TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technology
Department of Software Science

ITC70LT
Martin Leppik 153618IVCM

IMPROVING AWS S3 SECURITY AT A MEDIUM-SIZED COMPANY: CHALLENGES AND SOLUTIONS

Master thesis

Supervisor: Hayretdin Bahsi, Ph.D

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Martin Leppik

[19.05.2020]

Annotatsioon

Paljud ettevõtted on hakanud tänapäeval eelistama enda loodud andmekeskuste asemel avalikke pilveteenuseid. Üks populaarsemaid teenusepakkujaid selles sektoris on Amazon Web Services (AWS), kus kasutatakse AWS S3 nimelist teenust andmete hoiustamiseks. Sinna luuakse üksteisest isoleeritud loogilisi üksusi, mida nimetatakse ämbriteks, milles sisalduvad faile kutsutakse objektideks. Üks levinumaid AWS S3 kasutusjuhte on sinna salvestada objekte, mida ettevõtte pakutava teenuse kasutajad on üles laadinud. AWS S3 teenust on lihtne üles seada kasutades vaikesätteid, aga seetõttu on samuti lihtne teha vigu pääsuõigustes, mille tagajärjel võivad kasutajate poolt salvestatud objektid muutuda avalikult Internetist kättesaadavaks.

Käesolev lõputöö uurib, kuidas tõsta AWS S3 teenuse turvalisust keskmise suurusega ettevõttes. Esmalt antakse ülevaade S3 ämbrite konfiguratsiooni valikutest, pääsumehhanismidest, AWS-i sisse ehitatud turvateenustest ning kolmanda osapoole S3 auditeerimise tööriistadest. Analüüsi osa on jaotatud neljaks erinevaks faasiks, mis puudutavad järgnevaid AWS S3 osasid: ämbrite konfiguratsioon, pääsumehhanismid, monitooring ja tarkvaraarendusega seotud probleemid. Igas faasis kasutatakse toiming-uurimus metodoloogiat, mille kaudu kirjeldatakse peatüki eesmärki, kasutatavaid meetodeid, meetrikat, probleemide tuvastamist ja vajalikke muudatusi ning hinnatakse faasi edukust vastavalt püstitatud eesmärkidele.

Selle lõputöö panuseks on laiaulatuslik AWS S3 teenuse turvalisuse audit kasutades AWS-i ja kolmanda osapoolete tööriistu, mis on eskaleeritav keskkonnas, kus on sadu ämbreid. Lisaks antakse soovitused, kuidas tõsta üldist S3 turvalisust vältides enamlevinuid vigu pääsumehhanismides. Rakendatakse S3 konfiguratsiooni standard ja juurutamise protsess ning lisatakse faililaiendi põhine valgefiltreerimine, mis takistab pahatahtlike skriptide käivitamist otse veebilehitsejas pärast faili avamist. Selle lõputöö tulemused põhinevad päris ettevõttel.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 69 leheküljel, 7 peatükki, 1 joonist ja 12 tabelit.

Abstract

Many companies have started to prefer public cloud services to private data centers. Amazon Web Services (AWS) with its S3 service is one of the more popular services for storing data. Logically isolated units created there are called buckets which in turn contain objects. One popular use case for S3 is to store objects uploaded by users via company-provided services so that these objects can later be accessed by the user online. It is easy to configure S3 but on the other hand it is also easy to make mistakes in access control mechanisms, rendering the user-uploaded objects publicly accessible from the Internet.

This thesis discusses the challenges of and solutions for improving the security of AWS S3 at a medium-sized company. Firstly, an overview of bucket configuration options, access control mechanisms, AWS native S3 security services, and 3rd party security tools is given. The analysis part is divided into four separate phases which touch the following AWS S3 parts: bucket configuration options, access control mechanisms, monitoring and development related issues. Every phase uses the action-research methodology describing the goal, methods, metrics, problem identification, action items and evaluating the results based on the success criteria.

The contribution of this thesis is a comprehensive security audit of the Simple Storage Service using AWS native and 3rd party tools which scale in an environment with hundreds of buckets. Additionally, recommendations are given for increasing S3 security posture by avoiding the most common mistakes in access control mechanisms, implementing the bucket configuration standard and deployment procedure, and applying MIME type-based whitelisting to prevent malicious scripts from executing after object retrieval. The results of the thesis are based on a real company.

This thesis is written in English and is 69 pages long, including 7 chapters, 1 figure and 12 tables.

Table of abbreviations and terms

ACL	Access Control List
ACP	Access Control Policy
Amazon S3	Amazon Simple Storage Service
AWS	Amazon Web Services
AWS IAM	AWS Identity and Access Management
CLI	Command Line Interface
CMS	Content Management System
CSB	Cloud Storage Broker
CSS	Cloud Storage Service
EC2	Elastic Compute Cloud
GDPR	General Data Protection Regulation
HTML	Hypertext Markup Language
KMS	Key Management System
MIME	Multipurpose Internet Mail Extensions
POC	Proof of Concept
RDS	Relational Database Service
SMT	Satisfiability Modulo Theories
URL	Uniform Resource Locator
XSS	Cross Site Scripting

Table of contents

List of figures

List of tables

1. Introduction

Amazon S3 is an AWS service that is used by many companies of all sizes for object storage. The logically isolated units created in S3 are called buckets which contain objects which are essentially files with metadata. While setting up object storage is relatively simple compared to building and maintaining your own data center, it is very easy to make configuration mistakes that lead to data leaks. There have been many cases where big corporations have suffered a data breach due to such mistakes. These breaches are likely to continue as long as it is possible to make S3 buckets public.

The Amazon S3 service launched in 2006 and from 2017 onward there have been many S3-related data breaches in public and private sectors [1]. For example, in 2017 UpGuard's Cyber Risk Team discovered one of the largest US voter data leaks which exposed 198 million records containing personal information and profiling data [2]. In 2016 Uber suffered a data leak and exposed 57 million records which included customer' personal information [3]. In 2019 UpGuard's data breach research team discovered another case where Attunity (Isreali IT firm) exposed its customer backups of Fortune 100 companies like Ford, Netflix and TD Bank via publicly exposed buckets [4].

AWS provides multiple mechanisms for controlling access to S3 resources: Identity and Access Management (IAM) policies, S3 bucket policies, and S3 Access Control Lists (ACLs). While IAM policies are attached to users, bucket policies and ACLs are configured at bucket level. It is not easy to decide which mechanism should be used because the appropriate mechanisms depend on the specific use case. There is even an authorization process where multiple control mechanisms are used together. If these aspects are not well understood, then mistakes are prone to happen.

Objects are most commonly made public with bucket or object ACLs. Amazon provides its tenants predefined ACL user groups that can be used for granting access. The authenticated user group is very often misunderstood. The wording leads AWS users to believe that they are granting permissions to their AWS IAM users within the same account but instead are allowing any authenticated AWS user in the world to access the S3 bucket.

Tenants commonly use multiple AWS accounts to separate different environments such as development, test, and production. These accounts may contain several hundred buckets. How to audit access controls and configuration options if there are so many buckets in separate accounts? AWS has tools that give valuable insights for the audit but their coverage is not comprehensive.

AWS S3 is also capable of bucket and object level logging with a service called AWS Cloudtrail. Not all companies have it configured. To take full advantage of Cloudtrail audit logs, log analysis needs to be set up. For example, this can be accomplished through open source tools like Elastic stack or 3rd party paid solutions. Unfortunately, many companies do not have the time or manpower to make sure that audit logs are stored and visualized.

1.1. Problem statement and author’s contribution

This thesis focuses on multiple aspects of improving the AWS S3 security based on the example of a medium-sized company. This study is important because the organization uses S3 extensively and has experienced security issues in relation to this in the past. The insights of this thesis can also be used by others to strengthen their S3 security posture.

The thesis relies on action-research methodology. This means that the author assumes the role of security consultant, finds issues related to the company’s use of S3, and collaborates with developers and infrastructure teams who apply fixes based on the author’s recommendations. The work is divided into multiple phases where action items are identified and executed and metrics about the improvements are collected. The first phase focuses on bucket configuration options. S3 access control mechanisms are audited in the second phase. The third phase discusses bucket log analysis and monitoring. The last phase inspects a sample of bug bounty program reports that are related to S3 development issues.

Auditing all bucket configuration options is not an easy task when there are hundreds of buckets in multiple AWS S3 accounts. AWS provides some native security services that are capable of checking only a few essential options. For example, AWS Config allows to check the status of server access logging, bucket replication, versioning, server-side encryption and also checks if there are policies that give public read or write access to the buckets. One of the goals is to find a quicker way to get an overview of all bucket configurations options in development, test and production AWS accounts. For example, if versioning, server access and object level logging, default encryption and lifecycle are configured. Areas for improvement are suggested after documenting the bucket configurations of three AWS accounts. One of the problems is related to finding out which buckets options should be configured. At first, the current bucket configuration and deployment procedure is reviewed. Then a new AWS S3 bucket configuration standard and deployment procedure is written and implemented. Later a video training about the aforementioned documents is created and launched. Lastly, the configuration options will be changed according to the set standard.

Incorrectly configured S3 access control mechanisms can lead to data breaches. Even the company that this thesis refers to has received multiple reports about misconfigured policies from its bug bounty program. This thesis focuses on auditing all of the S3 access control mechanisms in three different AWS accounts (development, test and production). Two tools are used for auditing: at first manual inspection of policies is done with AWS CLI and later Netflix SecurityMonkey is deployed to review the automated findings in relation to policy issues. Firstly, all access-related policies are documented using AWS CLI and manually reviewed. Secondly, Netflix SecurityMonkey is configured to automatically analyse access control mechanisms and produce findings on problematical policies. The challenge here is to eliminate false positive findings by determining which findings are actually problematic for the company. Finally, tickets are made for the infrastructure team to remedy the access control mechanisms issues based on the author's recommendations.

AWS Cloudtrail log analysis can be used for multiple purposes. For example, for checking compliance, conducting security analyses, troubleshooting, and getting more visibility into user and resource actions under the corresponding AWS account. Unfortunately, the analytics importance of these logs is often overlooked because it requires time and effort to set up the monitoring solution. Two different solutions are configured and tested in this thesis. Firstly, Rackspace Compass is tested and evaluated and secondly Wazuh is configured to start ingesting Cloudtrail logs. The goal is to be able to analyse the whole Cloudtrail log by capturing bucket and object level actions and then build dashboards upon them to get visibility into AWS S3. Log analysis findings can subsequently be used to adjust IAM service user policies that are too permissive.

Not all issues are related to bucket configuration or access control mechanisms. This thesis includes some reports from the company's bug bounty program which demonstrate the importance of correctly configuring object metadata. For example, if developers attach the wrong metadata to objects then script files that either execute XSS or allow malicious actors to steal other users' files can be run inline in the browser. The goal is to fix the issues by evaluating MIME type-based whitelisting and blacklisting together with attaching the correct metadata.

The constraint for tools and services used in this thesis is that they have to be either open source, AWS native services, or already implemented by the company. This is done to avoid onboarding new paid solutions as the company might already have something in place that can be used for this thesis.

The main contributions of the thesis are:

- Comprehensive AWS S3 bucket configuration and access control mechanism audit with multiple AWS, open source, and third-party solutions;
- Bucket configuration standard and deployment procedure for the company;
- Implementation of S3 monitoring solution.

The contribution of this thesis is a comprehensive security audit of Simple Storage Service using AWS native and 3rd party tools which scale in an environment with several hundred of buckets. Additionally, recommendations are given for improving the S3 security posture by avoiding the most common mistakes in access control mechanisms, implementing the bucket configuration standard and deployment procedure, and applying the MIME type-based whitelisting to prevent malicious scripts from executing after object retrieval. The study is different from other guides as it looks at the security of AWS S3 as a whole in a real environment. Also, the challenges, findings, and solutions are based on experience gained from working at a real company.

1.2. Outline

Chapter 2 gives background information of AWS S3. Section 2.1 describes the main concepts of S3. Section 2.2 lists AWS native security auditing tools and their capabilities in terms of S3. Finally, section 2.3 gives an overview of open source S3 auditing tools (ScoutSuite, Netflix SecurityMonkey) that can be used to audit access control mechanisms and bucket configurations options.

Chapter 3 covers academic papers on S3 security.

Chapter 4 describes the action-research methodology used in chapters 5.1 to 5.4.

Chapter 5.1 concentrates on auditing the AWS S3 bucket configuration options using multiple tools (AWS CLI, Config, Trusted Advisor and ScoutSuite). Subsequently, improvement areas are described based on the audit results. Action items are implemented and finally the changes are evaluated.

Chapter 5.2 focuses on auditing AWS S3 access control mechanisms using AWS CLI and Netflix SecurityMonkey. Identified issues are documented and fixed. Finally, the successfulness of this phase is evaluated.

Chapter 5.3 evaluates Rackspace Compass and Wazuh for Cloudtrail log analysis. The capabilities of tools are described and the successfulness of this phase is evaluated.

Chapter 5.4 analyses a sample of bug bounty program reports that relate to AWS S3 development issues. Subsequently, findings are fixed and changes are evaluated.

Chapter 6 provides an overview of lessons learned in chapters 5.1 through 5.4.

2. Background information

2.1. AWS S3

AWS S3 is an object storage service provided by Amazon Web Services cloud computing platform. It is marketed to be an industry leader at scalability, data availability, security, and performance. S3 is used by companies of all sizes to store different types of data [5].

Object-based storage manages data as objects and differs from classical architectures like file systems and block storages that manage data as file hierarchies and blocks [6]. Buckets created inside S3 are logically separated units which store objects containing data and metadata which are distinguished by unique identifiers. Bucket name must also be globally unique. The name of the object is called the object key. Key name prefixes are used to simulate the concept of folders [7].

2.1.1. Access management

There are multiple ways how to give access to S3 buckets:

- Bucket Access Control Lists;
- Object Access Control Lists;
- Bucket Policies;
- Identity and Access Management policies;
- Signed URL-s.

ACLs can be either configured at bucket or object level. They specify which AWS accounts, users, or groups have a specific type of access. The predefined set of groups can be configured to give access to authenticated users or all users and to log delivery. ACL permissions are READ, WRITE, READ_ACP, WRITE_ACP and FULL_CONTROL which correspond to specific IAM or bucket policy permissions. For example, when the bucket ACL has WRITE permission granted then s3:PutObject and s3:DeleteObject are allowed, and READ_ACP allows s3:GetBucketAcl actions. The bucket and object ACL permissions do not always correspond the same bucket or IAM policy actions. For example, READ ACL at bucket level allows to execute the s3:ListBucket action and the same permission at object level allows to run s3:GetObject [8].

Bucket and IAM policies can be used to set more granular permissions when compared to ACLs. They are in JSON format and specify which actions are allowed or denied at bucket or object level. The main difference between them is that bucket policies are directly attached to the buckets and include a principal (for example IAM user, role or group) whereas IAM policy is directly attached to IAM users, roles or groups. When an asterisk is used in the action part then 92 actions are allowed which include all bucket and object level actions while the bucket and its objects are described in the resource section [9].

Amazon considers ACLs as a legacy access control mechanism and does not recommend using these as these predate IAM. Bucket and IAM polices should be preferred to ACLs because these are more granular. Both policies are written in JSON format and have their own use cases. For example, bucket policies are great when granting access to another AWS account. IAM policies are suited to giving access to employees by directly attaching these to the user or user group. Another popular use case is to attach policies to IAM users allocated to the company's applications that are responsible for object upload and retrieval. Even though the same outcome can be achieved using either approach, it is recommended to consistently use one method [9].

Companies often use signed URL-s as an alternative method to show the contents of private buckets to their customers. For example, when customers want to see their objects, IAM service user's credentials and permissions are used to sign the getObject request with the bucket and object key parameters. The end result is that a temporary URL is generated that allows the customers to see their content [10].

All of these access control mechanisms are evaluated together in the authorisation process (Figure 1). By default, all requests towards the resources are denied. In order to decide whether to deviate from the default, all access control mechanisms are evaluated based on the principle of least-privilege. If there is an explicit *deny* then the decision is to deny the access. If there is not an explicit *deny* then the process continues and checks if there is an *allow*, then the final decision is to allow the request. If there is not an *allow* in the policies, then final decision is again to deny.

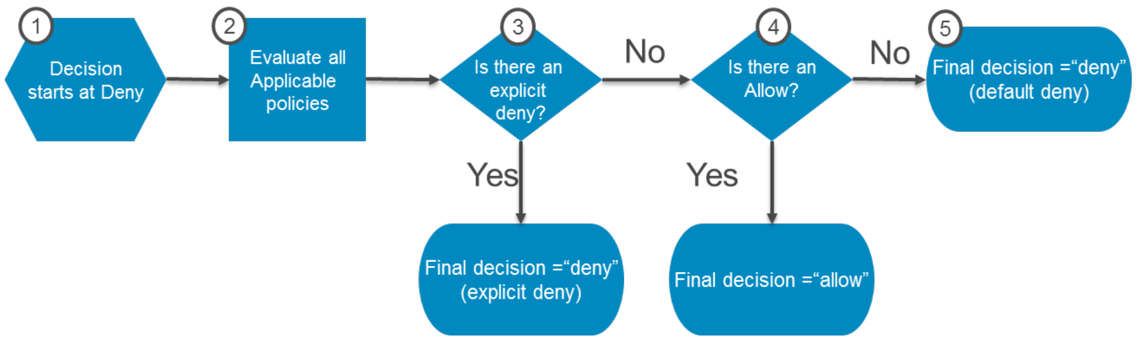
1
Decision starts at Deny
2
Evaluate all Applicable policies
3
Is there an explicit deny?
No
4
Is there an Allow?
No
5
Final decision ="deny" (default deny)
Yes
Final decision ="deny" (explicit deny)
Yes
Final decision ="allow"

Figure 1. Authorization process of all access control mechanisms [9]

2.1.2. Bucket configuration options

Each S3 bucket has its own configuration options. The properties tab configuration options are:

- **Versioning** - Versioning enables to keep multiple versions of an object in one bucket [11];
- **Server access logging** - Server access logging provides detailed records for the requests that are made to the bucket. It is similar to the Cloudtrail's object-level logging, the differences are briefly discussed in the section 2.2 [12];
- **Static website hosting** - Allows host a static website on Amazon S3 [13].
- **Object-level logging** - Records object-level API activity by using CloudTrail data events [14];
- **Default encryption** - Mandates that all objects in a bucket must be stored in encrypted form without having to construct a bucket policy that rejects objects that are not encrypted [15];
- **Object lock** – When enabled the objects cannot be deleted or overwritten [16];
- **Tags** – Can be used for adding more details for billing with AWS cost allocation. Also allows to attach information as a key-value pair that represents a label that is assigned to a bucket [17];
- **Transfer acceleration** - Amazon S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between the client and an S3 bucket [18];
- **Events** - Enables certain Amazon S3 bucket events to send a notification message to a destination whenever the events occur. For example, when objects are created, removed, restored or replicated [19];
- **Requester pays** – The requester (instead of the bucket owner) pays for requests and data transfers when this option is enabled [20].

The permissions tab configuration options are:

- **Public access settings for this bucket** – These settings allow to block access control mechanisms that give public access to the bucket or its objects. This can be also configured at the account level. These settings protect against situations when public access is granted mistakenly. Can be also used to block public access granted using object ACLs. However public access block settings cannot be enabled when a bucket uses deliberately public access to host a website or relies in some other ways on public access [21];

- **Bucket Level Access Control List (ACL)** - Amazon S3 access control lists (ACLs) enable to manage access to buckets and objects. Each bucket and object have an ACL attached to it as a sub resource. It defines which AWS accounts or groups are granted access and describes the type of access [8];
- **Object Level Access Control List (ACL)** - Amazon S3 access control lists (ACLs) enables to manage access to objects. Each object has an ACL attached to it as a sub resource. It defines which AWS accounts or groups are granted access and the type of access [8];
- **Bucket Policy** - S3 bucket policies are attached only to S3 buckets. S3 bucket policies specify what actions are allowed or denied for which principals on the bucket that the bucket policy is attached to (e.g. allow user Alice to PUT but not DELETE objects in the bucket) [9];
- **CORS configuration** - CORS allows client web applications that are loaded in one domain to interact with resources in another domain [22].

The management tab configuration options are:

- **Lifecycle** - Allows to manage objects so that they are stored cost effectively throughout their lifecycle and lets to define rules to delete the objects based on defined schedule. A lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects [23];
- **Replication** - Cross-region replication (CRR) enables automatic, asynchronous copying of objects across buckets in different AWS Regions. Buckets configured for cross-region replication can be owned by the same AWS account or by different accounts [24];
- **Analytics** – The usage Amazon S3 analytics storage class analysis permits to analyse storage access patterns and help to decide when to transition the right data to the right storage class. This new Amazon S3 analytics feature observes data access patterns to helps to determine when to transition less frequently accessed objects from STANDARD storage to the STANDARD_IA (IA, for infrequent access) storage class [25];
- **Metrics** - Amazon CloudWatch metrics for Amazon S3 can help to understand and improve the performance of applications that use Amazon S3 [26];
- **Inventory** - Amazon S3 inventory is one of the tools Amazon S3 provides to help manage the storage. It can be used to audit and report on the replication and encryption status of objects for business, compliance, and regulatory needs. Lets also to simplify and speed up business workflows and big data jobs using Amazon S3 inventory, which provides a scheduled alternative to the Amazon S3 synchronous List API operation [27].

2.2. AWS S3 security auditing tools and services

AWS CloudTrail

CloudTrail is a web service that records AWS API calls and delivers the log files to a S3 bucket. The information includes the identity and the source IP address of the API caller, time of the API call, request parameters (specified separately in the documentation of each AWS service), and the response returned by the AWS service [28]. For example, when someone deletes an S3 bucket then the CloudTrail log event contains the information shown in Table 1 (only relevant fields are included as showing all 57 fields in the example request would be impractical).

Table 1. AWS Cloudtrail S3 log event example about bucket deletion

No.	Field name	Value
1	awsRegion	The name of the destination region (data center)
2	aws_account_id	The ID of target AWS account
3	eventName	DeleteBucket
4	sourceIPAddress	Source IP address of the API caller
5	eventTime	The time of the API call
6	userIdentity.userName	For example, the name of the IAM user who made the request
7	requestParameters.bucketName	The name of the bucket that got deleted

S3 log files can be sent to CloudWatch logs which is an AWS native service for collecting and analysing logs. When Cloudtrail logs are sent to CloudWatch events service then a rule can be created which searches the logs for an event and triggers an action, e.g. sending a notification. These logs can be also analysed by Elastic stack, open source tools, or paid 3rd party services.

AWS S3 Server Access Logging

Server access logging is similar to Cloudtrail data event logs but has to be configured for each bucket separately. While Cloudtrail logs are in JSON format and therefore easily parsable by monitoring tools, server access logs are loosely structured and need a self-written special decoder to parse the logs. Additional issues with S3 server access logs are that the delivery of logs is done on best effort basis (logs might be delivered within a few hours or in some cases not delivered at all) and that some events like bucket creation and deletion are not logged [29]. While S3 server access logs only contain information about bucket events, Cloudtrail can also be used to monitor all AWS account level events as well as bucket and object level events [30].

AWS Config

Config is a native AWS service that enables to assess, audit, and evaluate the configurations of the AWS resources. AWS Config has many uses, but its main benefit comes from the ability to audit resource configuration changes. Simple Notification Service alerts can be triggered when a certain configuration change happens [31]. For example, it is possible to check whether an IAM user has multi-factor authentication configured or not. S3 bucket-related checks are shown in Table 2 [32].

Table 2. AWS Config S3 bucket related checks

No.	Field name	Value
1	Cloudtrail-enabled	Checks whether CloudTrail is enabled or not
2	S3-bucket-blacklisted-actions-prohibited	Checks for blacklisted actions in bucket policies
3	S3-bucket-logging-enabled	Checks the status of bucket logging
4	S3-bucket-policy-not-more-permissive	Checks for policies that are more permissive than the policy provided in this check
5	S3-bucket-public-read-prohibited	Check for buckets that allow public read access through bucket policy or ACL
6	S3-bucket-public-write-prohibited	Check for buckets that allow public write access through bucket policy or ACL
7	S3-bucket-replication-enabled	Checks for bucket replication status
8	S3-bucket-server-side-encryption-enabled	Checks whether bucket policy denies unencrypted put requests
9	S3-bucket-ssl-requests-only	Checks whether policies require SSL for communicating with bucket
10	S3-bucket-versioning-enabled	Checks for the bucket versioning status

AWS Trusted Advisor

Trusted Advisor is a native AWS service that offers a set of best practice checks and recommendations across five categories: cost optimization, security, fault tolerance, performance, and service limits [33]. Trusted Advisor has three AWS S3 checks [34]:

- **Amazon S3 Bucket Permissions (Free)** – Checks for buckets that have open access permissions;
- **Amazon S3 Bucket Logging (Needs higher support plan than basic)** – Checks for bucket logging;
- **Amazon S3 Bucket Versioning (Needs higher support plan than basic)** – Checks whether versioning is enabled on buckets.

AWS Macie

Macie is a security service that analyses and classifies data stored in AWS S3 buckets. It detects personally identifiable information and sensitive data. Additionally, it monitors data access activity with the help of Cloudtrail audit logs [35]. Macie can be also used for detecting public buckets and objects.

2.3. Open source third party S3 auditing tools

Netflix Security Monkey

Security Monkey is an open source tool created by Netflix. It makes an inventory of AWS resources, tracks changes, and has built in checks for auditing. The auditor module identifies potential issues in ACLs, bucket and IAM policies. There is a friendly account detection when multiple AWS accounts have been added to the tool. If friendly accounts are not added, then there will be alarms about unknown cross accounts. Deployment and configuration of Security Monkey takes some time and effort. Some of the functionalities that worked on the development branch did not work on the master branch and vice versa.

NCC Group ScoutSuite

ScoutSuite (previous name Scout2) is an auditing tool that scans through AWS configuration options and highlights problematic areas. It can be easily installed locally and only needs AWS credentials with read permissions to work. The S3 dashboard shows buckets that are publicly accessible in multiple categories. Additionally, the tool detects the buckets that do not have the following configuration options enabled: default encryption, versioning, access logging, and versioning enabled without MFA delete. The dashboard also shows access control-related findings. For example, if buckets or their permissions are world-readable or world-writable and if various actions (delete, get, list, manage, put) are allowed to all principals.

3. Literature Review

This chapter looks at existing work related to AWS S3 security. Firstly, the author presents the reasoning behind the sudden increase of reported breaches over the past few years. Then multiple papers are introduced which talk about the cloud challenges. The studies about cloud penetration testing and shared responsibility model are discussed next. Then the cloud storage broker (CSB) technology is introduced and how automated security auditing can be applied to it. Afterwards the AWS automated reasoning technology for finding policy misconfiguration at scale is introduced. Finally, aspects that academic literature has not yet discussed are brought up.

There are multiple reasons why AWS S3 related security breaches have been more frequent since 2017. Firstly, security researchers and hackers started developing tools that scan for public buckets. Some examples of these tools are: AWSBucketDump, Bucket Finder, Bucket Stream, BuQuikker, inSp3ctor, s3-fuzzer, S3Scanner and Teh S3 Bucketeers [36]. There are even paid services (https://buckets.grayhatwarfare.com/) that search for public buckets and their contents. Secondly, reports of breaches might have increased due to the GDPR requirement to report discovered data breaches within 72 hours, with the failure do so being grounds for a significant fine.

This conference paper [37] brings out that 2017 was the year of data breaches when entities like NSA, Pentagon and Accenture suffered from a data breach due to misconfigured S3 buckets. The work points out that the main cloud challenges revolve around identity and access management, logging and visibility, and incident response. The study indicates that gaining visibility is needed to stop threats and improve the security posture. This can be done by analysing the logs with monitoring tools and creating dashboards, reports and incident response workflows. The paper also shows how the integration of SIEM tool such as Splunk can help to address the visibility problem.

This study [38] looks into cloud security threats, issues, challenges and their solutions. The paper states that the main reason of data loss and leakage in case of cloud providers is lack of authentication, authorization, and access control, weak encryption algorithms, weak keys, risk of association, unreliable data center, and lack of disaster recovery. Also proposes some prevention methods: Secure API, data integrity, secure storage, strong encryption keys and algorithms, and data backup. The study concludes that the cloud users should be aware of cloud vulnerabilities, threats and existing attacks which helps to adopt the cloud technologies in a faster rate.

This paper [39] explores how to conduct penetration testing in AWS using non-authenticated and authenticated approaches. The study suggests using a tool named Sandcastle to conduct bucket enumeration by providing wordlist that might be connected to targeted company. It is possible to tell if the bucket exists or is publicly accessible based on the HTTP response codes shown in the tool. Example case of authenticated penetration test, a tool called WeirdAAL (AWS attack library) was tested in the aforementioned paper which can be used to interact with AWS S3 and other services. For example, this tool helps to determine if there are any mistakes in the access controls by trying out various bucket and object level API calls. This article [40] shows how the access control issues can be also discovered by interacting with S3 API and demonstrates how various configured ACLs impact the security of the buckets and its content.

The authors of this study [41] explain what are the responsibilities of AWS customers in regards to cloud provided services as AWS Elastic Computer Cloud (EC2), Relational Database Services (RDS), S3, Glacier S3 Storage, and DynamoDB. The paper concludes that in the shared responsibility model Amazon clearly provides information about their and its customers security responsibilities. They additionally state that EC2 and S3 require the most customer attention which means that if the shared responsibility model is not understood then serious consequences are prone to happen.

This paper [42] discusses about another approach to storage security by introducing CloudRAID which is a broker solution for cloud storage providers. They tackle security, availability and vendor lock problems by encrypting the original data and distributing the data across multiple cloud storage providers. This study [43] extends the previous paper by introducing CSBAuditor which is an automated security auditing tool for cloud storage brokers which was built due to the fact that existing cloud auditing systems do not solve the challenges related to cloud storage misconfigurations.

There have even been articles [44] stating that S3 security is flawed by design. The article’s author brought up that while all buckets are private by default, it is too easy to misconfigure access control mechanisms. Additionally, Amazon should provide its customers better native security features and not force AWS users to invest more money into alternative solutions to resolve fundamental issues.

In fact, Amazon has started slowly but steadily implementing features to help its customers avoid data breaches with an initiative called Provable Security which researches how to apply automated reasoning technology to detect policy misconfiguration. This paper [45] explains how AWS Zelkova is using math to turn policies into satisfiability modulo theories (SMT) that at the end help detect policy misconfigurations using SMT solvers. Zelkova has been implemented for example into AWS S3, Macie, Config, Trusted Advisor, and GuardDuty. The first Zelkova powered feature in S3 console automatically gave users information if buckets were public or private based on the configured access control mechanisms. It also displayed a warning if it identified objects as public. Unfortunately, this feature did not show which policies and their parts were misconfigured. This paper [46] discusses how formal mathematical methods are easier to use with AWS compared to classic self-hosted microservices. It is easier to check misconfiguration because AWS has its own policy language where Zelkova can be applied. AWS has made their customers' lives easier by introducing one-click formal methods. For example, S3 uses Zelkova to disallow policies that give public access, if the user chooses to block public access in settings.

Zelkova is promising technology and AWS has been working together with other companies to make it more useful. In December 2019, AWS released IAM access analyzer [47] which can be enabled with a few clicks. It uses automated reasoning which in simple terms asks numerous yes-no questions about policies without any user interaction and automatically produces findings about policies that provide external access. A user then has to review the findings and decide whether or not the policies were intentionally configured like this.

A paper by I. Saeed, S. Baras and H. Hajjdiab [48] compares the security and privacy of AWS S3 and Azure Blob storage service. It looks into which compliance standards both cloud service providers support. Additionally, it compares how the features of both storage solutions meet compliance requirements. For example, how identity and access management works, how data is protected in transit and at rest. Although Azure Blob complies with two additional standards (ISO 20000 and ISO 22301) compared to AWS S3, the authors conclude that in the end choosing the right object storage service comes down to business requirements and prior experience. Both of them comply with most commonly accepted standards and also use encryption at data and network level.

The author of this thesis found that there are very few academic papers that discuss the topic of AWS S3 security. Some topics that have not been addressed in academic literature: how to audit bucket configuration options and access control mechanisms at scale, how to deploy and configure buckets and policies to avoid data breaches, how to monitor AWS S3 and how to design upload services securely. AWS has started taking the right steps towards fixing the fundamental issues with Zelkova but there is still a lot of room for improvement.

Action research methodology is generally applied in the social sciences but it can be also used in other fields such as information security or computer technologies. For example, this conference paper [49] used action research methods for improving security in agile processes. Another conference paper [50] applied action research for improving information security policies at medium-sized companies. In the following paper [51] the action research principles were employed to understand the post-breach security changes.

4. Methods

This thesis uses action-research methodology [52] which is used by practitioners to solve problems in a cyclical manner. The number of steps in cycles vary in different action-research examples, the steps in general are plan, act, observe, reflect and repeat. One important aspect of this methodology is that the practitioner involves individuals or groups with a common purpose of improving the current situation. The term phase is used in this thesis instead of cycle.

In this thesis the author assumes the role of security consultant, finds issues related to the company's use of S3, and collaborates with developers and infrastructure teams who apply fixes based on the author's recommendations. The work is divided into several phases. The steps in each phase are:

- Identifying the goals of the phase;
- Deciding which methods will be used to achieve the goals;
- Identification of problems;
- Understanding the current situation by collecting metrics;
- Defining success criteria for the phase;
- Defining action items and implementing them;
- Evaluating the changes.

The steps of the **first phase** are shown in Table 3.

Table 3. Phase 1 – Audit of configuration options

No.	Step name	Explanation
1	Goal	Audit the configuration options of S3 buckets.
2	Tools and methods used	AWS CLI is used to pull the configuration options in a semi-automated way. The documented results are manually reviewed. Semiformal interviews are conducted with infrastructure engineers.
3	Problem identification	Lack of S3 configuration standard and deployment procedure. Bucket default encryption not configured. Bucket tags do not include meaningful information about the bucket. Lack of ticket template for requesting a bucket. Lack of AWS S3 training for engineers. There are unused buckets that need to be removed.
4	Metrics	The ratio of buckets with encryption enabled. The ratio of buckets in AWS production account with tagging configured. The ratio of engineers who have completed the AWS S3 video training. The number of unused buckets deleted.

5	Success criteria	Every bucket should have encryption enabled. Tagging must be configured for every bucket in AWS production account. At least 50% of the engineers must complete the AWS S3 video training. All identified unused buckets must be removed.
6	Action items	Creation of configuration standard and deployment procedure. Creation of bucket requesting template for the engineers. Enabling of bucket default encryption and implementing bucket tagging. Creation of AWS S3 video training. Removal of empty or unused buckets.
7	Change Evaluation	The phase was successful as it met the defined success criteria. Based on the bucket configuration standard, the default encryption and tags were enabled for the 28 buckets in test and 88 buckets in production environment. 125 engineers out of 199 completed the AWS S3 training created by the author of this thesis. 18 empty or unused buckets were removed based on the audit results.

The steps of the **second phase** are show in Table 4.

Table 4. Phase 2 - Access control mechanism audit

No.	**Step name**	**Explanation**
1	Goal	Audit S3 access control mechanisms.
2	Tools and methods used	AWS CLI is used to pull the access control mechanisms in semi-automated way, and the results are documented. Netflix SecurityMonkey's automated findings (issues about the access control mechanisms) are manually reviewed.
3	Problem identification	Multiple buckets contain object ACLs in test and production AWS accounts that give read access to any authenticated AWS user in the world. Multiple bucket ACLs in development AWS account allow either any authenticated AWS user or everyone in the world to do everything, read bucket objects or bucket permissions. All three AWS accounts have IAM policies attached to some service users that allow to do every action against every bucket within the same account. Test and production AWS accounts contain many IAM policies that allow the service users to do every action against a specific bucket. Test and production AWS accounts contain bucket policies that give access to data warehouse team's old AWS account which is no longer used.
4	Metrics	Access control mechanisms' audit results are shown in Table 9. The ratio of fixed critical severity findings. The ratio of fixed high severity findings. The ration of fixed medium severity findings.
5	Success criteria	All critical and high severity access control findings must be fixed. At least 75% of the medium severity findings must be fixed.
6	Action items	Remove the part of code that attaches any authenticated AWS ACL to objects. Also enable blocking public ACLs at the bucket level. Remove bucket ACLs that permit access for everyone in the development environment. Remove IAM users which have policies attached that allow to make any action against all of the buckets within the same AWS account and substitute them with IAM users with more restricted policies. Limit the actions for policies that allow to do every action against a specific bucket. Adjust bucket policies by removing access from data warehouse's old AWS account.
7	Change evaluation	5 critical severity bucket ACLs findings were fixed in the development AWS account and 9 findings were addressed in the rest of the environments. 6 high severity IAM policies were fixed.

		8 medium severity IAM policies were adjusted in the test AWS account and 9 policies were adjusted in the production AWS account. 3 medium severity bucket policies were adjusted. 33 low severity bucket policies with cross account access were fixed, more specifically 21 policies were adjusted and 12 buckets got deleted.

The steps of the **third phase** are shown in Table 5.

Table 5. Phase 3 - AWS S3 monitoring implementation

No.	Step name	Explanation
1	Goal	Find ways how to monitor AWS S3.
2	Tools and methods used	Analysing AWS Cloudtrail audit logs with Rackspace Compass and Wazuh;
3	Problem identification	CloudTrail logs were collected but not analyzed with a monitoring tool.
4	Metrics	The number of AWS S3 events received by Wazuh.
5	Success criteria	All AWS S3 events must be received for further analysis.
6	Action items	Start using AWS CloudTrail and analyze the logs using Compass and Wazuh;
7	Change Evaluation	This phase did not meet the success criteria as the Cloudtrail log monitoring was configured but out of 92 events only 19 were logged into Wazuh.

The steps of the **fourth phase** are shown in Table 6.

Table 6. Phase 4 - AWS S3 development related issues

No.	Step name	Explanation
1	Goal	Fix the development-related issues reported in the bug bounty program.
2	Tools and methods used	Communication with the developers and analysis of the cases.
3	Problem identification	Malicious MIME types can be uploaded using company's file upload service which will execute inline in the browser after opening.
4	Metrics	The ratio of AWS S3 related bug bounty program reports fixed.
5	Success criteria	All of the AWS S3 reports must be fixed.
6	Action items	Implement correct MIME types-based whitelisting.
7	Change Evaluation	This phase met the success criteria as 18 total of AWS S3 related bugs have been fixed, including the 5 issues demonstrated in this thesis.

5. Results

The next step was to conduct semi-structured interviews with the infrastructure team to understand what kind of services they are using in AWS. Multiple team members said that AWS S3 is the most used service inside the company. This meant that the scope of research needed to be narrowed down to the one main service (AWS S3) and other components (bucket configuration options, access control mechanisms, IAM and Cloudtrail) connected to it. Ten AWS accounts owned by company were identified but not all of these contained S3 buckets. Most of the buckets were under development, test, and production accounts owned by the infrastructure team. The data warehouse team's AWS accounts were not included in this study, even though they use S3 to some extent. The same audit methods can also be applied to that team in the future. The final scope of work was to audit the three AWS accounts belonging to the infrastructure team using action research methodology.

As a prerequisite to the audit, the author needed access to the AWS accounts with read-only permissions and credentials for the CLI. Test and production AWS accounts were chosen for data collection as these are completely managed by the infrastructure team while developers have full permissions in the development account.

The analysis part of the thesis will be divided into four subchapters called phases. The first phase is auditing AWS S3 configuration options. Access control mechanisms will be reviewed in the second phase. The next phase will discuss S3 monitoring. The last phase will be an analysis of issues related to AWS S3 development.

5.1. Phase 1 – Audit of configuration options

This chapter discusses the audit of AWS S3 configuration options. Bucket configuration standard and deployment procedure documents are outlined, and action items are implemented based on the aforementioned documents. Finally, the results are evaluated.

5.1.1. Goal

The goal of this phase is to audit bucket configuration options. Firstly, AWS native auditing tools Trusted Advisor and Config were used for the initial assessment. Unfortunately, Trusted Advisor had only two configuration checks for S3 (logging and versioning) which were disabled as the company did not have necessary support plan to use them. AWS Config provided information only about 4 configuration options: Cloudtrail, logging, replication, and versioning. As the native security auditing tools were only able to check a few bucket configurations options, it was necessary to find another approach to collecting information on how the S3 buckets have been configured. Alternative approaches using ScoutSuite and AWS CLI are discussed in the next paragraph.

ScoutSuite (former name Scout2) was also tested to find more insights into problematic areas. The S3 module mostly addressed permission issues which will be discussed in the next phase. The states of the following configuration options were provided by this tool: default encryption, versioning, and bucket access logging. As ScoutSuite was also not able to check all of the bucket configuration options, AWS CLI commands were used instead to get the metrics for the initial configurations shown in Table 7.

5.1.2. Problem identification

Additional interviews were conducted with the infrastructure team to understand the information collected. The aim was to identify the reasoning behind enabling each of the configuration options. It became apparent that bucket configuration was based on developers' requests, whereas this approach is not ideal as not all developers possess the same level of knowledge about AWS. The process for S3 bucket requests by the developers and deployment by the infrastructure team had some flaws. The configuration inconsistencies between all of the environments were happening because developers deployed buckets to the development AWS account (at times also done by infrastructure engineers) and infrastructure engineers deployed to the test and production accounts. Also, it was not agreed who has final say over the configuration. The author decided to create a deployment procedure and configuration standard to harmonise the configuration of current and future S3 buckets and avoid confusion. A ticket template for requesting an S3 bucket was also created to streamline communication between infrastructure engineers and developers.

Another identified issue was that buckets were not mapped to owners and services. This became evident when attempting to obtain clarifications about certain buckets as it required the author to match the company service to the bucket and then find the team who owns the bucket. Surprisingly, some of the teams did not know that they owned a specific AWS S3 bucket.

The configuration options audit revealed that none of the S3 buckets used the default encryption. This is used for encrypting the objects at rest using Amazon S3-managed keys. It is very easy to enable this feature and it does not need any additional configuration or changes.

The total amount of S3 buckets was somewhat concerning. A closer look identified empty and unused buckets that added additional management overhead and increased the attack surface. Furthermore, some test environment buckets were under the production AWS account, thus violating best practices. Tickets were created for the infrastructure team to fix those issues.

5.1.3. Action items and implementation

The following action items were implemented in this phase:

- AWS S3 configuration standard;
- AWS S3 deployment procedure;
- Ticket template for requesting a bucket from the infrastructure team;
- AWS S3 online training;
- Removal of unused buckets;
- Enabling disk encryption on every bucket;
- Attaching tags to the buckets.

AWS S3 configuration standard

The aim of creating this standard was to see if there are any configuration options that could be enabled on every bucket. To this end, an Excel table was created containing every option together with its description.

Then the buckets were categorized into three types in order to give better recommendations:

- Service buckets – used by company services to serve files to the customers;
- Utility buckets – used internally by the infrastructure team;
- Website redirect buckets – used for redirecting various domains to the company's main website.

Each type of bucket received a separate recommendation for every configuration option. The three mandatory options for every bucket were: object level logging, default encryption, and tags. Object level logging was enabled to start logging object level actions using AWS Cloudtrail which can later be used for monitoring. Default encryption was configured to encrypt the object inside the bucket. Tags were added to give extra context about the buckets. The rest of the items in the table were optional and had instructions when to use them as buckets and services may have different requirements.

The configuration standard also included information on how to apply tags to the buckets:

- Owner - team name;
- Type - utility/company service/website redirect;
- Visibility - internal/external;
- Service - company service or internal service name;
- Data type - customer personal data/No customer personal data.

Four different ways to give access to the buckets were explained:

- Bucket policy: Use when access is given to another AWS account. Prefer bucket policy instead of ACL rules when you need everyone to be able to read the object;
- Bucket (and object) ACL rules: As a general rule, AWS recommends using S3 bucket policies or IAM policies for access control. S3 ACLs is a legacy access control mechanism that predates IAM;
- IAM policy: Use when access is given to a service user or an employee;
- Signed URL: Use when a company service is giving access to the company's customers.

Generally, there is little difference whether bucket policy or IAM policy is used. The important part is to start using one mechanism instead of using both interchangeably. This helps make policy management and auditing easier. S3 bucket or object level ACL-s should not be used because they do not allow adding granular permissions and developers then have a possibility to change them if IAM or bucket policy permits this. Additionally, object level ACLs are difficult to audit as there can be thousands of objects in the buckets which might have different ACLs attached. One way is to manually open sample of objects in the bucket and inspect the ACL configuration.

The configuration standard also contained a bucket naming convention (COMPANYNAME-SERVICENAME-ENVIRONMENT-REGION). Lastly, useful links were provided to avoid common mistakes relating to the AWS S3 service.

AWS S3 deployment procedure
The deployment procedure was again divided into categories based on the three types of buckets. It was agreed that developers deploy the company's service buckets to the AWS development account and infrastructure engineers deploy to the test and production environment. However, a new AWS development account was created and will be taken into use in the near future which will be fully managed by the infrastructure team. This will give control over all of the environments to the infrastructure team and ensure that all deployed buckets are configured the same way. Utility and website redirect buckets will be fully deployed by the infrastructure team using an automation tool called Terraform.

Ticket template
The following template was drafted to help submit bucket deployment tickets to the infrastructure team:

- Bucket name;
- Bucket environments;
- Bucket locations;
- New AWS IAM user permissions;
- Existing IAM user permissions;
- Bucket configuration options;
- Public access settings;
- Owner;
- Service name(s);
- Data type;
- AWS Cloudfront, Lambda integration, Bucket events;
- Block public access settings.

AWS S3 video training
The author created an online S3 training. It covered the following topics:

- Short overview of AWS and S3;
- Bug bounty program's S3 bounty statistics;

- Most common S3 mistakes;
- Examples of good and bad IAM policies;
- Dangerous ACLs;
- Overview of S3 configuration standard;
- Overview of S3 deployment procedure.

Removal of unused buckets
The audit revealed that there are some buckets that are empty. The infrastructure team checked the buckets and deleted the ones that are no longer in use. Additionally, there were some buckets where the object count or total size of the bucket had not increased for a long time. These were also checked and deleted as appropriate.

Enabling disk encryption
Default disk encryption was enabled on every bucket because none of them had been configured to use it.

Attaching tags to the buckets.
Firstly, tags were defined in the configuration standard. Then an Excel table was created to collect the tag information for all buckets currently in the production AWS account with the help of developers and infrastructure engineers. Then the results were shared with the infrastructure team who were tasked with applying the tags to the buckets. All future tagging information is provided by developers in the bucket request ticket and applied by infrastructure engineers when creating the bucket.

5.1.4. Metrics

The AWS GUI and CLI were used to get a broader overview of S3 configuration. The states of fifteen configuration options for 28 buckets in test and 88 buckets in production environment were documented into Excel sheets. Most of the information could be retrieved using AWS CLI, however some of the settings were documented directly from the AWS GUI as the CLI did not have commands for these. The results collected from test and production AWS accounts are shown in Table 7. For example, in the first row, 13/28 means 13 of the 28 buckets have the configuration enabled in the test environment. This phase is considered successful when default encryption is enabled for all buckets, tags have been configured for all production buckets, and the rest of the action items presented in the section 5.1.3 are implemented.

Table 7. The status of configuration options in test and production environment

No.	Configuration Option Status	Property Type	Test	Production
1	Versioning	Native	13/28	32/88
2	Server Access Logging	Native	0/28	4/88
3	Static Website Hosting	Native	1/28	22/88
4	Object Level Logging	Native	28/28	88/88
5	Default Encryption*	Native	0/28	0/88
6	Tags*	Native	23/28	22/88
7	Transfer Acceleration	Native	0/28	0/88
8	Events	Native	0/28	0/88
9	Requester Pays	Native	0/28	0/88
10	Bucket Policy	Native	10/28	20/88
11	CORS Configuration	Native	3/28	4/88
12	Empty bucket*	Custom	4/28	27/88
13	Automated deploy	Custom	24/28	20/88
14	Cross-Platform Replication	Native	2/28	4/88
15	Lifecycle	Native	4/28	30/88

* These metrics are evaluated in this phase.

5.1.5. Change evaluation

The improvements made in the AWS S3 bucket configuration audit phase are shown in Table 8. The phase was successful as default encryption was enabled for all buckets and tagging was added to buckets in the AWS production account. Additionally, all action items were implemented.

Table 8. Change evaluation in the configuration audit phase

No.	Change name	Improvments
1	AWS S3 configuration standard	Configuration standard was created.
2	AWS S3 deployment procedure	Deployment procedure was created.

3	Ticket template for requesting a bucket	Template was created.
4	AWS S3 online training	AWS S3 online training was created. From 199 engineers 125 have completed the training.
5	Unused buckets	At least 18 buckets got deleted
6	Default encryption	Default encryption was enabled on 88 production and 28 test account buckets
7	Bucket tags	Tags were assigned to all 88 production buckets

5.2. Phase 2 – Access control mechanism audit

In this chapter the access control mechanisms are audited and fixed. The success of the action items is evaluated at the end.

5.2.1. Goal

The goal of this phase was to audit AWS S3 access control mechanisms. Various methods were used to identify shortcomings. Firstly, the current state of mechanisms was documented using manual UI auditing and AWS CLI. Secondly, AWS native security tools were used to check for issues. Thirdly, open source tools like Security Monkey and ScoutSuite were implemented and the findings were reviewed. All of the three AWS accounts were included in the scope. The following access control mechanisms were audited: S3 bucket policies, bucket & object ACLs and IAM policies. The problematic policies were documented and passed on to the infrastructure team for remediation.

5.2.2. Problem identification

The first step in auditing access control mechanisms was to understand the current situation. AWS Trusted Advisor was used to quickly find out if there are any publicly accessible buckets. There were some access control related issues under the development account where buckets were made public using bucket ACLs (lines 3-5 in Table 9). The test and production accounts had only one result which was a false positive as this bucket serves public websites (line 12 in Table 9).

AWS Config which allows to check if any of the bucket policies or bucket ACLs permit public read or write was also briefly tested. This tool was not very useful at that time as access to S3 is mainly given through IAM policies and there were only a few bucket policies and bucket ACLs in all of the environments. There were a few findings but, other than one false positive due to a bucket serving public content in all environments, the shortcomings were limited to the development account. This tool also did not have functionality to check object ACLs or IAM policies. The search for a tool with functionality to scan IAM policies and preferably also bucket access control mechanism continued.

The next step was to find a better way to scan all access control mechanisms for permission issues. Most of the information was pulled using AWS CLI and missing pieces were added using manual methods by checking from the web console. IAM policies, bucket policies, and bucket ACLs were documented to understand for which purposes particular access control mechanisms were used. The author determined that IAM policies were used to give access either to employees or service users. Bucket policies were used to give access to data warehouse AWS accounts. Bucket ACLs were used to give access to AWS root account and in some cases to log delivery groups.

There was a need to find a better alternative to using AWS CLI as it was time-consuming, and semi-automated. ScoutSuite was tested with read-only AWS permissions. All three environments were scanned. It created HTML reports that gave a simple overview of S3. Nothing critical was found on test and production accounts, meaning that there were no access controls that exposed the bucket or their contents to the Internet. There were some S3 buckets with public access in development account but these were used only for testing and did not contain anything sensitive or were empty. ScoutSuite was a great tool to quickly get AWS S3 overview reports concerning some bucket configuration options and access control mechanisms but it did not produce many findings or actionable items.

Next Netflix Security Monkey was set up to automatically analyse all access control mechanisms. The difference between SecurityMonkey and ScoutSuite was that the ScoutSuite's scan produces every time a new static HTML report about bucket policies and ACLs that give public read or write access, whereas Security Monkey, in addition to the aforementioned ScoutSuite features, was able to continuously scan IAM policies and added extra finding categories like internet-accessible, sensitive permissions, cross-account access etc. It takes up to one hour or even less to set up and configure ScoutSuite as it just needs a read-only IAM user with AWS credentials in all of the AWS accounts and a local Python script to produce results. In the case of Netflix SecurityMonkey, it took the author a day to manually deploy and configure it on the AWS development account. After that it took multiple days for an infrastructure engineer to adjust the script to be able to automatically deploy it to the test and production environments. It needed one AWS instance, one RDS database, and IAM credentials with read-only permissions. All three environments were scanned and various types of issues were found in bucket policies, ACLs, and IAM policies. There were mainly three types of problems: publicly accessible buckets, limited public access, and sensitive permissions. Buckets were made publicly accessible using bucket policy or bucket ACLs. Limited public access was given using object ACLs where specific objects were publicly readable either by everyone in the world or only by authenticated AWS users when the object link was known. Sensitive permissions were mostly about bucket or IAM policies that allowed every action on a specific bucket, every action on all of the buckets, or contained other sensitive actions. Cross-account bucket policy permissions were also detected, fortunately these buckets were owned by the company's data warehouse team.

Netflix Security Monkey is a great open source tool to continuously monitor S3 access control mechanisms. The downside is that it lists multiple issue types per access control mechanism but does not indicate the exact issue in the policy. Every result needs to be manually reviewed to find legitimate issues. For example, one specific access control mechanism like bucket or IAM policy can produce multiple lines of findings about sensitive permissions, unknown and cross-account access etc. Then all the findings should be identified from a single policy and determined if they need to be fixed. Fortunately, the tool did not produce an overwhelming number of findings and it did not take too much effort to analyse these. The analysed findings are shown in Table 9.

5.2.3. Metrics

Table 9 lists all access control-related findings. There were 12 main types of permission issues which were categorized per issue type and policy type. Issues labelled as 'internet accessible' allowed everyone on the Internet access to the bucket. Limited internet access allowed either everyone on the Internet or any AWS user to read a specific object if the object's name was known. Sensitive permissions did not directly expose the bucket or objects to the Internet but permitted the company's service users too many bucket or object-level actions which can be potentially exploited.

The goal for this phase is to eliminate as many permission-related issues as possible. Severity levels are used to categorise the access control issues. The critical level findings concern mechanisms that expose buckets or their contents to the Internet. High level issues are IAM policies that give a specific IAM service user the capability to perform any action against all buckets within one AWS account. Medium level issues concern IAM policies that allow all actions against a specific bucket. Low level issues give access to another AWS account owned by the company. Severity level 'none' is assigned to findings that were intentionally configured to give public access, e.g. for hosting the corporate website or providing a service which needs the objects to be public if the object name is known. This phase is considered successful when all critical-high and 75% of medium level issues are fixed.

Table 9. Access control mechanism issues

No.	Issue	Issue type	Policy type	Severity	Dev AWS Account	Test AWS Account	Live AWS Account
1	Any AWS user can read the objects	Internet Accessible (limited)	Object ACL	Critical	0	5	4
2	Everyone can read a specific object	Internet Accessible (limited)	Object ACL	None	0	3	10
3	Any AWS user can do everything	Internet Accessible	Bucket ACL	Critical	2	0	0
4	Everyone can read bucket permissions	Internet Accessible	Bucket ACL	Critical	2	0	0
5	Everyone can read bucket objects	Internet Accessible	Bucket ACL	Critical	1	0	0
6	Policy allows to perform all actions on any bucket	Sensitive permissions	IAM policy	High	3	2	3
7	Policy allows to perform all actions on a specific bucket	Sensitive permissions	IAM policy	Medium	1	10	11
8	Policy allows to perform all actions on a specific bucket	Sensitive permissions	Bucket Policy	Medium	0	3	0
9	Cross-account read access	Sensitive permissions	Bucket Policy	Low	0	3	19
10	Cross-account write access	Sensitive permissions	Bucket Policy	Low	0	1	0
11	Cross-account read and write access	Sensitive permissions	Bucket Policy	Low	10	0	0
12	Everyone can read the objects	Internet Accessible	Bucket policy	None	1	1	1

5.2.4. Action items and implementation

Bucket ACLs are very often misunderstood and configured incorrectly which may lead to a data breach. For example, access is accidentally given to everyone or to any authenticated AWS user in the world. Bucket level ACL, bucket and IAM policy issues are detected with Netflix Security Monkey, ScoutSuite, manual web console inspection, or AWS CLI. Object ACL issues are not easy to detect because scanners are not able to check them. One way is to review the code that interacts with the bucket. In this thesis, the manual approach was used by opening a sample of objects and inspecting the object ACLs.

Bucket ACLs

Bucket ACL issues were only found in the AWS development environment and are shown in Table 9 on lines 3 through 5. Two S3 buckets had an ACL set to give full control to all authenticated AWS users in the world which can be abused by hackers if they make an authenticated request towards the bucket using their own AWS account. Another two buckets had an ACL which gave all users (everyone in the world) bucket read access and the ability to read the bucket permissions. These issues were documented by the author and addressed by the infrastructure team.

Object ACLs

AWS test and production environments were checked for object ACL issues and the findings are shown in Table 9 on lines 1 and 2. There were mainly two types of issues: any AWS user or everyone in the world can read the object when the direct link is known. The "any AWS user" access is removed because there are no use cases where this would be necessary for the company. "Everyone can read the specific object" access is often used by services to make the file visible to customers and must be reviewed case by case (alternatively bucket policy can be used instead of object ACL).

Object ACLs were attached to the object by the service using the S3 bucket. The best way to fix the ACLs was to block these on the bucket level instead of removing these programmatically as there might be large number of objects in the bucket and every request towards the objects is billed. Any AWS user ACL issues were forwarded to the infrastructure team for remediation. Access for everyone to read the specific objects was not substituted with a bucket policy as such access was intentional.

Bucket Policies

Bucket policy findings are shown in Table 9 on lines 9 to 12. There were three bucket policy findings in the test environment that allowed a specific company service user to perform every action on a bucket. These policies were removed as these are already present in the IAM policies. Test and live environments had cross-account read and write access from three Data Warehouse AWS accounts and access was revoked for one deprecated account. Every environment had a bucket policy which allowed everyone to read the CMS bucket objects, this was considered an exception as it was done on purpose to host the corporate website. When a static website is hosted in an AWS S3 bucket then the objects should be publicly readable to show the content to website visitors.

IAM policies

IAM policy findings are shown in Table 9 on lines 6 and 7. There were mainly two types of IAM policy issues. The biggest problems were found in a couple of policies that allowed to perform every action on any bucket. Additionally, these IAM users were used by multiple company services. These problematic IAM users needed to be removed by creating a new user per company service and limiting the actions. Then there were many policies that allowed a specific IAM user to perform every action. The fix for these policies was to review them, understand which bucket or object level actions are actually needed, and limit the actions as appropriate.

5.2.5. Change evaluation

A comprehensive S3 access control mechanism audit was done in this phase. All findings were documented and tickets were submitted to the infrastructure team with fix recommendations. The most serious problems were found and fixed in the AWS development account, which were fortunately created only for testing purposes and was not connected with any of the company services. There were two types of object ACL issues in the test and live environment as shown in Table 10. The first issue indicated on line 1 potentially exposed the objects to any authenticated AWS users in the world but only when the specific object name was known. In this case the part of code that added this ACL was removed and ACLs were blocked in the bucket settings. The second ACL issue on line 2 which gave read access to everyone was determined to be a false positive as it was set up this way by design and did not need fixing.

The rest of the issues were categorised as sensitive permissions as these did not directly expose the bucket or its objects to the Internet but could still potentially be exploited. For example, when AWS credentials are leaked, developers accidently abuse the credentials, or someone manages to get access to the buckets through a specific company service which caters to the customers. There were mainly two types of sensitive permissions. The first type allowed a specific IAM user to perform every bucket or object level action against all buckets. These IAM users were removed and new ones were created with adjusted policies. The second type gave permission to perform every bucket or object level action against a specific bucket. The affected policies were adjusted to limit actions to those that are actually necessary.

Table 10 shows the total amount of issues found and describes which actions were taken to remedy them. Five critical and six high severity issues mentioned in the metrics section were fixed in this phase which means that the success criteria were met. Additionally, twenty medium level issues were fixed and an additional seven medium severity problems are documented and will be addressed in the near future as each of the finding needs some investigation to adjust the policy to the IAM service user needs. Also, 33 low severity bucket policy issues were fixed.

Table 10. Access control mechanism change evaluation

No.	Issue	Issue type	Policy type	Severity	Dev AWS Account	Test AWS Account	Live AWS Account	Change Evaluation
1	Any AWS user can read a specific object	Internet Accessible (limited)	Object ACL	Critical	0	5	4	ACL adding part was removed from the code. AWS S3 ACL block was enabled for affected buckets.
2	Everyone can read a specific object	Internet Accessible (limited)	Object ACL	None	0	3	10	These findings were not fixed because they were intentionally added this way.
3	Any AWS user can do everything	Internet Accessible	Bucket ACL	Critical	2	0	0	One bucket deleted. One ACL adjusted.
4	Everyone can read bucket permissions	Internet Accessible	Bucket ACL	Critical	2	0	0	One bucket deleted. One ACL adjusted.
5	Everyone can read bucket objects	Internet Accessible	Bucket ACL	Critical	1	0	0	One bucket deleted.
6	Policy allows to perform all actions on any bucket	Sensitive permissions	IAM policy	High	3	2	3	Dev – One IAM user deleted. Ticket created to remedy two remaining issues. Test – One IAM user deleted. One policy adjusted.

								Live – One IAM user deleted. Two IAM users deleted and new IAM users created with adjusted policies.
7	Policy allows to perform all actions on a specific bucket	Sensitive permissions	IAM policy	Medium	1	10	11	Dev – Ticket created to remedy the issue. Test – Eight policies adjusted. Ticket created to remedy the remaining two issues. Live – Two policies deleted. Seven policies adjusted. Ticket created to remedy the remaining two issues.
8	Policy allows to perform all actions on a specific bucket	Sensitive permissions	Bucket Policy	Medium	0	3	0	Policies adjusted.
9	Cross-account read access	Sensitive permissions	Bucket Policy	Low	0	3	19	Test – Three buckets delted Live – Eight buckets deleted. Eleven policies adjusted,
10	Cross-account write access	Sensitive permissions	Bucket Policy	Low	0	1	0	Bucket deleted
11	Cross-account read and write access	Sensitive permissions	Bucket Policy	Low	10	0	0	Bucket policies adjusted
12	Everyone can read the objects	Internet Accessible	Bucket policy	None	1	1	1	No action taken, as this bucket serves public content.

5.3. Phase 3 – AWS S3 monitoring implementation

In this chapter two Cloudtrail monitoring solutions are configured and their capabilities are evaluated.

5.3.1. Goal

The goal of this phase was to review the current AWS S3 log monitoring situation in the development, test and production AWS accounts, document the findings, find improvement areas, and implement them. Firstly, AWS CLI was used to get the configuration of server access logging and object-level logging for all buckets. Server access logging is configured for each bucket separately while object-level logging is managed centrally using Cloudtrail. The differences between these are presented in chapter 2.2. The need to enable or disable these configuration options were reviewed. Secondly, AWS Cloudtrail configuration was reviewed and adjusted for 3rd party monitoring tools as it can be used to get logs about bucket and object levels actions. Thirdly, Cloudtrail log monitoring tools like Rackspace Compass and Wazuh were tested.

5.3.2. Problem identification

The initial need for monitoring API calls against S3 buckets arose after analysing the access control mechanism policies. There were IAM policies that allowed every bucket and object level action against a single bucket. One way to understand what actions IAM service users actually used was to start analysing Cloudtrail logs which would later help to limit the actions in the policies. The analysis of these logs can also be used to improve visibility within the AWS account and help find compliance and security issues.

Although Amazon provides a feature called Event history to search from Cloudtrail trails, it not ideal for analytics as it does not show complete logs, filtering is limited, and it is not possible to create dashboards. Alternatively, Amazon provides guidelines for combining multiple AWS tools to visualize the logs. One of these combines Lambda function, AWS Glue, AWS Athena and AWS Quick Sight together. The second stack combines Amazon Cognito, Kibana, Amazon Elasticsearch service, AWS Lambda, Amazon Cloudwatch and other supporting services together which, in addition to Cloudtrail, supports VPC flow network logs as a log source. As the implementation would have taken time and effort from the infrastructure team and they did not want to onboard another monitoring solution, the 3rd party tools Rackspace Compass and Wazuh were tested instead.

5.3.3. Metrics

The author discovered that AWS S3 logs were collected but no further action was taken. For example, object-level logging was enabled on all buckets due to having at least one Cloudtrail trail configured. It captures bucket-level and object levels API calls against all buckets and stores these in a separate bucket. Server Access logging was found to be configured only for a few buckets at the request of developers and was not implemented for all buckets because Cloudtrail already has a similar functionality built in. The success criterion for this phase is to be able to analyse full Cloudtrail logs. Table 11 shows the number of buckets having server access logging, object-level logging, and Cloudtrail enabled in all AWS environments.

Table 11. AWS S3 logging metrics

AWS Account Type (environment)	Server Access Logging	Object-Level Logging	CloudTrail	Comment
Development	2/73	73/73	1 Trail Enabled	Cloudtrail logging was enabled, but not monitored.
Test	0/49	49/49	1 Trail Enabled	Cloudtrail logging was enabled, but not monitored.
Production	5/70	70/70	1 Trail Enabled	Cloudtrail logging was enabled, but not monitored.

5.3.4. Action items and implementation

Rackspace Compass

The integration between Compass and AWS accounts was configured by Rackspace engineers because they provide it as their managed service. They configured an IAM policy for Compass which gave access to AWS accounts. Additionally, a new Cloudtrail trail was created for the tool. The next step was to understand what kind of functionalities Compass provides for AWS S3.

Rackspace Compass has feature that allows to monitor Cloudtrail and provides alerts based on a predefined list of events. The predefined list provided by Compass only had one S3-related alert which was about IAM policies being changed. Fortunately, it offered the possibility of adding custom alerts. For example, these alerts were tested based on specific Cloudtrail events:

- A new bucket was created;
- Bucket configuration changed;
- Bucket delete actions;
- IAM policies changed.

The alerts feature provided the ability to send notifications to email, SNS topic, PagerDuty, syslog, Slack, and Lambda function. The email integration was tested and alarms were sent to a Slack channel.

There was a separate section for Cloudtrail log. It showed a summary of events by region, service, event name, users, IP, resource, and AWS account. Among the six common searches there was one which allowed to find all activity for a specific IAM user. Unfortunately, this search was very slow and the UI did not help to understand what kind of S3 actions were performed by a specific IAM service user even after filtering was applied. The Cloudtrail events section allowed custom event searches. In this case, the UI provided more options and filters however it still did not satisfy the need to find useful information about buckets or IAM users. Additionally, the search feature was slow and occasionally crashed.

The Rackspace Compass service was unfortunately discontinued in December of 2019. In addition to S3 features, this service added more visibility into all company-owned AWS accounts. It provided some value in terms of AWS Cloudtrail and configuration monitoring however it did not meet the requirements set by the author as the Cloudtrail log search was very limited and did not provide the functionality of creating custom dashboards.

Wazuh

The guide for integrating Cloudtrail with Wazuh is shown in the Appendix 1. Wazuh has a built in Amazon AWS dashboard for monitoring the following AWS services: Cloudtrail, Config, VPC, GuardDuty, Macie, KMS, Inspector and Trusted Advisor. The dashboard was not useful because the pre-built visualizations were designed to give a general overview of all integrations but only CloudTrail was configured at this phase.

The next step was to build a custom dashboard consisting of multiple custom visualizations to understand what kind of events can be seen from Wazuh. For example, the following visualizations were created for counting events:

- data.aws.eventName: Descending (Cloudtrail event names);
- data.aws.eventSource: Descending (AWS services like IAM, S3, KMS etc);
- data.aws.userIdentity.userName: Descending (IAM users);
- rule.level: Descending (Wazuh rule levels).

It was determined that not all Cloudtrail events were logged. Wazuh Amazon rules [53] logging is based on predefined list of events [54]. Wazuh rules and events are shown in Appendix 2. These events were all for bucket level actions and object level actions were not logged due to Wazuh rules which discarded these as these were not in the predefined list of events. For example, in the period of 90 days there were nine types of bucket level events related to the S3 service: DeleteBucket, PutBucketPolicy, PutBucketCors, CreateBucket, PutBucketAcl, PutBucketTagging, PutBucketWebsite, PutBucketLifecycle, PutBucketVersioning, and PutBucketCors. In this thesis only the capabilities of Wazuh Cloudtrail integration were evaluated by creating a few visualizations to see which events Wazuh receives. In the future, the full Cloudtrail log will be sent to Wazuh enabling the author to create actionable dashboards which give a better overview of what is happening in AWS S3.

5.3.5. Change evaluation

In this phase, AWS S3 related monitoring features were tested in Rackspace Compass. It provided a feature that summarised Cloudtrail events and allowed to make simple searches with filtering. Unfortunately, this feature did not meet expectations as it was slow, crashed at times, and did not enable the author to run good searches or create useful dashboards. On the other hand, the event-based alert notification feature worked as expected and allowed to send emails when previously defined AWS events were seen in the Cloudtrail logs. Unfortunately, the Rackspace Compass service was closed down in December 2019 and it also did not meet the success criteria.

Wazuh Cloudtrail integration was configured for development, test, and production AWS accounts. It provided a functionality to automatically pull the Cloudtrail logs which were then analysed based on a predefined list of rules and events. Wazuh's Kibana interface allowed to make log searches and dashboards consisting of custom visualisations. This tool did not meet the success criterion as the Cloudtrail integration was configured but it did not allow to ingest the full Cloudtrail log by default. Only 19 out of 92 AWS S3 related events were received by Wazuh. For example, it was not possible to see object level actions like ‘getObject’ or ‘putObject’ which could be used, for example, to adjust IAM or bucket policies. The ability to ingest the full Cloudtrail logs will be tested in the future.

5.4. Phase 4 – AWS S3 development related issues

In this chapter, a sample of bug bounty cases related to development are presented and fixed.

5.4.1. Goal

The company’s bug bounty program was launched in 2015 and has been active since then. The author joined the company in 2018 and started to test the reported bugs. There have been 18 AWS S3 related bugs that have been reported from 2016 to 2019 and the total sum paid out as bounties for these is 26 800 USD. S3 bugs usually get the highest bounties as they often involve unauthorised access to other customers’ files.

The 18 reports contained:

- 5 bugs that allowed read or write access to the buckets due to incorrect access control mechanism implementation;
- 9 bugs that allowed to run code inline in the browser due to incorrect implementation of content-disposition and file type-based blacklisting;
- 4 bugs that allowed access to other users’ files due to service design issues.

The goal of this phase was to review and fix the AWS S3-related issues that were reported in the company’s bug bounty program during the writing of the thesis. The cases were reviewed and a sample of these was chosen to demonstrate how exploits worked and how bugs were fixed. The chosen bugs were either tested by the author, matched a certain issue category like being able to run code in the browser, or were about service design issues. In these cases, developers were engaged to fix the code. This phase will show that not all of the issues are related to access control mechanisms or bucket configuration options. Some actually are the result of code written by developers.

5.4.2. Metrics

There have been 18 AWS S3-related bugs that have been reported to the company's bug bounty program. A sample of code-related issues is presented in the Table 12 and was chosen to demonstrate development problems. The author was also involved in resolving these cases. The success criterion for this phase is that all of the bugs are fixed.

Table 12. Example of bugs reported to the bug bounty program

Ticket No.	Description	Root cause	Related causes
1	AppCache and Cookie bombing attack allows to retrieve direct S3 file links after the victims' browser has been poisoned and new files are being opened	HTML script files are shown inline in the browser	Improper implementation of file type-based whitelisting and blacklisting
2	Service worker allows to retrieve direct S3 file links when the malicious file is opened and the new files are opened by the victim	HTML script files are showed inline in the browser	Improper implementation of file type-based whitelisting and blacklisting
3	Bypass to ticket nr 2. Certain file extensions still make it possible to run scripts inline.	HTML script files are showed inline in the browser	Improper implementation of file type-based whitelisting and blacklisting
4	XSS can be triggered by uploading SVG files	Files with SVG extensions are shown inline in the browser	Improper implementation of file type-based whitelisting and blacklisting
5	First 1000 files owned by various users can be listed and then downloaded	No user input validation	IAM service user's policy allows ListObject action

5.4.3. Problem identification

Ticket 1

This Proof of Concept (POC) allowed the hacker to upload HTML files to the company's AWS S3 bucket which were used to carry out AppCache and CookieBombing attacks [55]. The POC of this and the next tickets worked only when the hacker had created a trial account, logged in, and started to use the built-in feature of the service that allowed users to upload files. Firstly, the hacker uploaded the fallback HTML file, then extracted the file URL. Secondly, uploaded the cache manifest with the previously extracted URL. Thirdly, uploaded the POC file and included the previous file's URL. The final step was to send the last file's URL to the victim to poison the browser cache. The end result was that the attacker was able to get the direct S3 file links when the victims opened a file. This attack was possible because HTML script files were shown inline in the browser. The company had not correctly implemented MIME type-based whitelisting or blacklisting. Only safe MIME types should be allowed to run inline, unsafe MIME types should be downloaded.

Ticket 2

The second ticket was similar to the first one as it enabled to get access to other users' uploaded files using a service worker. Firstly, the attacker uploaded a service worker script to the files' S3 bucket. The script rewrites a response to an iframe pointing to attacker's Burpcollaborator instance, so when response is rewritten, the attacker receives a callback with referrer containing the full path including the AWS signature. Service workers run forever and only need one initialization. Secondly, the attacker uploaded a XHTML POC file containing the link to the service worker. Thirdly, the attacker shared the XHTML file link to the victim which poisoned the victim's browser. Finally, the attacker was able to receive AWS S3 file links when the victim opened any of the files. This attack was possible because the XHTML script file was shown inline in the browser.

Ticket 3

The third ticket is a bypass to the second ticket. The hacker found that there are other file extensions in addition to XHMTL that are shown inline and allow to execute the POC. This meant that having a MIME type-based blacklist was not a viable solution as it is difficult to maintain. Additionally, the hacker found a second files' API endpoint which allowed the execution of the POC of ticket 2.

Ticket 4

The fourth ticket enabled to upload an SVG file which triggered an XSS alert within the browser. This demonstrates again that only whitelisted MIME types should be allowed to run inline and everything else should be blacklisted.

Ticket 5

The fifth ticket demonstrated how the hacker was able to list the first 1000 objects (files) belonging to other users from a single S3 bucket and then download them. This was possible due to fact the there was no input validation in the POST request and the IAM service user's policy allowed 'ListObjects' action, so the hacker was able to create a request to list all of the files instead of his own. The first 1000 objects limitation came from the AWS API.

5.4.4. Action items and implementation

The POCs of tickets 1 through 4 worked mainly because the MIME type whitelist and blacklist were used together. When an unsafe MIME type was reported then it was added to the blacklist. The solution was to start using only a whitelist for safe MIME types that needed to be shown inline in the browser. For example, if the MIME type is image/JPG then the application adds ContentType as image/JPG and ContentDisposition as inline to the AWS S3 object metadata which tells the browser to show the object inline instead of downloading the image. If the MIME type is not in the whitelist, then the ContentType will be always application/octet-stream and the ContentDisposition will be attachment which tells the browser to download the file instead of showing it inline. This way the malicious file types will be always prompted to download, and the scripts will not run inside the user's browser when a file is opened.

In case of ticket 5 the fix consisted of two parts. The first part was to remove the IAM user whose policy allowed any action against all of the company-owned buckets within the production AWS account. A new user was created with a policy that allowed only necessary AWS S3 bucket and object level actions against a single bucket. As the new user's IAM policy did not contain the ListObjects permission then it was no longer possible to list the first 1000 objects. The second part of the fix was about adding input validation to the API requests to eliminate the possibility to list the objects and then later download them.

5.4.5. Change evaluation

Tickets 1 through 4 from the bug bounty program helped the company notice the issues related to allowing script files running inline in the browser. At first, the issues were fixed by implementing a blacklist for unsafe MIME types but this was not a viable long-term solution as bad file types continued to be discovered by the hackers. Later a whitelist for safe MIME types was built which allowed to show certain files inline in the browser and prompted a download of the MIME types that were not in the list.

Ticket 5 proved that an IAM service user with an excessively permissive policy found in phase two can be abused. The fix was to create a separate user with a policy that allowed access to only one bucket with limited actions. Additionally, the developers added input validation to the files service which fixed the part that allowed to manipulate the API requests to list the objects and later download them. No new AWS S3 related tickets have been submitted after the last fixes were introduced in 15.07.2019. In October 2019 additional penetration testing was conducted by a 3rd party against the S3 service and no issues were found. This phase was successful as all of the S3 bug bounty reports were fixed.

6. Discussion

Reviewing all the configuration options is not an easy task if there are many AWS S3 buckets within multiple AWS accounts. Even though AWS does not have built-in functionality to see all of the configuration options for all buckets from a single place (each of the buckets needs to be opened one by one too see the configuration), there are still some AWS native tools like Trusted Advisor and Config that will quickly give an overview of a few essential options that have been enabled or disabled. Alternatively, NCC Group's ScoutSuite can be used to get a quick HTML-based security report. In addition to the checks that AWS native tools have, ScoutSuite has further S3 security best practice checks. The aforementioned tools do not show all of the bucket configuration options (a list of the options can be found in section 2.1.2). One solution is to use AWS CLI to get the desired bucket properties for all buckets. This is still considered a semi-automated method as each of the properties must be pulled separately and documented unless scripting is used. Running AWS CLI commands manually is a good solution if all of the properties need to be pulled only once because the documented results will get outdated as soon as changes are introduced to the buckets, meaning that the same time-consuming semi-automated process needs to be done again to see what has been changed. Boto3 (AWS SDK for Python) would be a good choice for scripting which enables to pull all of the configuration options for example to a Confluence or Excel table. The creation of the script can take some time but getting a snapshot of the current situation later will then be much easier and faster than using AWS CLI commands one by one. None of the tested tools had the functionality to get a single page report of all configuration options for all of the buckets. For example, ScoutSuite concentrates only on showing findings based on predefined checks that help to determine the most critical issues but there is a need for a tool that enables to audit all configuration options.

The next step after documenting the initial status of the bucket configuration options is to understand their purpose and how they work. Some of the options are rarely used and have very specific use cases. A good way to start evaluating if there are options that should be configured is to find patterns from the initial setup and ask questions why something is enabled or disabled. One approach is to create a configuration standard which defines properties that should be configured for all buckets. For example, object level logging, default encryption, and tags. It is likely that only a few options can be applied to all buckets. In this case, if the buckets can be categorised based on their purpose, data, or some other characteristic, then a standard can be created for each bucket group. For example, in the company where the author works, the categories were buckets serving the customers, buckets used only by the employees, and website buckets. There might be still some buckets with properties that deviate from the standard but this is acceptable as it is not always possible to standardise the edge cases.

Auditing and fixing access control mechanisms like ACLs, bucket and IAM policies is another significant topic. All of these mechanisms can be used together and the access to a specific bucket is evaluated in the authorisation process. When the number of buckets exceeds a threshold, manual inspection is no longer feasible and a way to automate at least some of the work must be found. AWS CLI can be used to quickly pull bucket policies and ACLs but getting object ACL-s is more complicated as there can be millions of objects inside a bucket. In this case, a sample of objects can be manually inspected in each of the buckets. AWS CLI can be also used to get IAM policies attached to the IAM users. In this case the policies do not always contain AWS S3-related access and they need to be extracted manually. Analysing data pulled from the CLI requires a good understanding of the access control mechanisms and an ability to see problematic areas in the policies. Inevitably, it also takes a significant amount of time. The only tool that was able to continuously analyse the policies and produce actionable findings was Netflix SecurityMonkey which is unfortunately no longer supported. The codebase maintainers recommend using NCC Group's ScoutSuite instead. In general, there is a need for a better tool for auditing access control mechanisms while Amazon continues its efforts to make Zelkova-based automated reasoning technology more useful for their customers.

The major red flags are usually related to access controls that give public access to everyone or to every authenticated AWS user in the world, unless that was the intention. For example, ACLs use AllUsers or AuthenticatedUsers and policies use asterisk for the principal value. A bit less critical but still troubling IAM policies are the ones that use an asterisk in the resource value, giving access to all of the buckets inside a single AWS account. Sensitive permissions are also related to bucket or IAM policies which have an asterisk in the action field, meaning that all of the bucket or object level actions are allowed depending on whether the resource field contains only the bucket or also the objects. Another potential issue to check is if there are policies that give access to another AWS account, if these are still needed, and if these were given intentionally.

Setting up Cloudtrail log monitoring can be also challenging as there are multiple ways to do it. For example, one could use the AWS centralized logging stack which combines Amazon Elasticsearch and Kibana with other AWS supporting services to provide the monitoring solution. Another option is to deploy a self-managed Elastic stack if the AWS managed stack is not suitable. There are many different commercial services (Sumo Logic, Datadog, New Relic, AlienVault USM Anywhere, Solarwinds Loggly, Dynatrace etc.) that already have built-in dashboards based on Cloudtrail event types. As it was already deployed, the author tested Wazuh which is essentially Elastic stack with Wazuh rulesets. The problem with the default configuration of Wazuh is that it logs only such CloudTrail events that are in the predefined list and the prebuilt AWS dashboard is not useful at all. So, the visualisations and dashboards need to be created by the user. Full log analysis can be done when the Wazuh AWS ruleset is configured to log all Cloudtrail events. This helps track compliance, conduct security analysis, and troubleshoot issues with the IAM service users. It also gives better overall visibility into what is happening not only with AWS S3 but also with other services within one AWS account. Overall, when there is a need to analyse the full Cloudtrail log and customize dashboards then Elastic stack is a good solution for it, otherwise any 3rd party tool or service is also capable of showing the most important events that are happening within the AWS account.

Bug bounty programs like Bugcrowd and HackerOne can help to identify AWS S3-related issues with the help of ethical hackers before real breaches happen. Usually these reports get the highest bounties because buckets contain customer data but compared to the reputation loss when a real breach happens it is still cheaper to pay these bounties. When a report comes in about one misconfigured AWS S3 bucket then it recommended to check if other buckets are also affected. Occasionally hackers find bypasses to previously resolved issues when the initial fix had not taken into consideration the other exploit methods. An example of this would be when new file extensions are reported that allow to run scripts inline in the browser to steal other customer’s files. Most S3-related reports are fixed by infrastructure engineers but sometimes developers also need to adjust their code. For instance, they might need to add input validation for the service that is responsible for file uploads and retrieval, adjust object metadata, or remove the part of code that adds problematic ACLs to the objects.

One way to ensure that new buckets are configured correctly is to define an AWS S3 configuration standard and deployment procedure. Subsequently, training on the documents should be provided to developers and the infrastructure team. The training could also include information on the most common mistakes and how to avoid them. Additionally, it is good to have a ticket template for bucket requesters with guiding questions. Mistakes can be also limited when bucket deployment and access control mechanism configuration is done by infrastructure engineers because it more difficult to control permissions when the developers assign ACL rules to the bucket or objects. This means that adjusting or troubleshooting the access control policies requires good coordination between infrastructure engineers and developers. Not all of the mistakes are fixed by the infrastructure team – at times developers need to adjust their code which interacts with S3 buckets. For example, they might be required to disallow malicious MIME types to be run inline in the browser by adjusting the object metadata. Auditing and monitoring can also help to validate that buckets and permissions are set up correctly. Advanced companies can also build automations using AWS native tools (for example AWS Lambda and SNS) that continuously check bucket and policy configurations in order to notify and remedy issues that do not meet the company's defined requirements.

The main concepts of this thesis can be applied also for other object storage solutions. Auditing configuration properties and access controls is necessary to secure the storage service and to comply with standards. For instance, ScoutSuite enables to scan the following cloud providers in addition to AWS: Azure, Google Cloud Platform, Aliyun (Alibaba Cloud), Oracle Cloud Infrastructure, and OpenStack. The object storage configuration standard and deployment procedure can also be applied to other providers. The configuration properties and access controls might not be exactly the same but settings for encrypting data at rest and in transit should be there. If other object storage solutions are able to produce logs, then they can also be analysed, for example with Elastic stack. In case of every object storage provider the company's service responsible for the files should be developed securely to avoid data leaks or other malicious uses.

7. Summary

When companies use AWS S3 as their object storage solution, they often fail to notice the importance of security and privacy which, in turn, can lead to data breaches. Simple mistakes in access control mechanisms can expose bucket contents to the public. Moreover, not much attention is paid to bucket configuration settings which can be used further to improve security and compliance. Even when companies have properly configured their buckets and policies, the significance of S3 monitoring is usually underestimated.

The goal of this thesis was to increase the security of AWS S3 in a real medium-sized company. Action-research methods were used, meaning that the author assumed the role of security consultant, audited S3, gave recommendations on how to fix the issues, and ensured that the identified problems were addressed. The work was divided into four phases concerning the following aspects of S3: bucket configuration options, access control mechanisms, Cloudtrail monitoring, and development related mistakes.

In the first part of the thesis, bucket configuration options were audited in the test and production AWS accounts. During this process AWS native security tools (Trusted Advisor and Config) and a 3rd party tool called ScoutSuite were tested but none of them provided an overview of all of the bucket configuration options. For this reason, AWS CLI commands were used instead to get information from the S3 API. Based on the documented results, a configuration standard was created with recommended options for three bucket categories (service, utility, website redirect) while default encryption, object-level logging, and tags were enabled for all buckets. Tagging was used to specify the bucket owner, the name of the internal or external facing service, and whether the bucket contained customer personal data. This information could be later used for more effective incident response and for troubleshooting purposes. Additionally, a bucket deployment procedure was formulated and template for requesting a bucket was created. Subsequently, a video training concerning the aforementioned documents and most common AWS S3 policy mistakes was created.

In the second part of the thesis, the audit of all AWS S3 access control mechanisms was conducted in the development, test and production AWS accounts using manual inspection and AWS CLI at first and then Netflix SecurityMonkey (now unfortunately discontinued). All critical and high severity ACLs in test and production accounts which gave access either to everyone in the development account or any authenticated AWS user in the world were fixed. Also, medium level findings like polices that gave certain IAM service users permissions to do any action against all of the buckets under one account were removed. Additionally, most of the low-level findings which allowed a specific IAM user to do any action against a specific bucket were fixed.

In the third part of the thesis, CloudTrail log monitoring was configured using two tools. Rackspace Compass features were evaluated but unfortunately the service was closed down during the final stages of writing this thesis. Additionally, a tool named Wazuh was tested which by default did not allow to analyse the full Cloudtrail log and therefore the goal was met only partially.

The fourth part of the thesis provided statistics on AWS S3-related bug bounty reports. A sample of these which the author helped resolve was chosen for demonstration. The issues were mostly development related where scripts were normally uploaded to the files bucket and subsequently executed inline in the browser when opened, allowing the malicious actor to steal other users' files. These issues were fixed by implementing a whitelist for safe MIME types that can be shown inline after opening and prompting all other MIME types to be downloaded.

The main contribution of this thesis was a thorough security audit of Simple Storage Service at a medium-sized company using AWS and 3rd party tools which help to reduce manual efforts in an environment with hundreds of buckets. In addition, recommendations were given on how to enhance S3 security by steering clear of the most common mistakes in access controls, applying a bucket configuration standard and deployment procedure, and implementing MIME type-based whitelisting. The study is different from best practice guides as it examined the security of AWS S3 as a whole, documented the findings based on metrics and showed the implementation results in a real environment. The challenges, findings and solutions were based on the experience received from working in a real company.

Future work in this area would involve testing if Wazuh is capable of receiving the full Cloudtrail log and building dashboards for getting a better overview of AWS S3 events. Also, the remaining sensitive IAM policies will be fixed. Lastly, the IAM access analyser is configured and the findings regarding policies that give access to external entities are reviewed.

References

[1] E. Chickowski, "Leaky Buckets: 10 Worst Amazon S3 Breaches," 28 January 2018. [Online]. Available: https://businessinsights.bitdefender.com/worst-amazon-breaches. [Accessed May 2020].

[2] D. O'Sullivan, "The RNC Files: Inside the Largest US Voter Data Leak," 19 June 2017. [Online]. Available: https://www.upguard.com/breaches/the-rnc-files. [Accessed May 2020].

[3] D. Lee, "Uber concealed huge data breach," 22 November 2017. [Online]. Available: https://www.bbc.com/news/technology-42075306. [Accessed May 2020].

[4] UpGuard, "Data Warehouse: How a Vendor for Half the Fortune 100 Exposed a Terabyte of Backups," 27 June 2019. [Online]. Available: https://www.upguard.com/breaches/attunity-data-leak. [Accessed May 2020].

[5] Amazon Web Services, "Amazon S3," 2019. [Online]. Available: https://aws.amazon.com/s3/. [Accessed April 2019].

[6] Y. León and T. Piscopo, "Object Storage versus Block Storage: Understanding the Technology Differences," September 2019. [Online]. Available: https://www.druva.com/blog/object-storage-versus-block-storage-understanding-technology-differences/. [Accessed May 2020].

[7] Amazon Web Services, "Object Key and metadata," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMetadata.html. [Accessed April 2019].

[8] Amazon Web Services, "Access Control List (ACL) Overview," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html. [Accessed April 2019].

[9] K. Zhao, "IAM Policies and Bucket Policies and ACLs! Oh, My! (Controlling Access to S3 Resources)," January 2019. [Online]. Available: https://aws.amazon.com/blogs/security/iam-policies-and-bucket-policies-and-acls-oh-my-controlling-access-to-s3-resources/. [Accessed April 2019].

[10] T. Sallai, "How S3 Signed URLs work," 30 October 2018. [Online]. Available: https://advancedweb.hu/2018/10/30/s3_signed_urls/. [Accessed April 2019].

[11] Amazon Web Services, "Using Versioning," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html. [Accessed April 2019].

[12] Amazon Web Services, "Amazon S3 Server Access Logging," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html. [Accessed April 2019].

[13] Amazon Wen Services, "Hosting a Static Website on Amazon S3," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html. [Accessed April 2019].

[14] Amazon Web Services, "How Do I Enable Object-Level Logging for an S3 Bucket with AWS CloudTrail Data Events?," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/user-guide/enable-cloudtrail-events.html. [Accessed April 2019].

[15] Amazon Web Services, " Amazon S3 Default Encryption for S3 Buckets," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-encryption.html. [Accessed April 2019].

[16] Amazon Web Services, "Introduction to Amazon S3 Object Lock," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/object-lock.html. [Accessed April 2019].

[17] Amazon Web Services, "Object Tagging," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/object-tagging.html. [Accessed April 2019].

[18] Amazon Web Services, "Amazon S3 Transfer Acceleration," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/transfer-acceleration.html. [Accessed April 2019].

[19] Amazon Web Services, "Configuring Amazon S3 Event Notifications," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html. [Accessed April 2019].

[20] Amazon Web Services, "Requester Pays Buckets," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/RequesterPaysBuckets.html. [Accessed April 2019].

[21] J. Barr, "Amazon S3 Block Public Access – Another Layer of Protection for Your Accounts and Buckets," 15 November 2018. [Online]. Available: https://aws.amazon.com/blogs/aws/amazon-s3-block-public-access-another-layer-of-protection-for-your-accounts-and-buckets/. [Accessed April 2019].

[22] Amazon Web Services, "Cross-Origin Resource Sharing (CORS)," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/cors.html. [Accessed April 2019].

[23] Amazon Web Services, "Object Lifecycle Management," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html. [Accessed April 2019].

[24] Amazon Web Services, "Cross-Region Replication," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/crr.html. [Accessed April 2019].

[25] Amazon Web Services, "Amazon S3 Analytics – Storage Class Analysis," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/analytics-storage-class.html. [Accessed April 2019].

[26] Amazon Web Services, "Metrics Configurations for Buckets," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/metrics-configurations.html. [Accessed April 2019].

[27] Amazon Wen Services, " Amazon S3 Inventory," 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/storage-inventory.html. [Accessed April 2019].

[28] Amazon Web Services, Inc, "DevOps and AWS," 2018. [Online]. Available: https://aws.amazon.com/devops/. [Accessed January 2018].

[29] J. Hwong, "AWS S3 Logjam: Server Access Logging vs. Object-level logging," 30 July 2019. [Online]. Available: https://www.netskope.com/blog/aws-s3-logjam-server-access-logging-vs-object-level-logging. [Accessed May 2020].

[30] J. Naglieri, "AWS Security Logging Fundamentals — S3 Bucket Access Logging," 8 January 2020. [Online]. Available: https://medium.com/panther-labs/aws-security-logging-fundamentals-s3-bucket-access-logging-93099ab80e38. [Accessed May 2020].

[31] Amazon Web Services, Inc, "AWS Config," 2018. [Online]. Available: https://aws.amazon.com/config/. [Accessed 2018].

[32] Amazon Web Services, "List of AWS Config Managed Rules," 2019. [Online]. Available: https://docs.aws.amazon.com/config/latest/developerguide/managed-rules-by-aws-config.html. [Accessed April 2019].

[33] Amazon Web Services, "AWS Trusted Advisor Best Practice Checks," 2019. [Online]. Available: https://aws.amazon.com/premiumsupport/technology/trusted-advisor/best-practice-checklist/. [Accessed April 2019].

[34] Amazon Web Services, " AWS Trusted Advisor Best Practice Checks," 2019. [Online]. Available: https://aws.amazon.com/premiumsupport/technology/trusted-advisor/best-practice-checklist/. [Accessed April 2019].

[35] T. Walker, "Launch – Hello Amazon Macie: Automatically Discover, Classify, and Secure Content at Scale," 17 August 2017. [Online]. Available: https://aws.amazon.com/blogs/aws/launch-amazon-macie-securing-your-s3-buckets/. [Accessed April 2019].

[36] Linux Security Expert, "Amazon S3 bucket scanners," May 2020. [Online]. Available: https://linuxsecurity.expert/security-tools/amazon-s3-bucket-scanners. [Accessed may 2020].

[37] E. Han, V. Urias, B. P. Van Leeuwen, W. M. Stout, G. K. Kao and H. W. Lin, "It's Raining Clouds: Maintaining Visibility in the Haze," Nashville, 2018.

[38] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *Journal of Network and Computer Applications,* vol. 79, pp. 88-115, 2016 November 2017.

[39] R. Szabo, "Penetration testing of aws-based environments," November 2018. [Online]. Available: http://essay.utwente.nl/76955/1/Szabo_MSc_EEMCS.pdf. [Accessed May 2020].

[40] F. Rosen, "A deep dive into AWS S3 access controls – taking full control over your assets," 13 July 2017. [Online]. Available: https://labs.detectify.com/2017/07/13/a-deep-dive-into-aws-s3-access-controls-taking-full-control-over-your-assets/. [Accessed may 2020].

[41] K. Bennett and J. Robertson, "Security in the Cloud: understanding your responsibility," in *Cyber Sensing 2019*, Baltimore, 2019.

[42] M. Schnjakin and C. Meinel, "Evaluation of Cloud-RAID: A Secure and Reliable Storage above the Clouds," in *22nd International Conference on Computer Communication and Networks (ICCCN)*, Nassau, 2013.

[43] M. I. H. S. T. S. H. G. F. C. a. C. M. K. A. Torkura, "CSBAuditor: Proactive Security Risk Analysis for Cloud Storage Broker Systems," in *IEEE 17th*

International Symposium on Network Computing and Applications (NCA), Cambridge, 2018.

[44] K. Sen, "S3 Security Is Flawed By Design," 23 November 2018. [Online]. Available: https://www.upguard.com/blog/s3-security-is-flawed-by-design. [Accessed May 2020].

[45] J. Backes, P. Bolignano, B. Cook, C. Dodge, A. Gacek, K. Luckow, N. Rungta, O. Tkachuk and C. Varming, "Semantic-based Automated Reasoning for AWS Access Policies using SMT," in *Formal Methods in Computer Aided Design (FMCAD)*, Austin, 2019.

[46] J. Backes, P. Bolignano, B. Cook, A. Gacek, K. Luckow, N. Rungta, O. Tkachuk and C. Varming, "One-Click Formal Methods," 22 October 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8880058. [Accessed May 2020].

[47] B. West, "Identify Unintended Resource Access with AWS Identity and Access Management (IAM) Access Analyzer," 2 December 2019. [Online]. Available: https://aws.amazon.com/blogs/aws/identify-unintended-resource-access-with-aws-identity-and-access-management-iam-access-analyzer/. [Accessed May 2020].

[48] I. Saeed, S. Baras and H. Hajjdiab, "Security and Privacy of AWS S3 and Azure Blob Storage Services," in *IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, Singapore, 2019.

[49] T. Oyetoyan, B. Milosheska, M. Grini and D. Cruzes, "Myths and Facts About Static Application Security Testing Tools: An Action Research at Telenor Digital," in *19th International Conference XP 2018*, Porto, 2018.

[50] H. Kinnunen and M. Siponen, "Developing Organization-Specific Information Security Policiesby using Critical Thinking," in *Pacific Asia Conference on Information Systems*, 2018.

[51] A. Demjaha, T. Caulfield, A. Sasse and D. Pym, "2 Fast 2 Secure:A Case Study of Post-Breach Security Changes," in *2019 IEEE European Symposium on Security and Privacy Workshops*, Stockholm, 2019.

[52] V. Koshy, "Action Research for Improving Practice," 2005. [Online]. Available: https://dl.uswr.ac.ir/bitstream/Hannan/132060/1/Valsa_Koshy_Action_Research_for_Improving_Practice_A_Practical_Guide__2005.pdf. [Accessed May 2020].

[53] Wazuh, "0350-amazon_rules.xml," April 2020. [Online]. Available: https://github.com/wazuh/wazuh-ruleset/blob/master/lists/amazon/aws-eventnames. [Accessed April 2020].

[54] Wazuh, "AWS-eventnames," November 2019. [Online]. Available: https://github.com/wazuh/wazuh-ruleset/blob/master/lists/amazon/aws-eventnames. [Accessed April 2020].

[55] F. Rosén, "Bypassing and exploiting Bucket Upload Policies and Signed URLs," 2 August 2018. [Online]. Available: https://labs.detectify.com/2018/08/02/bypassing-exploiting-bucket-upload-policies-signed-urls/. [Accessed April 2020].

[56] Amazon Web Services, Inc., "Native AWS Security-Logging Capabilities," 30 March 2017. [Online]. Available: https://d1.awsstatic.com/aws-answers/AWS_Native_Security_Logging_Capabilities.pdf. [Accessed January 2018].

[57] Amazon Web Services, Inc, "Server Access Logging," [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html. [Accessed 2018].

Appendix 1 – Wazuh's Cloudtrail integration guide

The configuration steps for Wazuh Cloudtrail integration in all of the AWS accounts:

1. Install necessary dependencies in Wazuh manager which are needed for communication with AWS;

```
# apt-get update && apt-get install python-pip
# pip install boto3
```

2. Create a new Cloudtrail trail;

```
Apply trail to all regions: yes
Read/Write events: all
Select  all  S3  buckets  in  your  account:  read,  write
Create a new bucket for this trail
```

3. Create a new IAM user;
4. Create and attach the following policy;

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::CLOUDTRAIL-BUCKET-ENV",
                "arn:aws:s3:::CLOUDTRAIL-BUCKET-ENV/*"
            ]
        }
    ]
}
```

5. Create AWS credentials for the IAM user;
6. Change ossec.conf in the Wazuh manager;

```
Test environment example:
<wodle name="aws-s3">
    <disabled>no</disabled>
    <interval>10m</interval>
    <run_on_start>yes</run_on_start>
    <skip_on_error>yes</skip_on_error>
    <bucket type="cloudtrail">
      <name>CLOUDTRAIL-BUCKET-ENV</name>
      <path>s3</path>
      <access_key>PLACEHOLDER</access_key>
      <secret_key>PLACEHOLDER</secret_key>
    </bucket>
  </wodle>
```

7. Go to Wazuh extension settings in Kibana and enable „Amazon AWS";
8. If everything works correctly then the Cloudtrail logs will be pulled to Wazuh based on predefined Wazuh Amazon ruleset.

Appendix 2 – Amazon ruleset and predefined AWS events in Wazuh

These examples demonstrate how these rules work. Each rule is depending on previous rule which is shown by increasing the indentation:

```
Parent rule (<rule id="80200" level="0">) - This rule enables
Wazuh to read Cloudtrail json logs, but does not log them to
Wazuh due to rule level being 0.
   Child rule (<rule id="80202" level="3">) - This rule
matches eventnames from the predefined list and logs them to
Wazuh.
       Child rule (<rule id="80203" level="4">) - Increases
the rule level and changes the description when there is an
error related to an AWS event.
           Child rule (<rule id="80250" level="5">) -
Increases the rule level when there are events where access
is denied to some resources.
```

```
Parent rule (<rule id="80200" level="0">) - This rule
enables Wazuh to read Cloudtrail json logs, but does not
log them to Wazuh due to rule level being 0.
    Child rule (<rule id="80202" level="3">) - This rule
matches eventnames from the predefined list and logs them
to Wazuh.
        Child rule (<rule id="80251" level="3">) - This
rule logs down events related to object deletions.
            Child rule (<rule id="80252" level="10"
frequency="22" timeframe="600">) level="3">) - This rule
increases the rule level when there has been 22 object
deletion events in 10 minutes.
```

```
  Parent rule (<rule id="80200" level="0">) - This rule
enables Wazuh to read Cloudtrail json logs, but does not
log them to Wazuh due to rule level being 0.
    Child rule (<rule id="80202" level="3">) - This rule
matches eventnames from the predefined list and logs them
to Wazuh.
        Child rule (<rule id="80253" level="3">) - This
rule logs AWS account successful login attepts.
            Child rule (<rule id="80254" level="5">) - This
rule logs AWS account unsuccessful login attepts.
                Child rule (<rule id="80255" level="10"
frequency="6" timeframe="360">) - This rule triggers when
```

```
there has been 6 failed login attepts to AWS account within
6 minutes.
```

Here are some examples of Wazuh predefined events that can be used to monitor AWS S3:

- CreateBucket:S3;
- DeleteBucket:S3;
- DeleteBucketLifecycle:S3;
- DeleteBucketReplication:S3;
- DeleteBucketTagging:S3;
- PutBucketLifecycle:S3;
- PutBucketLogging:S3;
- PutBucketNotification:S3;
- PutBucketReplication:S3;
- PutBucketRequestPayment:S3;
- PutBucketTagging:S3;
- PutBucketVersioning:S3;
- DeleteBucketCors:S3;
- DeleteBucketPolicy:S3;
- DeleteBucketWebsite:S3;
- PutBucketAcl:S3;
- PutBucketCors:S3;
- PutBucketPolicy:S3;
- PutBucketWebsite:S3.