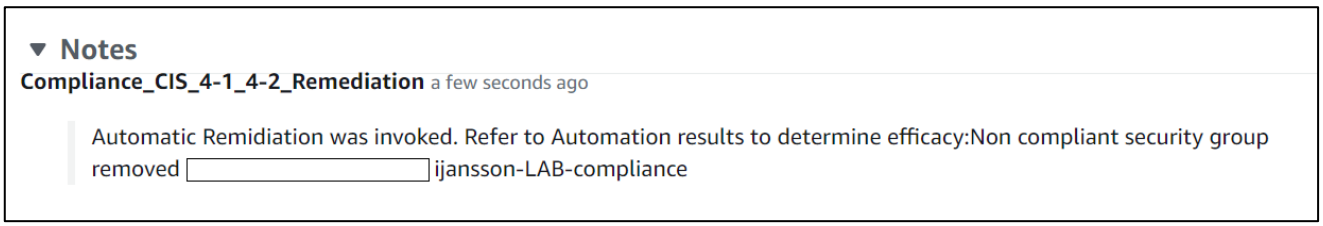


Figure 38 Security Hub finding note text of remediation attempt

The *Compliance_CIS_4-1_4-2_Remediation* Lambda function then collects information about the execution and formats the information into a JSON object that can be forwarded to other tasks in the StepFunction State machine. This using a standardized format that all similar Lambda functions follow, as seen in Figure 39.

```
statusCode = 200
if result == ComplianceResult.ERROR:
    statusCode = 500
return {
    'statusCode': statusCode,
    'body': {
        "function":lambdaFunctionName,
        "result": ComplianceResult.result_to_str(result),
        "message": result_msg,
        "event": event
    }
}
```

Figure 39 Compliance_CIS_4-1_4-2_Remediation: Information collection and return result

```
"DefaultState": {
  "Type": "Pass",
  "Result":{
    "Deafult": "Yes"
  },
  "ResultPath": "$.deafult",
  "Next": "FinalAutomationState"
},
"CatchAllFallback": {
    "Type": "Pass",
    "Next": "FinalAutomationState"
},
"FinalAutomationState": {
"Type": "Task",
"Resource":  "${lambdaComplianceAutomationFinalStateArn}",
"End": true
}
```

Figure 40 State Machine FinalAutomationState Task

When the *Compliance_CIS_4-1_4-2_Remediation* Lambda is done the StepFunction starts the next step referenced in the Task, this is the *FinalAutomationState* as seen previously in Figure 31. The *FinalAutomationState* Task forwards the JSON object from the previous Lambda into the next Lambda *lambdaComplianceAutomationFinalStateArn* for processing, as seen in Figure 40. The *ComplianceAutomationFinalState* Lambda is a final handler of the Compliance Automation Flow that determines what to do with the results of the compliance event and the executed remediation.

The whole flow of a corrected compliance event can be visualized in the console execution details shown in Figure 41. The execution details for the StepFunction in this example show a correction of a compliance event regarding CIS recommendation 4.1 triggered by a Security Hub finding as previously discussed. Triggering the referenced ComplianceCIS4142R Lambda.

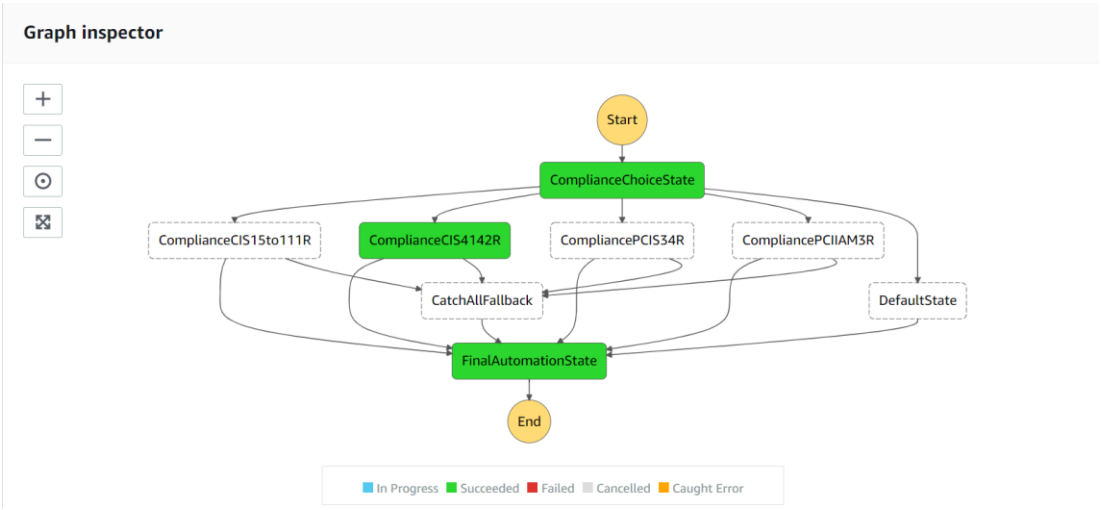


Figure 41 Step Function State machine flow of Corrected Compliance event for CIS recommendation 4.1

```
elif 'statusCode' in event.keys():
    function = event['body']['function']
    result = event['body']['result']
    if event['statusCode'] == 200:
        result_msg = event['body']['message']# the compliance could be corrected or is non correctable
    elif event['statusCode'] == 500 :
        result_msg = event['body']['error']# unexpected but handled error

    rule = str(event['body']['event']['detail']['findings'][0]['GeneratorId'])
    region  = str(event['body']['event']['region'])
    url =  "https://"+region+".console.aws.amazon.com/securityhub/home?region="+region+"#/findings"

    message = {
        'summary': "Automated Compliance",
        'title': "Compliance " + result + " by " + function + ". For account "+ alias +" in region "+ region,
        'text': result_msg + ". For rule " + rule,
        'urlName': "AWS Security Hub findings" ,
        'url': url
    }
    if result != "CORRECTED":
        message.update({'status': 'Alarm'})
    else:
        message.update({'status': 'No'})

    publish_compliance_sns(message)
    return message
```

Figure 42 ComplianceAutomationFinalState Collecting information and forward to publish_compliance_sns()

The *ComplianceAutomationFinalState* can be configured and coded in many ways. The possibilities are many. In this proof of concept, the Lambda code extracts the relevant information from the compliance events and compliance remediation results. Then publish the collected information as a message on an SNS queue for notifying the concerned parties. In *ComplianceAutomationFinalState,* as seen in Figure 42 the information is extracted and a message JSON is created with the relevant information needed for creating a notification.

```python
def publish_compliance_sns(message):
    sns = boto3.client('sns')
    a = sns.list_topics()
    dicts = a['Topics']
    arn = next((item for item in dicts if (item["TopicArn"]).find("sns-teams-compliance-topic") != -1), None)
    if arn is not None:
        try:
            response = sns.publish(
                TargetArn=arn['TopicArn'],
                Message=json.dumps({'default': json.dumps(message)}),
                MessageStructure='json'
            )
```

Figure 43 ComplianceAutomationFinalState: publish_compliance_sns() function for publishing messages on SNS queue

The message is then published on an asynchronous SNS message queue on a specific topic *sns-team-compliance-topic* that any subscriber can listen to, as seen in Figure 43. The message queue, in this case is a specific topic created in the CloudFormation stack as seen in Figure 44. The subscribers can be one of many, for example, Email, mobile push notification, and other AWS services. One of those services is AWS Lambdas that also can be assigned as a subscriber for these kinds of events.

```yaml
SNSTeamsComplianceTopic:
  Type: 'AWS::SNS::Topic'
  Properties:
    DisplayName: Teams Compliance Topic
    TopicName: sns-teams-compliance-topic
    Subscription:
      - Endpoint: !GetAtt [ComplianceTeamsWebHookLambdaFunction, Arn]
        Protocol: lambda
  DependsOn: [ComplianceTeamsWebHookLambdaFunction]
```

Figure 44 CloudFormation Configuration of SNS queue and specific subscriber endpoint

The Lambda registered as a Subscriber to the SNS topic is referenced in the CloudFormation template and will listen to messages sent over the specified topic. Once a message is published to the topic, the Lambda code executes with the message as a parameter. The Lambda then contains code for sending compliance events as chat messages to a Microsoft Teams channel. The chat messages are injected into Teams through a webhook that listens for messages sent to a specific address. In the *ComplianceTeamsWebHook* Lambda, the SNS message from the *ComplianceAutomationFinalState* is extracted and formatted into a message object that the Teams webhook can handle, as seen in Figure 45.

```
message = {
  "@context": "https://schema.org/extensions",
  "@type": "MessageCard",
  "summary": summary,
  "themeColor": data["colour"],
  "title": title,
  "text": text,
  "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "Log in to AWS Console",
            "targets": [{
                    "os": "default",
                    "uri": "[redacted]"}]
        },
        {
            "@type": "OpenUri",
            "name": urlName,
            "targets": [{
                    "os": "default",
                    "uri": str(url)}]
        }
    ]
}
req = Request(TEAMS_HOOK_URL, json.dumps(message).encode('utf-8'))
```

Figure 45 ComplianceTeamsWebHook Lambdafunction, creation of Teams chat message

The *TEAMS_HOOK_URL* variable is set as an environment variable for the Lambda and can be changed manually in the console by editing the environment variables for the Lambda *ComplianceTeamsWebHook*.

The Teams channel members are then notified about the compliance event that either was corrected or could not be corrected by the continuous compliance automation flow according to the implemented compliance requirements. A notification message for a corrected compliance problem for the previously discussed example CIS benchmark 4.1 is shown in Figure 46.

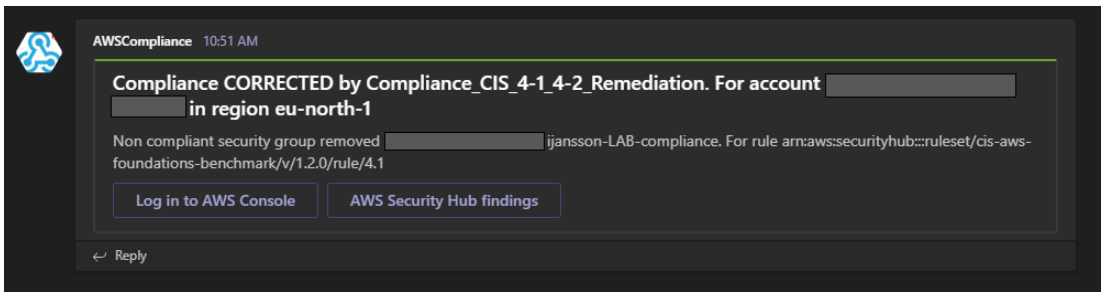


Figure 46 Teams message about Corrected Compliance event

If the remediation fails or the continuous compliance automation flow encounters any problems during execution, a notification message about the failed event is created. An example of this is a security group that is not part of the proof-of-concept example shown in Figure 47.

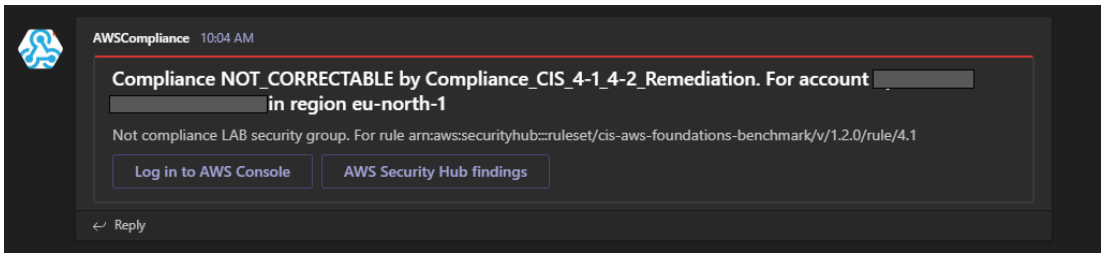


Figure 47 Teams message about Non-Correctable Compliance event

6.3.2. Implemented Compliance Remediation

The compliance remediations implemented are few due to the nature of the proof of concept, focusing on the architecture and automation of the entire process. But with the implemented remediations, this proof of concept shows that different resources and configurations can be remediated by code and automated almost completely. Fixing the compliance requirements or recommendations without involving human intervention.

The first remediation focuses on network related requirements with restricting ingress or egress to specific ports in a security group. This example was discussed earlier in section 6.3.1 as an example of the flow through the whole architecture. The overall structure of the code in all remediations follows the previous example with only minor deviations and changes in the way remediations are done. The critical part of the code in this example is revoking the ingress or egress to specific ports, as shown in Figure 35. This is done in the *Compliance_CIS_4-1_4-2_Remediation* lambda. This requirement is covered by both CIS recommendations and PCI DSS requirements as shown in Table 4 that contains the compliance mapping done for the proof of concept.

Table 4 Implemented compliance requirements and mappings

Compliance requirements	AWS Security Hub Title	AWS Security Hub rule
CIS AWS Foundations 4.1, 4.2 PCI DSS 1.2.1, 1.3.1, 2.2.2	[CIS.4.1] Ensure no security groups allow ingress from 0.0.0.0/0 to port 22 [CIS.4.2] Ensure no security groups allow ingress from 0.0.0.0/0 to port 3389	- arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/4.1 - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/4.2
CIS AWS Foundations 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 1.11 PCI DSS 8.1.4, 8.2.3, 8.2.4, 8.2.5	[CIS.1.5] Ensure IAM password policy requires at least one uppercase letter [CIS.1.6] Ensure IAM password policy requires at least one lowercase letter [CIS.1.7] Ensure IAM password policy requires at least one symbol [CIS.1.8] Ensure IAM password policy requires at least one number [CIS.1.9] Ensure IAM password policy requires minimum password length of 14 or greater [CIS.1.10] Ensure IAM password policy prevents password reuse [CIS.1.11] Ensure IAM password policy expires passwords within 90 days or less	- arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.5 - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.6 - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.7 - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.8 - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.9 - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.10 - arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.11
PCI DSS 3.4 CIS AWS Foundations 2.1.1	[PCI.S3.4] S3 buckets should have server-side encryption enabled	- pci-dss/v/3.2.1/PCI.S3.4
PCI DSS 7.2.1 CIS AWS Foundations 1.22	[PCI.IAM.3] IAM policies should not allow full "*" administrative privileges	- pci-dss/v/3.2.1/PCI.IAM.3

The second remediation implemented is enforcing password policy according to the CIS recommendations and PCI DSS requirements. This can also be done with code by simply updating the password policy if Security Hub notices change or too weak password policies. This Config rule is not triggered by config change and therefore only runs twice a day. If the password policy is not enforced properly, the remediation Lambda *Compliance_CIS_1-5_1-11_Remediation* updates the policy with compliant values, as seen in Figure 48.

```
response = iam.update_account_password_policy(
    MinimumPasswordLength=16,
    RequireSymbols=True,
    RequireNumbers=True,
    RequireUppercaseCharacters=True,
    RequireLowercaseCharacters=True,
    AllowUsersToChangePassword=True,
    MaxPasswordAge=90,
    PasswordReusePrevention=24,
    HardExpiry=False
    )
if response ['ResponseMetadata']['HTTPStatusCode'] == 200:
  result_msg = 'IAM Password Policy Updated :'
  result = ComplianceResult.CORRECTED
```

Figure 48 Compliance_CIS_1-5_1-11_Remediation: updating the password policy

The third remediation created is to enforce server-side encryption on S3 buckets. As mentioned in chapter 5.5 and Table 2. there is a requirement that data at rest needs to be encrypted to ensure that information even if stolen is safe. This requirement is a bit trickier to fix with code, but AWS has a service called AWS Systems Manager for automation of management tasks that can be called from the Boto3 SDK. There is a Systems Manager task that enables encryption on S3 buckets that can be used. This is used in the *Compliance_PCI_S3-4_Remediation* Lambda by calling the *start_automation_excexution()* function with the Document Name "AWS-EnableS3BucketEncryption" and the non-compliant s3 bucket as a parameter. We can assume that the execution was successful or, as the code does, wait for the response, this is not the best solution but a straightforward way to confirm success. The code can be seen in Figure 49

```
enableS3BucketEncryption = ssm.start_automation_execution(
    DocumentName='AWS-EnableS3BucketEncryption',
    DocumentVersion='1', # default
    Parameters={
        'BucketName': [ noncompliantS3Bucket ]
    }
)
count = 5 # wait for reslult max 5 seconds
while count > 0:
    --count
    automationExecutionResponse = ssm.get_automation_execution(
        AutomationExecutionId= enableS3BucketEncryption['AutomationExecutionId']
    )
    time.sleep(1) # wait on ssm result
    status = automationExecutionResponse['AutomationExecution']['AutomationExecutionStatus']
    if status != 'Pending' and status != 'InProgress' and status != 'Cancelling' and status != 'Waiting':
        break

if status == 'Success' :
    result = ComplianceResult.CORRECTED
    result_msg = 'Compliance Corrected for ' + noncompliantS3Bucket
```

Figure 49 Compliance_PCI_S3-4_Remediation: Using AWS Systems Manager for enforcing S3 Encryption

The last remediation created is an AWS policy enforcer that tries to remove liberal access policies with full administrative privilege; this is done in the *Compliance_PCI_IAM_3_Remediation* Lambda by using the Boto3 SDK to remove the policy as seen in Figure 50. To be able to remove a policy, all the previous policies need to be deleted first. Also, we should detach different resources and users as such. However, to complete the objective of the proof of concept, this was not necessary and therefore this is a future improvement.

```
iam = boto3.client('iam')
policy = iam.list_policy_versions(PolicyArn=iamPolicy)
# DeletePolicyVersion
for pol in policy['Versions']:
    if pol['IsDefaultVersion'] == False:
        iam.delete_policy_version(PolicyArn=iamPolicy, VersionId=pol['VersionId'])

response = iam.delete_policy(PolicyArn=iamPolicy)
```

Figure 50 Compliance_PCI_IAM_3_Remediation: Deleting too liberal IAM policy

6.3.3. Unsupported events

To further improve the proof of concept and be aware of unsupported compliance events, a mechanism for notifying about these events was also added. The CloudWatch event rule shown earlier in Figure 26 only filters out already implemented remediations and unsupported events are not ingested in the continuous compliance automation flow. Therefore, a manual way of handling unsupported events from the Security Hub

findings was created. This by adding a custom action target to the Security Hub findings console that triggers a CloudWatch event rule. The custom action target *ComplianceAutomation* can be selected in Security Hub when a finding is selected, as shown in Figure 51.

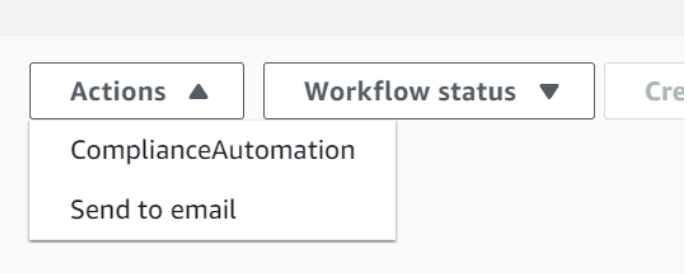


Figure 51 Custom Action target in Security Hub

The CloudWatch event rule catches the event and forwards the compliance event to the Step function *ContinuousComplianceAutomationStateMachine*. The compliance event is not present in the *ComplianceChoiceState* and therefore, the event is sent to a Task called *DeafultState*. The *DeafultState* adds information about the unsupported compliance event and forwards the information to the *FinalAutomationState,* as seen in Figure 52.

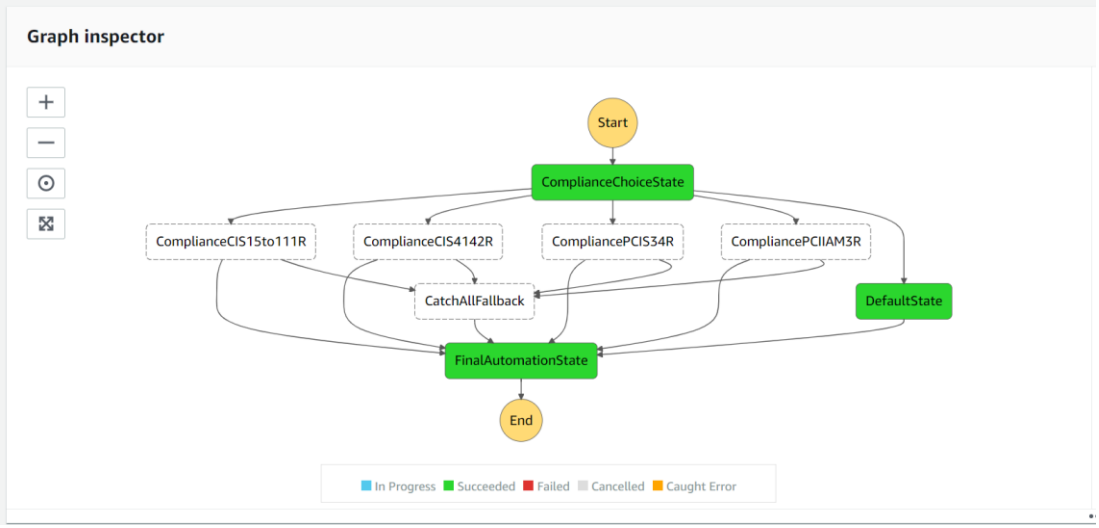


Figure 52 Step Function State machine flow of unsupported compliance event

The *FinalAutomationState* forwards the information to the *ComplianceAutomationFinalState* Lambda that extracts the information and formats the information as discussed previously. The message is then published on to the SNS queue and the *ComplianceTeamsWebHook* Lambda creates a message that is sent to

the Teams channel also as discussed previously. An example of the notification message for an unsupported event can be seen in Figure 53.

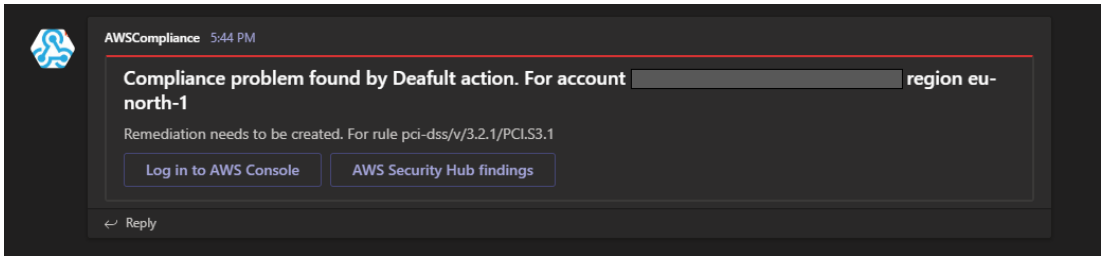


Figure 53 Teams message about unsupported compliance event

6.4. Design alternatives

As with many projects, there are many design alternatives and ways of solving issues. The proof of concept is only an example of how a continuous compliance automation architecture can be created. There are many ways this can be achieved by using the different services provided by AWS in different combinations. However, the proof of concept tries to provide an easy and flexible way of solving compliance problems and automate the remediations.

As discussed previously, the service Security Hub can be skipped and the AWS Config rules can be directly forwarded to the CloudWatch event rule for further handling, as shown in Figure 54.

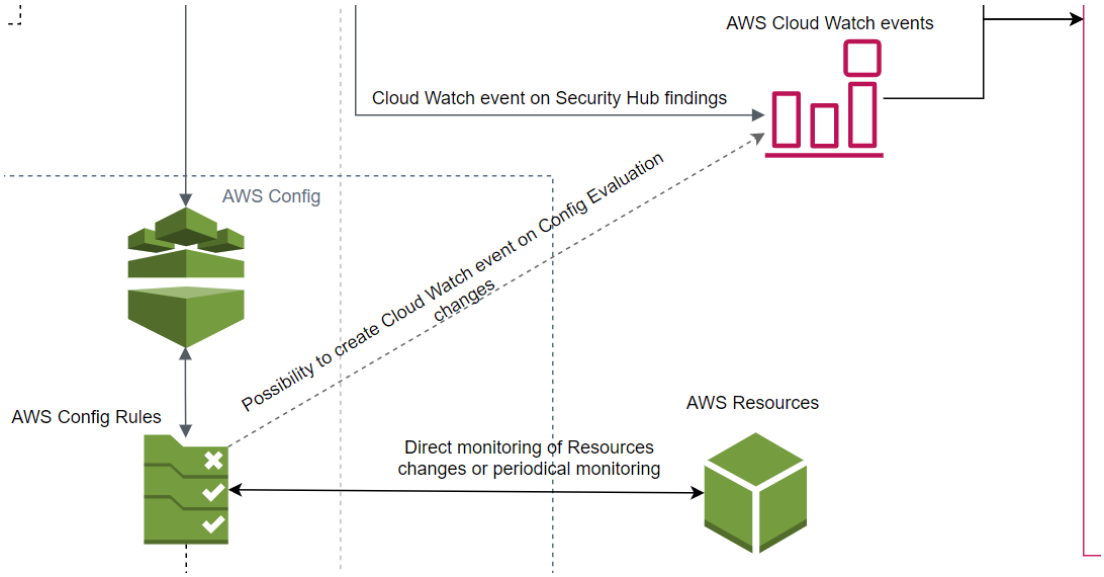


Figure 54 AWS Config directly forwarding the event to CloudWatch

The problem with this solution is that the AWS Config Evaluation does not directly contain information about compliance requirements and there is no easy way of tracking the different findings of Config rule evaluations. Security Hub provides more information on the non-compliant resource, what compliance rule was breached and severity level. Security Hub also provides a user-friendly interface for exploring compliance findings.

Another design alternative could be to skip the Step Function State machine used. Many CloudWatch event rules could instead be used for separating the differently implemented compliance remediations. This means creating one CloudWatch event rule for each compliance rule monitored. The alternative reduces the complexity and the tools used. However, this will decrease the visibility of the flow and the Lambda that handles the result needs to handle both errors and unsupported events in an efficient way. Also, failed Lambda executions could potentially cause problems if not handled correctly. This design alternative is illustrated in Figure 55.

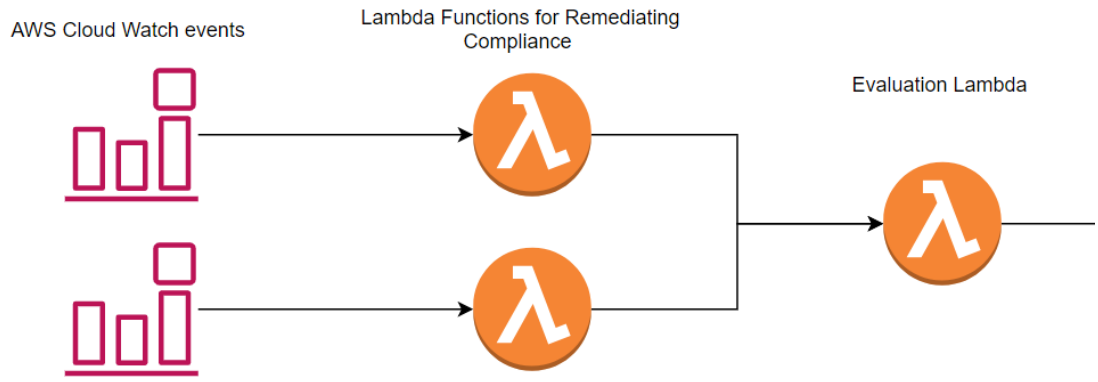


Figure 55 CloudWatch event rules forwarding events directly to Lambda functions

There are also many ways of handling the result of the continuous compliance automation flow. The possibility to send email instead of Teams messages is easily interchangeable by simply reconfigure the SNS queue. And unsupported/non-correctable events could be sent to a ticketing system like Jira for further investigation or the creation of new remediations. These alternatives can be seen in Figure 56.

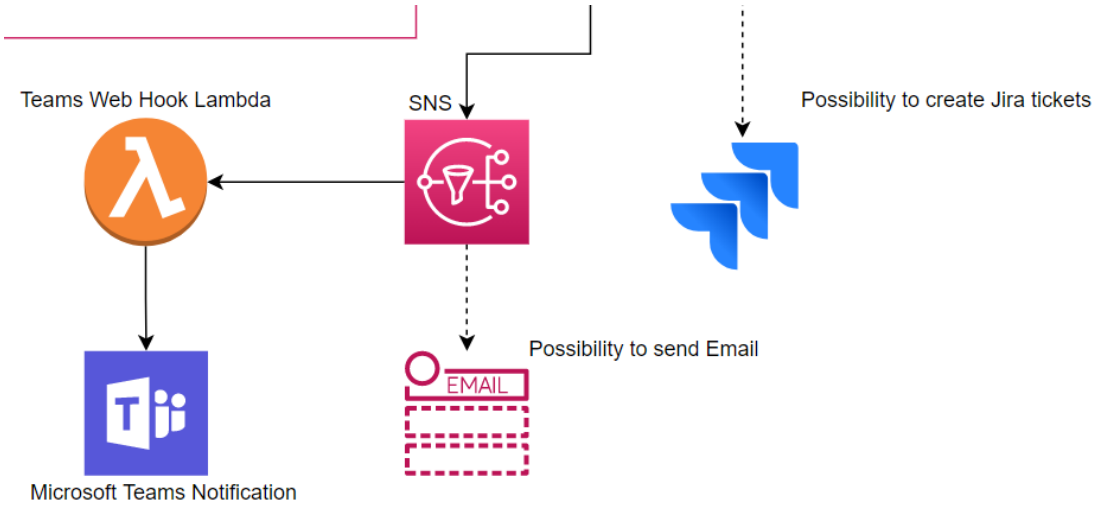


Figure 56 Different ways of notifying about result of the compliance remediations

7. Evaluation and future work

7.1. Improvements and future work

As the name says, the proof of concept is a solution that only shows that the theory works. Therefore, there are many possibilities to improve the solution and make it more efficient or manageable. Some of the possible improvements are technical and some are purely preferences as discussed in 6.4 Design alternatives.

The immediate technical improvements that should be made are to the code structure and the Ansible file structure. All the functions in the CloudFormation template are inline code and every resource included in the architecture is in the same file due to references between resources. CloudFormation has a limit of 51,200 bytes on the file size of a stack uploaded to AWS and by expanding the solution, the limit will be reached quite fast. The file size could easily be fixed by separating the Lambda code and creating an Ansible task that uploads the Lambdas to an S3 bucket in AWS. Also, separating code into different Ansible tasks where possible and using a parameter to pass references to the next task decreases the file size and creates better code separation.

When looking at code improvements, the most critical thing for improving the solution is compliance. Due to only focusing on proving that the proof of concept works, the issue of creating non-compliant resources was not checked; this means the proof of concept creates non-compliant resources itself. In some cases, this cannot be correct due to technical limitations, for example when a resource needs to be accessible from any IP address. However, tagging or marking the resources for exclusion in the remediation flow needs to be implemented. Also, verifying that the solution itself is compliant with the imposed compliance frameworks is a task that needs to be addressed when implementing the continuous compliance automation architecture. Depending on what compliance frameworks are needed, these improvements can vary.

When looking at implementing the continuous compliance automation architecture, there are also some considerations if this is done in a multi-AWS account setup for an organization. The proof of concept focuses on enabling all the services on one AWS account. However, some parts of the solution are ready for multi-account support, as

the enabling of AWS Config and Security Hub. To get all findings from other accounts to the Security Hub master account, the accounts must either belong to an AWS organization or accept invitations sent by the master account. Code changes are needed for this to work and a clear plan for handling everything from permissions, executions of remediations and logging is needed. This introduces complexity to the whole continuous compliance automation architecture but can be done by carefully planning and using the AWS services.

Also, for further work, the supported compliance frameworks and implemented compliance rules can be expanded and improved upon. This depends on the needs of the organization and what compliance frameworks they need to comply with. As discussed, the compliance process in the organization is highly dependent on what industry they operate in and what kind of infrastructure they have. Here, introducing third-party integrations into Security Hub can be an alternative if the organization already has any supported CCM tools or processes that can produce compliance events.

The possibilities for improving the proof of concept are many. But the idea was to create a solution that can easily be modified for different needs and show the strengths of implementing a continuous compliance automation architecture in a public cloud environment.

7.2. Evaluation

The evaluation of the proof of concept was done by implementing the continuous compliance automation architecture on an AWS account in the organizational infrastructure that this thesis was done for. The continuous compliance automation solution then automatically found and remediated any compliance problems in that AWS account within the implemented requirements. The solution was monitored and tested for a while and evaluated by the supervisor as well as other concerned parties in the company. The implemented solution worked as intended and remediated the found compliance problems that were supported. During the testing period, a couple of drawbacks were found that may need to be resolved if the solution is to be used in a larger scale organization.

The most relevant problem of the solution is that remediations take place automatically and, without exceptions, remove all non-compliant resources. The removal of non-compliant resources may cause problems for users who are not aware of what they did wrong. The solution does not currently support notifying individual users of their misconfigurations and the only way for a user to be notified if the resource was non-compliant is to look at the Security Hub findings or in the Teams channel. This does not stop the solution from working but can cause problems for unaware users. An immediate solution for this was not found, but AWS support provided information about possibilities to parse logs for identifying which user created the non-compliant resource, which would be a possible way of notifying specific users. However, this was not explored further due to the organizational setup with centralized logging and permission restrictions that are needed.

In some cases, the strict removal of non-compliant resources in combination with manual modifications can cause CloudFormation drift and mismatches during CloudFormation stack deployments. This problem is solved by being informed of the compliance remediations active and analyzing as well as resolving the drift when deployments fail. These kinds of problems are manageable and can be resolved by manual processes and by the developers being informed about how things work.

The continuous compliance automation architecture also does not remove all manual processes from the compliance work. There still need to be manual processes for resolving unhandled exceptions and fixing non-supported compliance events. Also, mappings of different compliance frameworks as well as defining what rules that need to be followed are still a requirement for the overall compliance process. However, repetitive tasks of proving compliance can be automated and remediated by the continuous compliance automation solution. The solution also supports continuous compliance with event-driven checks done in real-time instead of only snapshots of the compliance state at specific times when auditing occurs. The continuous compliance automation process ensures that the compliance posture is better maintained between audits and which is a welcomed feature to the compliance and security department. Current compliance processes are mainly focused on proving compliance at a specific point in time, but by using continuous compliance automation, the compliance state of the system can be more closely followed.

As there was already a plan for moving to public a cloud provider, this solution's increased work and cost are marginal. During the testing period, the cost of the continuous compliance automation solution was below $5, with the highest cost being for AWS Config rules and Security Hub. The Step Function and Lambda executions were minimal and ended up being under the Free Tier included in the AWS accounts used. However, an accurate estimate for a large solution with many accounts is hard to give. The highest cost will be from AWS Config rule triggers on resources and the more resources there are, the more cost it will accumulate.

In addition, the move to public cloud providers is often regarded as a risk for losing control of the hardware and the data processing control. However, the move enables exclusive services like AWS Config and Security Hub for improving safety and compliance. The risk can often be rectified by the possibility of improving the compliance posture and reducing many other security risks. Also, the possibility to implement a solution like continuous compliance automation is not possible in a private cloud solution to the same extent.

As discussed, the continuous compliance automation architecture does not solve the whole compliance process but can be used as an additional tool for solving compliance problems and improving the compliance posture. This especially when implementing new solutions built in AWS environments.

8. Conclusion

The purpose of the thesis was to reduce the compliance work needed when moving applications to an AWS environment by introducing automated continuous compliance tools and processes for remediations of compliance problems. When introducing new systems to an already existing compliance process, the work needed can be complex and challenging. The thesis highlight the most essential processes and tools available in AWS to reduce the overall compliance work. Most importantly, this thesis creates a proof of concept for a continuous compliance automation solution. The continuous compliance automation is constantly monitoring for non-compliant resources in the AWS environment as well as tries to remediate the found compliance breaching resources. By combining different tools in AWS and creating a flexible architecture, resources and configurations can be monitored for compliance breaching configurations almost in real-time. The event-driven compliance checks combined with automated remediations for non-compliant resources try to minimize human involvement in the compliance process. This goal is reached by fixing reoccurring compliance breaching resources with code. The benefits of using a public cloud provider are often overlooked and can be beneficial for the financial sector to improve both security and compliance problems.

The continuous compliance automation solution is not perfect but serves as a satisfactory proof of concept for showing that automatic remediation of compliance problems in a public cloud environment is possible. The possibilities and flexibility for implementation are also shown and the solution is a good start on architecture for supporting an automated continuous compliance flow. It does not fix all compliance problems itself but is a helpful tool to improve the compliance postures of the system and minimize repetitive tasks.

9. Swedish summary – Svensk sammanfattning

Introduktion

Behovet av att följa olika regelverk och bestämmelser har ökat inom Informationsteknologisektorn (IT) under de senaste åren. Antalet regelverk som skall följas och deras rekommendationer ökar också i komplexitet hela tiden, detta leder till ett ökat arbete inom IT-branschen för att tillfredsställa både kundernas och myndigheternas krav. Inom vissa branscher kan man inte ens utöva verksamhet om man inte följer dessa krav. En sådan bransch är finanssektorn där mycket stränga krav finns från flera aktörer.

I detta arbete studeras krav från några förordningar och deras inverkan på IT-system inom finanssektorn, samt vilken anpassning som måste utföras i processen för att kunna följa krav från förordningar när man flyttar IT-system från privata serverhallar till en publik molntjänst. I en publik molntjänst kan det ske stora förändringar i infrastruktur och konfigurering fort, vilket skapar problem om en människa skall följa dessa förändringar samt se till att förändringarna inte bryter mot någon förordning. Därför måste automation införas i förordningsprocessen för att klara av hanteringen av de många kraven. Diplomarbetet introducerar automation både när det gäller förordningsprocessen men också när det gäller återkommande åtgärder av resurser som inte följer kraven, detta med hjälp av kod. För att kunna skapa denna automation används Amazon Web Services (AWS) olika verktyg och mjukvarubibliotek. Förordningarna och kraven som detta arbete tar upp är ”Center for Internet Security (CIS) benchmark” och ”Payment Card Industry Data Security Standard (PCI-DSS)”.

Bakgrund

Bakgrunden till att det finns förordningar och krav som skall följas är att säkerställa att man har ett tillräckligt säkert IT-system. När man pratar om säkerhet är det ofta kopplat till att kunna känna sig trygg i sin verksamhet med de hot som finns, medan säkerhetsmålen med en förordning ofta försöker se till att alla skall följa vissa krav på ett visst sätt för att säkerställa en viss nivå på säkerhet. [1] För att kunna följa dessa förordningar behövs en process som går igenom kraven och säkerställer att alla delar av ett IT-system följer dessa krav. För de flesta företag finns det någon sådan process

men den är till stor del manuell. Det vill säga man försöker verifiera och evaluera ett system manuellt vid en viss tidpunkt för att få en överblick över hur bra man följer kraven. Detta ofta i samband med en auditering. Detta illustreras i Figure 1. [2]

För att förbättra denna process kan man introducera automation som går igenom kraven och jämför dessa mot existerade resurser, för att få en överblick av vilka resurser som bryter mot kraven. Om man också gör detta på en periodisk basis kan man få en bättre bild av systemet under alla dessa tidpunkter och då bättre förhålla sig till förändringar i resurserna samt till kraven. Detta illustreras i Figure 2. För att ytterligare förbättra denna process kan man om möjligt introducera händelsebaserade jämförelser som kan utvärdera en resurskonfigurering genast efter att den har ändrats och på detta vis snabbt fånga upp resurser som bryter mot kraven som ställs. En sådan process illustreras i Figure 3. [2] [3] [4]

För att underlätta arbetet med att bevisa för en auditör att ett system följer de krav som förordningarna ställer så skulle man föredra att alla resurser och konfigureringar verifieras så fort de ändras. Dock är detta inte alltid möjligt, men genom att försöka automatisera så mycket som möjligt kan arbetsbördan minskas något. Genom att fortsätta automatiseringen efter att man jämfört kraven mot en resurs kan man minska arbetet ytterligare, detta genom att automatiskt försöka åtgärda resursen som inte följer kraven, speciellt återkommande problem. Ett sådant flöde illustrerats i Figure 4. [7]

Om man automatiserar flera steg i förordningsprocessen behöver man också automatisera kodleveranserna. Detta för både infrastrukturkod och kod som åtgärdar infrastruktur samt konfiguration som bryter mot kraven. Detta kan göras med olika verktyg för kontinuerlig integration och leverans av kod, samt kodversionshanteringssystem.

Utförande

För att kunna bevisa att man kan automatisera förordningsprocessen och även automatisera åtgärdande av krav, utfördes en konceptvalidering. Konceptvalideringen gick ut på att använda olika verktyg för kodautomation samt olika verktyg i AWS för att automatisera förordningsprocessen. Infrastruktur som kod användes och därmed kan alla delar av konceptet skapas med versionshanterad kod.

För att kunna skapa infrastruktur som kod användes verktyget Ansible, som är ett automationsverktyg för konfigurering och hantering av IT-infrastruktur. För att kunna skapa resurser och infrastruktur i AWS används Ansible-modulen ”cloudformation” som möjliggör hantering av AWS CloudFormation kod som sedan kan laddas upp i AWS för att skapa AWS specifika resurser med dess konfigureringar. CloudFormation är ett mallverktyg som fungerar som resurshanterare i AWS och med en mall kan man skapa samt konfigurera resurser i sin AWS-infrastruktur. I detta fall används Ansible enbart som ett konfigureringsverktyg medan CloudFormation mallen används för att hantera infrastruktur samt dess specifika konfigurering. Genom att kombinera dessa kan koden hantera mer komplex infrastruktur samt hantera flera ”CloudFormation templates” och där med mer AWS-infrastruktur med samma automationslösning. All kod är sedan versionshanterad med Bitbucket och exekveras av Ansible Tower som är ett kontinuerligt integrationsverktyg.

För att skapa en hållbar lösning för att automatisera förordningsprocessen användes olika verktyg som AWS tillhandahåller och så små förändringar i dessa verktyg som möjligt. Detta för att minska på utveckling samt hålla lösningen så flexibel som möjligt. Det finns många designalternativ men en lösning som är flexibel och har lätt översikt valdes. En överblick av lösningens alla delar och arkitektur är illustrerad i Figure 19.

Lösningens första verktyg är AWS Security Hub, som är en samlingshub för krav ur olika förordningsramverk. Security Hub sammanfogar och samlar andra verktygs fynd och information om resurser som inte följer krav från olika förordnings ramverk. Ett av de verktyg som Security Hub använder är AWS Config som använder sig av Config Rules (konfigureringsregler) som är funktioner som övervakar förändringar av konfigureringar i resurser. AWS Security Hub använder dessa regler för att jämföra resurserna mot krav som ställs av de förordningar man har valt att använda. Om en resurs inte följer ett krav, skapas ett fynd som visas till användaren med indikationer på allvarlighet och vilket krav resursen bryter emot, denna information sparas också i ett JSON-objekt.

För att kunna använda sig av dessa fynd använder sig lösningen av AWS CloudWatch, som kan monitorera och reagera på fynd och information från flera verktyg i AWS. Ett av dessa verktyg är Security Hub och dess fynd. Lösningen använder sig av en tjänst i

AWS CloudWatch som heter CloudWatch Events för att fånga upp dessa fynd och kan filtrera ut fynd till enbart de implementerade kraven i lösningen. Dessa fynd skickas sedan vidare som information i ett JSON-objekt till nästa verktyg som är AWS StepFunction. AWS StepFunction är ett verktyg som hanterar arrangeringen av serverlösa funktioner, men hjälp av verktyget kan man skapa kedjor med funktioner och hantera olika händelsebaserade exekveringar. Med hjälp av detta verktyg kan lösningen reagera på olika krav, filtrera ut dem, samt starta en AWS Lambda-funktion som försöker åtgärda resursen som inte följer kraven.

En AWS Lambda är en funktion som serverlöst kan exekvera kod baserat på en händelse, vilket i detta fall är ett fynd som skickats från Secuirty Hub via AWS CloudWatch till AWS StepFunction som startar funktionen. Koden i Lambda-funktionen försöker sedan automatiskt åtgärda resursens konfigurering så att resursen inte bryter mot ett förordningskrav. Resultatet skickas sedan vidare till nästa Lambda-funktion i AWS Stepfunktion och koden evaluerar om resursen gick att åtgärda eller inte. All information samlas ihop och beroende på resultatet av åtgärden formateras ett meddelande på olika sätt i JSON-format som skickas vidare till en ”Amazon Simple Notification Service” (SNS) kö. SNS är en meddelandeservice som kan hantera olika meddelandetyper och olika kösystem för dessa meddelanden. En Lambda-funktion lyssnar på denna kö och startar en exekvering av kod så fort ett meddelande skrivs till kön. Lambda-funktionen formaterar om informationen till ett format som kan skickas till en så kallad Webhook, i detta fall en webbadress som kan hantera inkommande webbegäran i Microsoft Teams. Detta resulterar i en notifiering och ett meddelande i en chattkanal i applikationen Microsoft Teams som en användare sedan kan ta del av.

Hela denna process är automatisk och användaren behöver inte göra något förutom att reagera på icke-åtgärdade resurser, ifall lösningen inte klarade av att hantera dessa. Detta möjliggör att man kan skapa kod för att åtgärda återkommande problem med resurser som inte följer förordningskraven och på detta sätt minska återkommande arbete.

Resultat

Konceptvalideringen och lösningen implementerades i en aktiv AWS-miljö och konto, samt testades en tid med de implementerade förordningskrav som skapades. Lösningen

fungerade som den skulle och åtgärdade resurser som inte följde kraven. Dock stötte man på problem med användare som inte förstod varför vissa resurser slutade fungera som tänkt, samt att lösningen inte tog hänsyn till resurser som behöver bryta mot krav på grund av olika orsaker. En nyttig förbättring skulle vara att meddela resursskaparen om omkonfigurering av resursen, så att användaren kunde vara medveten om förändringen. Dessa problem kunde inte lösas under konceptvalideringen men det hindrar inte heller lösningen från att fungera. Ett fullskaligt test på flera AWS-konton gjordes inte heller, vilket ger en ofullständig bild av hur lösningen skulle fungera i en stor miljö. Dess ackumulerade kostnad kan inte heller förutspås. Kostnaden av konceptvalideringen var minimal med de tester som gjordes.

Lösningen togs emot som en positiv förbättring av arbetet som utförs med förordningar och krav när det gäller flytt till en publik molntjänst. Lösningen visar också möjligheter och flexibiliteten av att utnyttja verktyg i en publik molntjänst för att underlätta och automatisera arbetet med förordningsramverk och dess krav.

10. References

[1] K. Julisch, “Security compliance: The next frontier in security research,” *Proc. New Secur. Paradig. Work.*, pp. 71–74, 2009, doi: 10.1145/1595676.1595687.

[2] P. Kirby, “Managing Inventory, Change, and Compliance with AWS Config | Pluralsight.” [Online]. Available: https://www.pluralsight.com/courses/aws-config-inventory-change-compliance. [Accessed: 31-Mar-2020].

[3] R. Filepp, C. Adam, M. Hernandez, M. Vukovic, N. Anerousis, and G. Q. Zhang, “Continuous compliance: Experiences, challenges, and opportunities,” *Proc. - 2018 IEEE World Congr. Serv. Serv. 2018*, no. Cc, pp. 21–22, 2018, doi: 10.1109/SERVICES.2018.00029.

[4] J. Jarvinen, D. Hamann, and R. van Solingen, “On integrating assessment and measurement: Towards continuous assessment of software engineering processes,” in *International Software Metrics Symposium, Proceedings*, 1999, pp. 22–30, doi: 10.1109/metric.1999.809722.

[5] R. V O’connor, P. Elger, and P. M. Clarke, “Continuous Software Engineering-A Microservices Architecture Perspective,” 2017.

[6] T. Karvonen, “Continuous software engineering in the development of software-intensive products towards a reference model for continuous software engineering Teemu Karvonen,” 2017.

[7] B. Fitzgerald and K. J. Stol, “Continuous software engineering: A roadmap and agenda,” *J. Syst. Softw.*, vol. 123, pp. 176–189, Jan. 2017, doi: 10.1016/j.jss.2015.06.063.

[8] Ansible, “WHITEPAPER CI/CD with Ansible.” [Online]. Available: https://www.ansible.com/hubfs/pdfs/ContinuousDelivery-with-Ansible-WhitePaper.pdf. [Accessed: 31-Mar-2020].

[9] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, “DevOps: Introducing infrastructure-as-code,” in *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, 2017, pp. 497–498, doi: 10.1109/ICSE-C.2017.162.

[10] D. C. Cheng, J. B. Villamarin, G. Cu, and N. R. Lim-cheng, “Towards end-to-end Continuous Monitoring of Compliance Status Across Multiple Requirements,” vol. 9, no. 12, pp. 456–466, 2018.

[11] I. Amazon Web Services, “What is Cloud Computing.” [Online]. Available: https://aws.amazon.com/what-is-cloud-computing/?nc1=f_cc. [Accessed: 30-Mar-2020].

[12] S. Narula, A. Jain, and Prachi, “Cloud computing security: Amazon web service,” in *International Conference on Advanced Computing and Communication Technologies, ACCT*, 2015, vol. 2015-April, pp. 501–505, doi: 10.1109/ACCT.2015.20.

[13] IBM, “What is Hybrid Cloud? | IBM,” 2020. [Online]. Available: https://www.ibm.com/cloud/learn/hybrid-cloud. [Accessed: 30-Mar-2020].

[14] ICF and Centre for European Policy Studies(CEPS), *Study on the costs of compliance for the financial sector*, no. July. 2019.

[15] Center for Internet Security, “CIS Benchmarks™ FAQ.” [Online]. Available: https://www.cisecurity.org/cis-benchmarks/cis-benchmarks-faq/. [Accessed: 08-Mar-2020].

[16] CIS, “CIS Controls,” *Cent. Internet Secur.*, no. 7.1, 2019.

[17] PCI Security Standards Council, “Payment Card Industry (PCI) Data Security Standard Requirements and Security Assessment Procedures Version 3.2.1 Document Changes Date Version Description Pages,” 2018.

[18] PCI Security Standards Council, “Official PCI Security Standards Council Site - Verify PCI Compliance, Download Data Security and Credit Card Security Standards.” [Online]. Available: https://www.pcisecuritystandards.org/pci_security/maintaining_payment_security. [Accessed: 08-Mar-2020].

[19] I. Amazon Web Services, “About AWS.” [Online]. Available: https://aws.amazon.com/about-aws/. [Accessed: 03-Apr-2020].

[20] I. Amazon Web Services, “Regions, Availability Zones, and Local Zones -

Amazon Elastic Compute Cloud.” [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html. [Accessed: 07-Mar-2021].

[21] I. Amazon Web Services, “Shared Responsibility Model - Amazon Web Services (AWS).” [Online]. Available: https://aws.amazon.com/compliance/shared-responsibility-model/. [Accessed: 03-Apr-2020].

[22] I. Amazon Web Services, “What is AWS CloudFormation? - AWS CloudFormation.” [Online]. Available: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html. [Accessed: 03-Apr-2020].

[23] “What is IAM? - AWS Identity and Access Management.” [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html. [Accessed: 14-Feb-2021].

[24] I. Amazon Web Services, “AWS Config Developer Guide,” 2021. [Online]. Available: https://docs.aws.amazon.com/config/latest/developerguide/config-dg.pdf. [Accessed: 21-Feb-2021].

[25] I. Amazon Web Services, “AWS Security Hub User Guide,” 2021. [Online]. Available: https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub.pdf. [Accessed: 21-Feb-2021].

[26] I. Amazon Web Services, “What Is Amazon CloudWatch? - Amazon CloudWatch.” [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html. [Accessed: 03-Apr-2020].

[27] I. Amazon Web Services, “What Is AWS Lambda? - AWS Lambda.” [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. [Accessed: 03-Apr-2020].

[28] I. Amazon Web Services, “What is Amazon SNS? - Amazon Simple Notification Service.” [Online]. Available:

https://docs.aws.amazon.com/sns/latest/dg/welcome.html. [Accessed: 03-Apr-2020].

[29] I. Amazon Web Services, “What Is AWS Step Functions? - AWS Step Functions.” [Online]. Available: https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html. [Accessed: 03-Apr-2020].

[30] I. Amazon Web Services, “Boto3 documentation — Boto3 Docs 1.17.22 documentation.” [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html#. [Accessed: 07-Mar-2021].

[31] “amazon.aws.cloudformation – Create or delete an AWS CloudFormation stack — Ansible Documentation.” [Online]. Available: https://docs.ansible.com/ansible/latest/collections/amazon/aws/cloudformation_module.html. [Accessed: 14-Feb-2021].

[32] “How Ansible Works | Ansible.com.” [Online]. Available: https://www.ansible.com/overview/how-ansible-works. [Accessed: 14-Feb-2021].

[33] “Best Practices — Ansible Documentation.” [Online]. Available: https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html#directory-layout. [Accessed: 14-Feb-2021].

[34] “Managing AWS Infrastructure as Code using Ansible, CloudFormation, and CodeBuild | Programmatic Ponderings.” [Online]. Available: https://programmaticponderings.com/2019/07/30/managing-aws-infrastructure-as-code-using-ansible-cloudformation-and-codebuild/. [Accessed: 14-Feb-2021].

[35] Ansible, “Ansible Tower | Ansible.com.” [Online]. Available: https://www.ansible.com/products/tower. [Accessed: 14-Feb-2021].

[36] “About Python · HonKit.” [Online]. Available: https://python.swaroopch.com/about_python.html. [Accessed: 14-Feb-2021].

[37] “What is Python? Executive Summary | Python.org.” [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed: 14-Feb-2021].

[38] “YAML Ain’t Markup Language (YAML™) Version 1.2.” [Online]. Available: https://yaml.org/spec/1.2/spec.html#id2759572. [Accessed: 14-Feb-2021].

[39] “JavaScript Object Notation (JSON).” [Online]. Available: https://www.ietf.org/rfc/rfc4627.txt. [Accessed: 14-Feb-2021].

[40] P. Durkin and P. Durkin, “Project final version Can PCI DSS compliance be achieved in a cloud environment?”

[41] A. Jonathan Rau, “Continuous compliance monitoring with Chef InSpec and AWS Security Hub | AWS Security Blog.” [Online]. Available: https://aws.amazon.com/blogs/security/continuous-compliance-monitoring-with-chef-inspec-and-aws-security-hub/. [Accessed: 06-Apr-2020].

[42] AWS, “Cloud Security – Amazon Web Services (AWS).” [Online]. Available: https://aws.amazon.com/security/. [Accessed: 14-Apr-2020].

[43] AWS, “Continuous Compliance with AWS Security Hub | AWS Online Tech Talks.” [Online]. Available: https://pages.awscloud.com/Continuous-Compliance-with-AWS-Security-Hub_2019_0601-SID_OD.html?&trk=ep_card-el_a131L000005v8UeQAI&trkCampaign=NA-FY19-AWS-DIGMKT-WEBINAR-SERIES-June_2019_0601-SID&sc_channel=el&sc_campaign=pac_2018-2019_exlinks_ondemand_OTT_e. [Accessed: 14-Apr-2020].

[44] J. Rau, “jonrau1/ElectricEye: Continuously monitor your AWS services for configurations that can lead to degradation of confidentiality, integrity or availability. All results will be sent to Security Hub for further aggregation and analysis.” [Online]. Available: https://github.com/jonrau1/ElectricEye. [Accessed: 14-Apr-2020].

[45] J. Backes *et al.*, “Semantic-based Automated Reasoning for AWS Access Policies using SMT,” *Proc. 18th Conf. Form. Methods Comput. Des. FMCAD 2018*, pp. 206–214, 2019, doi: 10.23919/FMCAD.2018.8602994.

[46] Andrew Gacek, “How AWS uses automated reasoning to help you achieve security at scale | AWS Security Blog,” 2018. [Online]. Available:

https://aws.amazon.com/blogs/security/protect-sensitive-data-in-the-cloud-with-automated-reasoning-zelkova/. [Accessed: 07-Feb-2021].

[47] A. Web Services, “Introduction to AWS Security,” 2020. [Online]. Available: https://d1.awsstatic.com/whitepapers/Security/Intro_to_AWS_Security.pdf?did=wp_card&trk=wp_card. [Accessed: 07-Feb-2021].

[48] Center for Internet Security, “CIS Amazon Web Services Foundations Benchmark 1.3.0,” pp. 0–157, 2020.

[49] I. Amazon Web Services, “Payment Card Industry Data Security Standard (PCI DSS) 3.2.1 on AWS Compliance Guide,” 2021.

[50] A. Web Services, “Standardized Architecture for PCI DSS on the AWS Cloud Quick Start Reference Deployment,” 2020.

[51] Center for Internet Security, “CIS Controls Navigator.” [Online]. Available: https://www.cisecurity.org/controls/cis-controls-implementation-groups/. [Accessed: 21-Feb-2021].