SECURITY TESTING OF INFRASTRUCTURE AS CODE

PRESENTED BY

Conor McKenna

SUPERVISOR

Hisain Elshaafi

FINAL REPORT

TABLE OF CONTENTS

Introduction

Infrastructure as Code (IaC) is a popular method and powerful technique for automating the deployment and configuration of cloud infrastructures. By treating infrastructure as code, developers can version, test and reliably deploy cloud resources, leading to faster and more consistent deployments. With the increasing use of IaC, it has become essential to ensure that security is taken into consideration during the development and deployment process. In this project, we will be developing a static security testing tool that will help identify, insecure patterns, vulnerabilities, and misconfigurations in IaC.

The aim of this project was to develop and implement an automated tool (**Mckscat**) which will allow customisation of the security tests using rules and laws that will facilitate fixing security issues by cloud developers before rolling out new or alter infrastructure.

To develop the security testing tool, we will be using technologies such as AWS CloudFormation alongside AWS services. CloudFormation templates can be written in JSON format and will analyse JSON scripts.

Techniques

There are various techniques to analyze static source code for potential vulnerabilities that maybe combined into one solution. These techniques are often derived from compiler technologies.

Limitations

False Positives

A static code analysis tool will often produce false positive results where the tool reports a possible vulnerability that in fact is not. This often occurs because the tool cannot be sure of the integrity and security of data as it flows through the application from input to output.

False Negatives

The use of static code analysis tools can also result in false negative results where vulnerabilities result but the tool does not report them. This might occur if a new vulnerability is discovered in an external component or if the analysis tool has no knowledge of the runtime environment and whether it is configured securely.

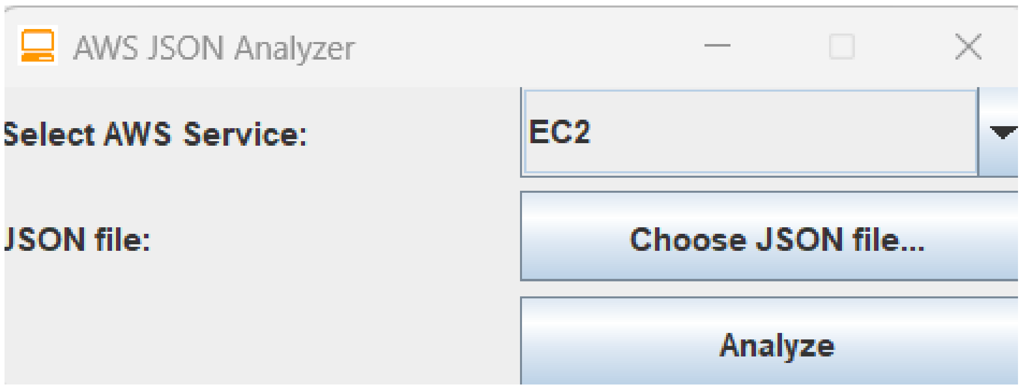
Description of Project

The main objective of this tool is to analyse JSON formatted AWS CloudFormation scripts, (Simple Storage Services (S3), Elastic Cloud (EC2) or a virtual private cloud (VPC)) that will identify potential security vulnerabilities related to the AWS services listed. This will allow a user select which file type they want to be analysed that best fits their scripts, recognising that there are different rules to be considered over different types of scripts.

1. Download the tool and pin to desktop.

Figure 1

2. Launch the application.



Select AWS Service: EC2

JSON file:

Figure 2

3. Select the AWS service you want analysed.

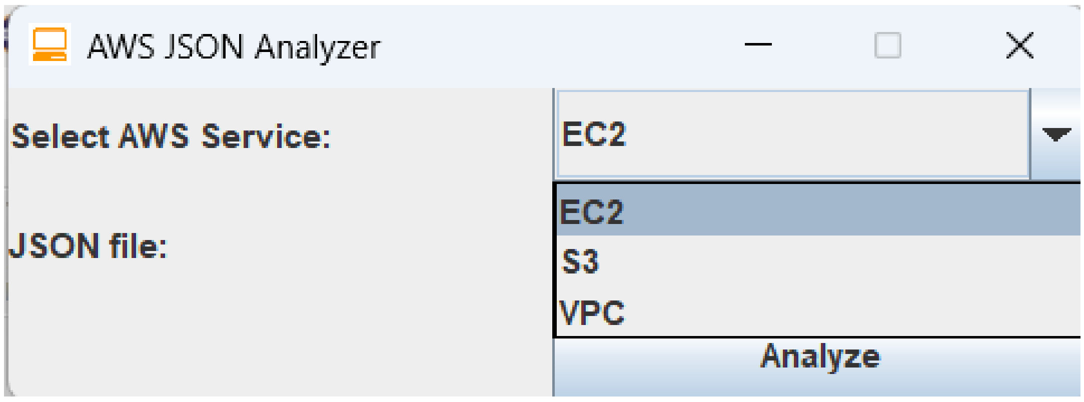


Figure 3

4. Select the file to be analysed.

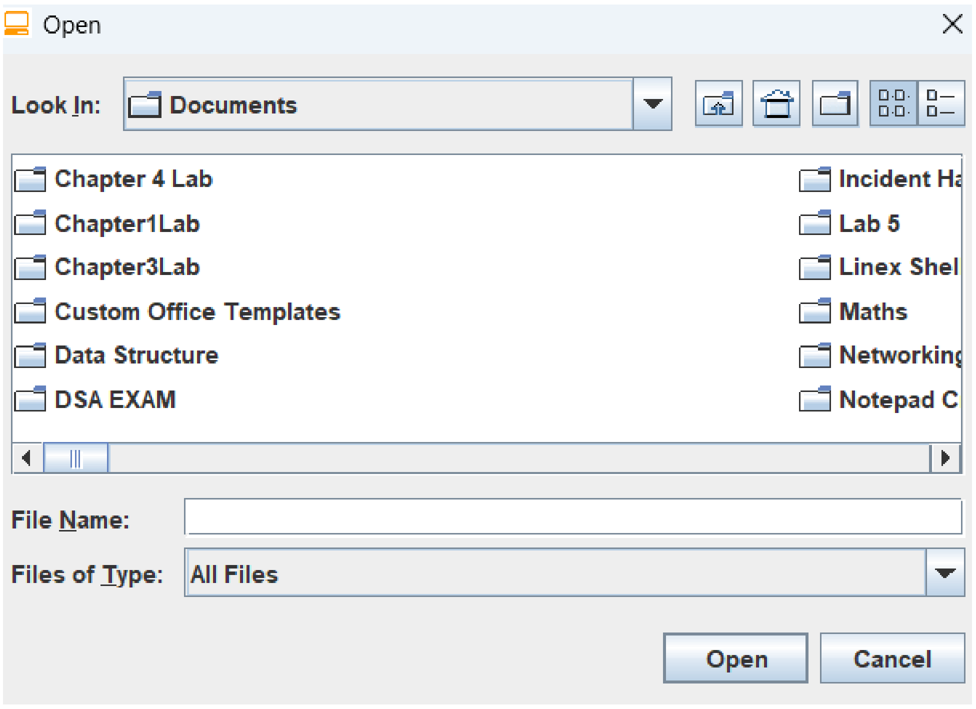
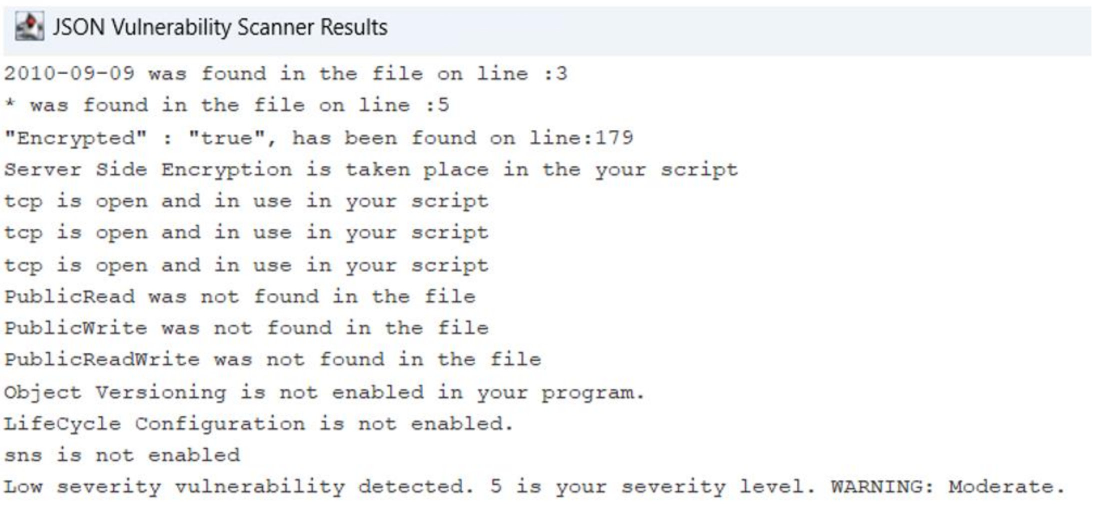


Figure 4

5. Report returns vulnerabilities present in the script.



JSON Vulnerability Scanner Results

```
2010-09-09 was found in the file on line :3
* was found in the file on line :5
"Encrypted" : "true", has been found on line:179
Server Side Encryption is taken place in the your script
tcp is open and in use in your script
tcp is open and in use in your script
tcp is open and in use in your script
PublicRead was not found in the file
PublicWrite was not found in the file
PublicReadWrite was not found in the file
Object Versioning is not enabled in your program.
LifeCycle Configuration is not enabled.
sns is not enabled
Low severity vulnerability detected. 5 is your severity level. WARNING: Moderate.
```

Figure 5

Once the user selects the file from the file explorer interface the script type selected will run in the background and output the feedback and vulnerabilities to the end user. With three different scripts, each of which uses different rules based on common vulnerabilities found in the script type, it will apply the rules to the selected formats. Some rules are based upon variables, including:-

- Strings representing AWS keywords
- Symbols and service versions
- Boolean flags indicating whether the keywords or specific symbols were found within the file.

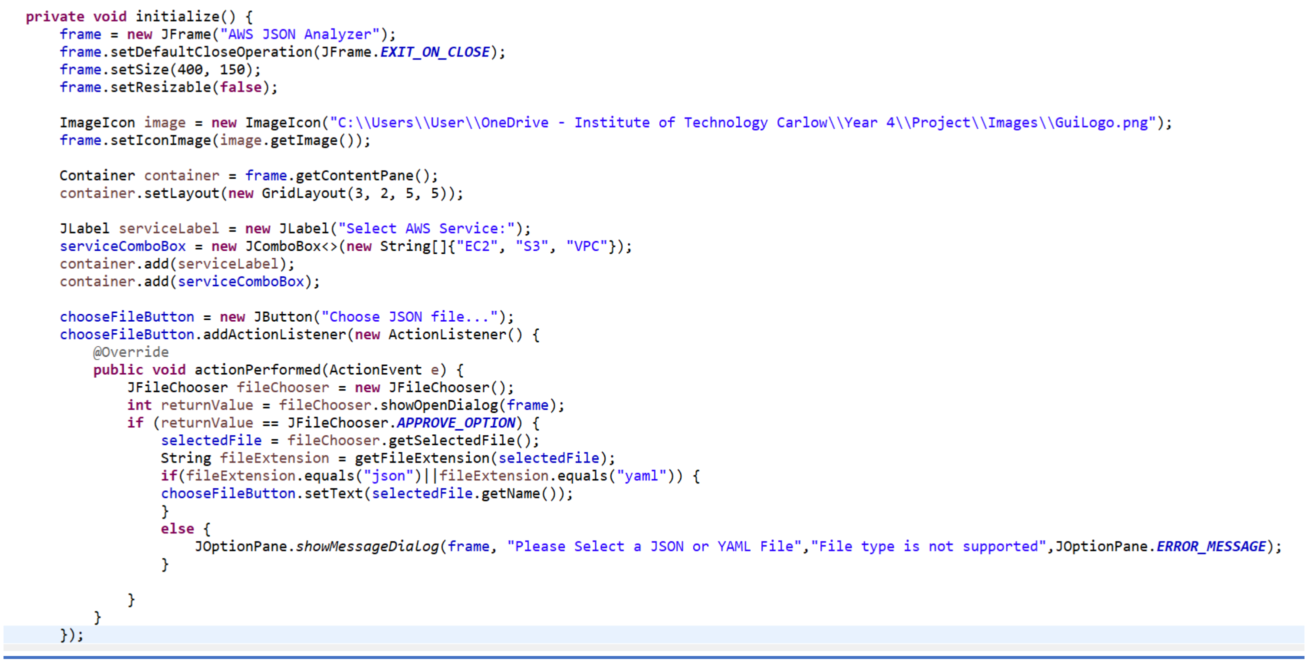
With the buffer reader implemented in the file, it will scan each line in the file and check the rules in the file. If a vulnerability is present in the script the method appends a message to the ResultFrame object indicating the severity level of the vulnerability and the line number where the security concern is found. Depending on the number of vulnerabilities found in the script it will give back a severity warning from no vulnerabilities found to a failed warning if the code has too many vulnerabilities in it.

Some of the Libraries/toolkits imported into this project are JSON simple. JSON simple is a Java library that provides a simple way to manipulate and read JSON formatted data. It provides a simple and efficient API that allows developers to work on JSON data while using Java-implemented functions like Maps, Lists or Arrays. The tool also helps the reading of JSON scripts easier which is the main function of this library in my project. Other libraries in use throughout this program are Java-based libraries e.g. java.io. * which is used in the handling of files and the input and output of files. Another library in use is java.util.StringTokenizer which is used in scanning lines in a script and checking the line for the appropriate rule.

Some of the benefits of using a static code analyser for AWS CloudFormation scripts we learned while developing the tool is that the detection of vulnerabilities and misconfigurations can help the end user resolves the issues. By using this application, you will improve the security of your infrastructure. Also, the use of our tool can save the company or end user time and money in fixing and securing their Infrastructure as Code.

Samples of Code

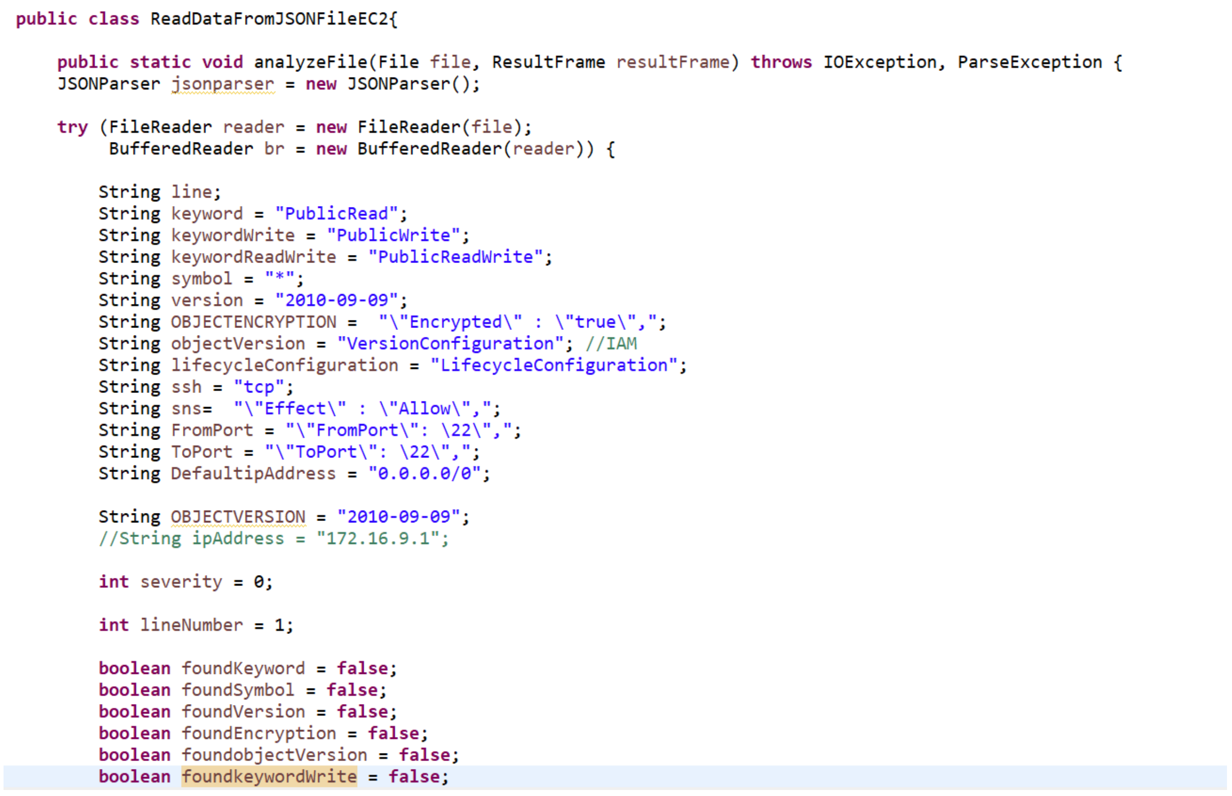
In the below snippets of code we can see the code implemented below is the prompt that greets the end-user once the application is launched. A prompt appears where the user can select the AWS Service they want the rules of either S3, EC2 or VPC. Once the file is selected the rules will be applied to the selected files. The below code also allows the users to select only a JSON file and any other file type will be dropped.



```
private void initialize() {
    frame = new JFrame("AWS JSON Analyzer");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400, 150);
    frame.setResizable(false);

    ImageIcon image = new ImageIcon("C:\\Users\\User\\OneDrive - Institute of Technology Carlow\\Year 4\\Project\\Images\\GuiLogo.png");
    frame.setIconImage(image.getImage());

    Container container = frame.getContentPane();
    container.setLayout(new GridLayout(3, 2, 5, 5));

    JLabel serviceLabel = new JLabel("Select AWS Service:");
    serviceComboBox = new JComboBox<>(new String[]{"EC2", "S3", "VPC"});
    container.add(serviceLabel);
    container.add(serviceComboBox);

    chooseFileButton = new JButton("Choose JSON file...");
    chooseFileButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            JFileChooser fileChooser = new JFileChooser();
            int returnValue = fileChooser.showOpenDialog(frame);
            if (returnValue == JFileChooser.APPROVE_OPTION) {
                selectedFile = fileChooser.getSelectedFile();
                String fileExtension = getFileExtension(selectedFile);
                if(fileExtension.equals("json")||fileExtension.equals("yaml")) {
                chooseFileButton.setText(selectedFile.getName());
                }
                else {
                    JOptionPane.showMessageDialog(frame, "Please Select a JSON or YAML File","File type is not supported",JOptionPane.ERROR_MESSAGE);
                }

            }
        }
    });
```

Figure 6

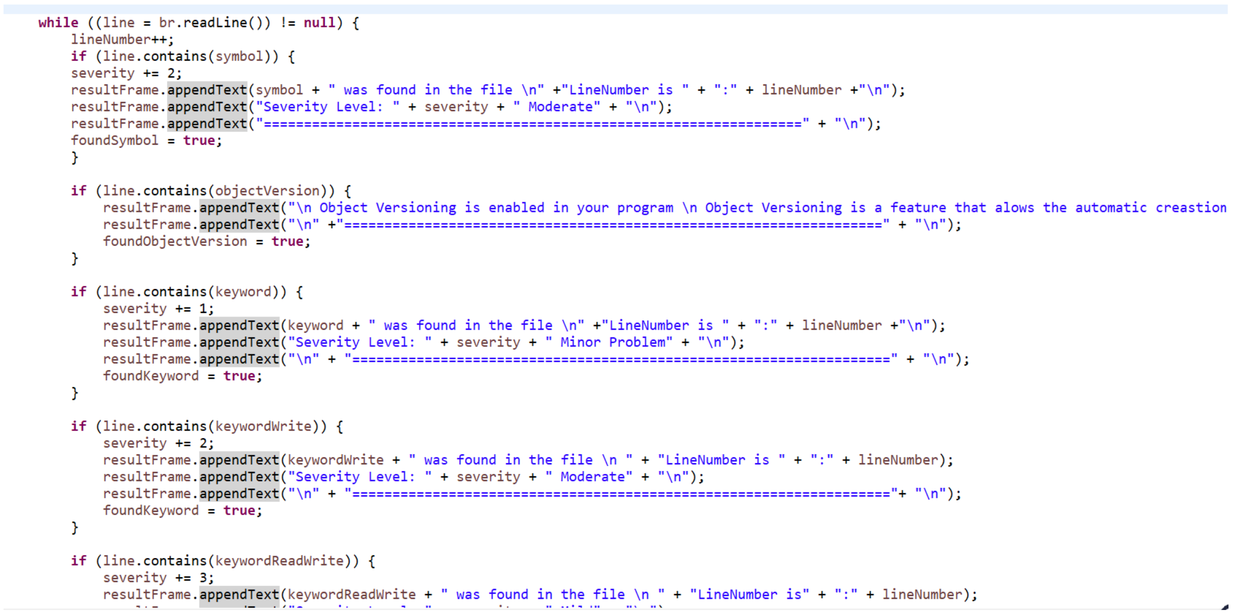
In the code below we see a snippet of code from one of the selected AWS services files which store some rules of the service. With the code allowing JSON files to be read in correctly it can also be seen that keywords, IP addresses & port numbers are some of the vulnerabilities we are examining in the end-users AWS script.



```java
public class ReadDataFromJSONFileEC2{

    public static void analyzeFile(File file, ResultFrame resultFrame) throws IOException, ParseException {
    JSONParser jsonparser = new JSONParser();

    try (FileReader reader = new FileReader(file);
         BufferedReader br = new BufferedReader(reader)) {

        String line;
        String keyword = "PublicRead";
        String keywordWrite = "PublicWrite";
        String keywordReadWrite = "PublicReadWrite";
        String symbol = "*";
        String version = "2010-09-09";
        String OBJECTENCRYPTION =  "\"Encrypted\" : \"true\",";
        String objectVersion = "VersionConfiguration"; //IAM
        String lifecycleConfiguration = "LifecycleConfiguration";
        String ssh = "tcp";
        String sns=  "\"Effect\" : \"Allow\",";
        String FromPort = "\"FromPort\": \22\",";
        String ToPort = "\"ToPort\": \22\",";
        String DefaultipAddress = "0.0.0.0/0";

        String OBJECTVERSION = "2010-09-09";
        //String ipAddress = "172.16.9.1";

        int severity = 0;

        int lineNumber = 1;

        boolean foundKeyword = false;
        boolean foundSymbol = false;
        boolean foundVersion = false;
        boolean foundEncryption = false;
        boolean foundobjectVersion = false;
        boolean foundkeywordWrite = false;
```

Figure 7

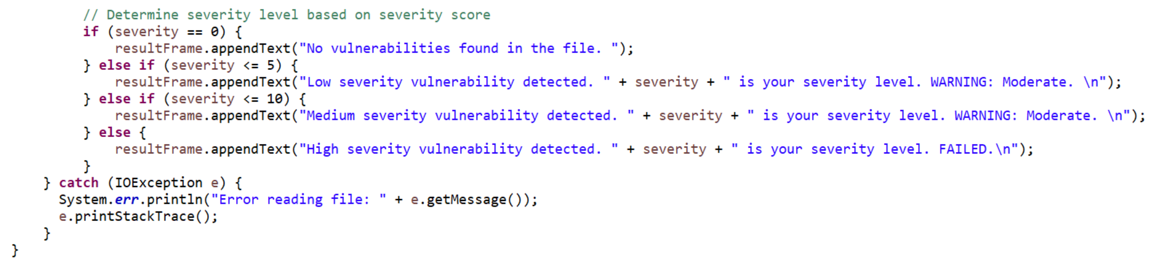
In the following clip we get a sample of what will be returned to the end user when the script is ran locally on their machine. We can see that the code is examining if a wildcard mask is present in the end users scripts & if object versioning is on. Some of the code will return helpful descriptions of the benefits of what some functions do in this case Object Versioning



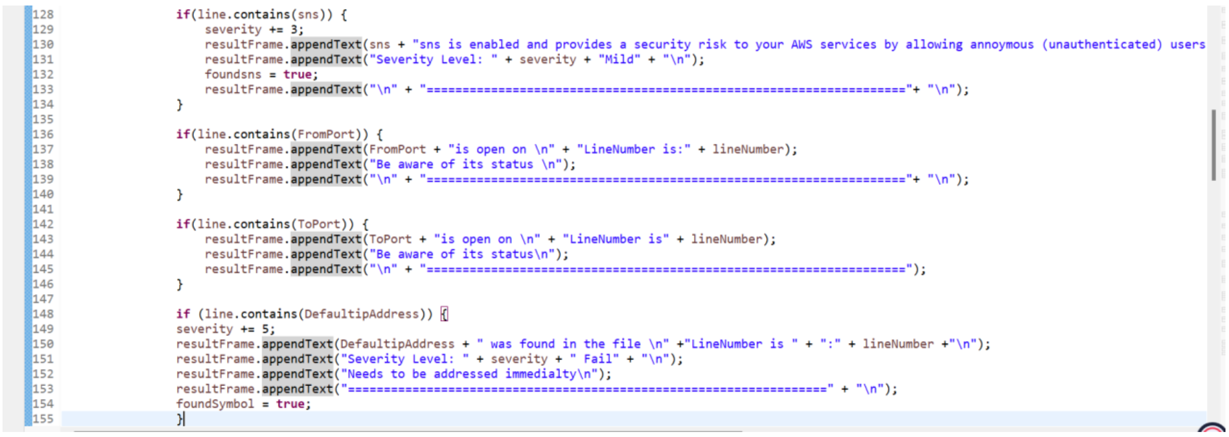
```java
    while ((line = br.readLine()) != null) {
        lineNumber++;
        if (line.contains(symbol)) {
        severity += 2;
        resultFrame.appendText(symbol + " was found in the file \n" +"LineNumber is " + ":" + lineNumber +"\n");
        resultFrame.appendText("Severity Level: " + severity + " Moderate" + "\n");
        resultFrame.appendText("==================================================================" + "\n");
        foundSymbol = true;
        }

        if (line.contains(objectVersion)) {
            resultFrame.appendText("\n Object Versioning is enabled in your program \n Object Versioning is a feature that alows the automatic creastion
            resultFrame.appendText("\n" +"==================================================================" + "\n");
            foundObjectVersion = true;
        }

        if (line.contains(keyword)) {
            severity += 1;
            resultFrame.appendText(keyword + " was found in the file \n" +"LineNumber is " + ":" + lineNumber +"\n");
            resultFrame.appendText("Severity Level: " + severity + " Minor Problem" + "\n");
            resultFrame.appendText("\n" + "==================================================================" + "\n");
            foundKeyword = true;
        }

        if (line.contains(keywordWrite)) {
            severity += 2;
            resultFrame.appendText(keywordWrite + " was found in the file \n " + "LineNumber is " + ":" + lineNumber);
            resultFrame.appendText("Severity Level: " + severity + " Moderate" + "\n");
            resultFrame.appendText("\n" + "=================================================================="+ "\n");
            foundKeyword = true;
        }

        if (line.contains(keywordReadWrite)) {
            severity += 3;
            resultFrame.appendText(keywordReadWrite + " was found in the file \n " + "LineNumber is" + ":" + lineNumber);
```

Figure 8

Additionally, in the next screenshot we are not looking for keywords but rather port numbers or port statuses as well as IP addresses. Instead of keywords we are looking for numbers and examining it the IP address is a default IP address or if it is a security risk.



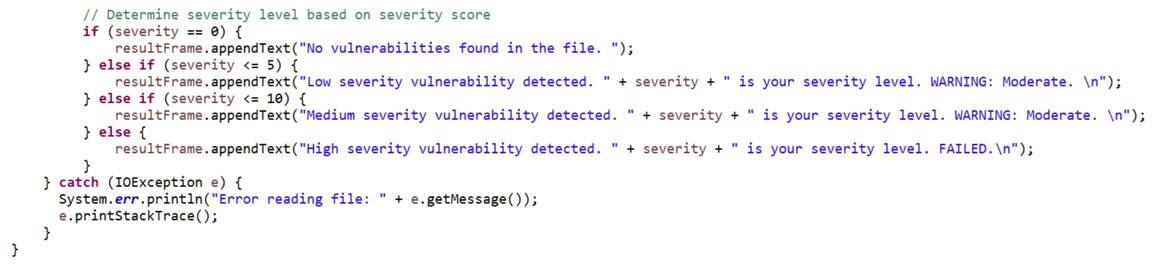
```java
            // Determine severity level based on severity score
            if (severity == 0) {
                resultFrame.appendText("No vulnerabilities found in the file. ");
            } else if (severity <= 5) {
                resultFrame.appendText("Low severity vulnerability detected. " + severity + " is your severity level. WARNING: Moderate. \n");
            } else if (severity <= 10) {
                resultFrame.appendText("Medium severity vulnerability detected. " + severity + " is your severity level. WARNING: Moderate. \n");
            } else {
                resultFrame.appendText("High severity vulnerability detected. " + severity + " is your severity level. FAILED.\n");
            }
        } catch (IOException e) {
          System.err.println("Error reading file: " + e.getMessage());
          e.printStackTrace();
        }
    }
```



```java
                    if(line.contains(sns)) {
                        severity += 3;
                        resultFrame.appendText(sns + "sns is enabled and provides a security risk to your AWS services by allowing annoymous (unauthenticated) users
                        resultFrame.appendText("Severity Level: " + severity + "Mild" + "\n");
                        foundsns = true;
                        resultFrame.appendText("\n" + "================================================================"+ "\n");
                    }

                    if(line.contains(FromPort)) {
                        resultFrame.appendText(FromPort + "is open on \n" + "LineNumber is:" + lineNumber);
                        resultFrame.appendText("Be aware of its status \n");
                        resultFrame.appendText("\n" + "================================================================"+ "\n");
                    }

                    if(line.contains(ToPort)) {
                        resultFrame.appendText(ToPort + "is open on \n" + "LineNumber is" + lineNumber);
                        resultFrame.appendText("Be aware of its status\n");
                        resultFrame.appendText("\n" + "================================================================");
                    }

                    if (line.contains(DefaultipAddress)) {
                    severity += 5;
                    resultFrame.appendText(DefaultipAddress + " was found in the file \n" +"LineNumber is " + ":" + lineNumber +"\n");
                    resultFrame.appendText("Severity Level: " + severity + " Fail" + "\n");
                    resultFrame.appendText("Needs to be addressed immedialty\n");
                    resultFrame.appendText("================================================================" + "\n");
                    foundSymbol = true;
                    }
```

Figure 9

The final image is a snippet of how the severity scanning is being calculated. With the scores being given out based on the severity of the error at the end of the program all severity present will be added together and given back to the end user and a message as well. If the script contains a security score above ten then it is at a fail state. It is recommended for the end user to go back and secure their script.



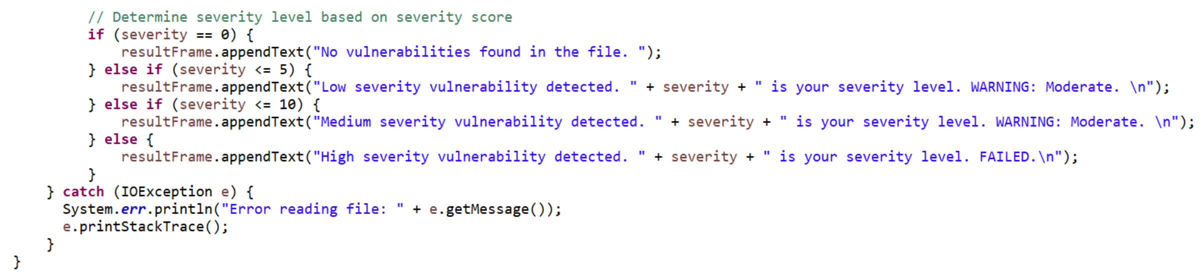
```java
            // Determine severity level based on severity score
            if (severity == 0) {
                resultFrame.appendText("No vulnerabilities found in the file. ");
            } else if (severity <= 5) {
                resultFrame.appendText("Low severity vulnerability detected. " + severity + " is your severity level. WARNING: Moderate. \n");
            } else if (severity <= 10) {
                resultFrame.appendText("Medium severity vulnerability detected. " + severity + " is your severity level. WARNING: Moderate. \n");
            } else {
                resultFrame.appendText("High severity vulnerability detected. " + severity + " is your severity level. FAILED.\n");
            }
        } catch (IOException e) {
          System.err.println("Error reading file: " + e.getMessage());
          e.printStackTrace();
        }
    }
```

Figure 10

Description of Learning

This project has proved challenging and somewhat of a learning curve to get it over the finish line. With no previous knowledge of Amazon Web Services (AWS), this was the starting point and an in-depth study of this organisation's cloud computing services was undertaken.

Although this project focused on Infrastructure as Code, AWS has a wide range of services which need to be understood, to set the context. The focus of this project is to focus on the analysis of IAC scripts and to mitigate against potential security vulnerabilities. We had to learn what the IaC scripts look like, what they contained and more importantly what made them secure while also learning what can cause them to be vulnerable. All scripts had to be examined and the rules to be applied to each specific AWS script type considered. I also had to learn about the different services it offers to individuals or companies that are using AWS as a service. Some of the different services they focus on are storage, networking or machine learning etc.



```
            // Determine severity level based on severity score
            if (severity == 0) {
                resultFrame.appendText("No vulnerabilities found in the file. ");
            } else if (severity <= 5) {
                resultFrame.appendText("Low severity vulnerability detected. " + severity + " is your severity level. WARNING: Moderate. \n");
            } else if (severity <= 10) {
                resultFrame.appendText("Medium severity vulnerability detected. " + severity + " is your severity level. WARNING: Moderate. \n");
            } else {
                resultFrame.appendText("High severity vulnerability detected. " + severity + " is your severity level. FAILED.\n");
            }
        } catch (IOException e) {
          System.err.println("Error reading file: " + e.getMessage());
          e.printStackTrace();
        }
    }
```

Figure 11

AWS and IaC not only streamline the infrastructure management process but also foster a culture of collaboration and efficiency in the development and operations team. This collaboration is often referred to as DevOps, which emphasizes the integration of development and operations processes to improve script quality, speed and more importantly reliability.

In conclusion, we learned that IaC provides a powerful combination for organisations and what it offers to companies cannot be underestimated. By learning what IaC is we can also solve some of the common vulnerabilities that may be present in an IaC script.

Conformance to Specification and Design

After reviewing the project specification, the goals and outline of the project brief we can confirm that most of the project brief has been achieved. We have developed a static code testing tool that scans AWS CloudFormation. We have the tool analysing three different types of AWS CloudFormation scripts (S3, EC2, VPC). We have the tool scanning potential vulnerabilities, insecure patterns and present misconfigurations. The tool also gives feedback on how to improve your script before publishing the user's script and I have implemented a set of rules for different script types.

If additional time allowed I would further implement integration of the tool into a continuous delivery (CD) pipeline. I would also like to implement more rules for each of the file types to look for more vulnerabilities. If I had more time, I would update the design of the application so that it would look more visually appealing and make it easier for the end user to use.

General Issues

The assignment of a "Programing" heavy project was initially a shock as this area of study is not my strength. Hence, the time invested was labourious, slow and stressful on many levels. The first problem I faced was that I believed that the project had to be completed in Python. For the first few weeks the focus was on learning python and trying to understand the best way to implement the project. As I did not have any previous experience of programming in Python it was extremely difficult to grasp an understanding in a short timeline and in particular to design and implement rules into the project. Once it became clear that Python was not essential, I decided to switch the programming language from Python to Java.

Another issue I had while completing this project was that I had no prior knowledge of Amazon Web Services. With the final year project based upon elements we learned from first year through to subjects done in fourth year, it seemed quite unrealistic that the concept of Cloud Computing was never covered. A considerable amount of time was taken on researching AWS and the services/vulnerabilities that are open to occurrences and in gaining an understanding on how best to approach this project and what were the key elements that should be focused on.

Ensuring a thorough understanding of the project specification was also a major cause for concern. The brief project description gave no clear steps/tips on how to approach and ultimately accomplish this project. The vague and unclear brief resulted in me taking the wrong path on a number of occasions and consequently having to go back to the drawing board and re-beginning again, a frustrating and stressful experience.

The later release of projects, compared to previous year groups, and an earlier submission date added to the pressures experienced on this project. Additional time would have been welcome to ensure a detailed, well-developed and resourceful tool.

Review of Project

Following completion of this project and having time to reflect on the experience, I can recognise the various pros and cons. Some of the pros have been getting the application to a stage where it runs to the same level described in the project brief. Some of the cons I found in this project is that I had no knowledge of AWS and the services I needed to implement into my project for it to run correctly. As mentioned earlier in the report one of the aspects I am missing in my project is the implementation of this project into a CI/CD pipeline. If I had to start the project again, I would focus more on the research aspect of this project brief as I did not enjoy the coding aspect of the project and struggled with it for long parts of the project. I would focus on the vulnerabilities the complexity of these vulnerabilities and the way in which these can be prevented and/or solved rather than create a program which scans a script for specific vulnerabilities. I believe that the technologies, libraries, and other Java programs implemented into the application are the best technologies to have implemented. They enhance the usability of the application and help get the program to the next level, so I believe all technologies used are necessary for this program.

Overall, I think I can confidently say this project has been a success. The application has been created and provides a user with key information and security concerns that are present in the users' Infrastructure as Code scripts. The program also allows the users to select the type of file they want to be analysed. It also lets the end user find the file they want to select with ease. It also provides end users with the precious location of the vulnerability and tells the user a solution to ensure that their IaC script is more secure.

Conclusion

Modern software development depends on achieving the “magic three” outcomes: faster releases, shorter cycles, and higher quality code. In order for code to be of high quality, it needs to be secure, so cybersecurity must be integrated with the development cycle in a way that does not hinder developers’ ability to release code quickly.

Static code analysis is a software verification technique that refers to the process of examining code without executing it in order to capture the defects in the code early, avoiding later costly fixations. While SAST offers many benefits, the most significant is its ability to detect security vulnerabilities and mark their precise location, including the file name and line number. This ability to pinpoint problems is crucial, as finding problems is one of the most time-consuming aspects of a developer’s work.

Infrastructure as Code deployment and automated security management have been proven to reduce security misconfigurations and eliminate security gaps. Security architects need to adopt building infrastructure access based on new forms of Authentication, Authorisation and Access (AAA) such as block-chain, compound identities and service principals.

APPENDICES

Acknowledgements

I would like to thank Hisain Elshaafi, my project supervisor, Richard Butler and all academic staff on Cybercrime and IT Security for their assistance. I would like to acknowledge the time they have dedicated to review and evaluate my work over the course of the last four years and for their encouragement, support and advice during the course of my studies.

Glossary

AAA – Authentication, Authorisation & Access

AWS – Amazon Web Services

ACL – Access Control List

CDK – Cloud Development Kit

CI/CD – Continuous Integration & Continuous Delivery

CSP – Cloud Service Providers

DevOps – Development Operations

DCA – Dynamic Code Analysis

EC2 – Elastic Compute 2

ECR – Elastic Container Registry

FURPS – Functionality/Usability/Reliability/Performance/Supportability

IaC – Infrastructure as Code

IAAS - Infrastructure as a Service

IAM – Identify Access Management

IDE – Integrated Development Environment

IoT – Information of Things

NIST – National Institute of Standards & Technology

PC – Public Cloud

S3 – Simple Storage Service

SAST – Static Application Security Testing

SCA – Static Code Analysis

SDI – Software Defined Infrastructure

UI – User Interface

VPC – Virtual Private Cloud

WFA – Well Architectured Framework

Bibliography

Aurum, A., Petersson, H. and Wohlin, C. (2022) *State-of-the-Art: Software inspections after 25 years*. Available at: https://onlinelibrary.wiley.com/doi/10.1002/stvr.243 (Accessed: November 15, 2022).

Campbell, B. (no date) *The Definitive Guide to AWS Infrastructure Automation*, *SpringerLink*. Apress. Available at: https://link.springer.com/book/10.1007/978-1-4842-5398-4 (Accessed: November 14, 2022).

Staff, T.T. (2022) *Infrastructure as code: 6 best practices to get the most out of IAC*, *Thorn Technologies*. Available at: https://thorntech.com/infrastructure-as-code-best-practices/ (Accessed: November 14, 2022).

Janani (2021) *Infrastructure as code (IAC): Definition, principles & more*, *Atatus*. DevOps and Software Engineering Glossary Terms | Atatus. Available at: https://www.atatus.com/glossary/infrastructure-as-code/ (Accessed: October 15, 2022).

Continella, A. *et al.* (2018) *There's a hole in that bucket!: A large-scale analysis ... - researchgate*, *There's a Hole in that Bucket! A Large-scale Analysis of Misconfigured S3 Buckets*. Available at: https://www.researchgate.net/publication/340602033_There's_a_Hole_in_that_Bucket_A_Large-scale_Analysis_of_Misconfigured_S3_Buckets (Accessed: 2022).

Loukis , E., Janssen, M. and Mintchev, I. (2018) *Determinants of software-as-a-service benefits and impact on firm performance*, *Decision Support Systems*. North-Holland. Available at: https://www.sciencedirect.com/science/article/pii/S016792361830201X (Accessed: October 25, 2022).

Guerrirero, M. *et al.* (no date) *Adoption, support, and challenges of infrastructure-as ... - IEEE xplore*, *Adoption, Support & Challenges of Infrastructure-as-Code*. Available at: https://ieeexplore.ieee.org/abstract/document/8919181 (Accessed: October 27, 2022).

Jimenez, W., Mammer, A. and Cavalli, A.R. (2010) *Software vulnerabilities, prevention and detection methods: A review 1*, *Software Vulnerabilities, Prevention & Detection Methods: A Review 1*. Available at: https://www.researchgate.net/publication/253704494_Software_Vulnerabilities_Prevention_and_Detection_Methods_A_Review_1 (Accessed: October 30, 2022).

Kumara , I. *et al.* (2021) *The do's and don'ts of infrastructure code: A systematic gray literature review*, *Information and Software Technology*. Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S0950584921000720 (Accessed: 2022).

Morris, K. (no date) *Infrastructure as code*, *O'Reilly Online Learning*. O'Reilly Media, Inc. Available at: https://www.oreilly.com/library/view/infrastructure-as-code/9781491924334/ (Accessed: 2022).

Test-driven infrastructure with chef, 2nd edition (no date) *O'Reilly Online Learning*. O'Reilly Media, Inc. Available at: https://www.oreilly.com/library/view/test-driven-infrastructure-with/9781449372576/ (Accessed: 2022).

Palma , S.D. *et al.* (2020) *catalog of software quality metrics for infrastructure code*, *Journal of Systems and Software*. Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S0164121220301618 (Accessed: 2022).

Rahman, A., Hezaveh, R.M. and Williams, L. (2018) *A systematic mapping study of infrastructure as code research*, *Information and Software Technology*. Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S0950584918302507 (Accessed: 2022).

Rahman, A., Farhana, E. and Williams, L. (2020) *The 'as code' activities: Development anti-patterns for infrastructure as code - empirical software engineering*, *SpringerLink*. Springer US. Available at: https://link.springer.com/article/10.1007/s10664-020-09841-8 (Accessed: 2022).

M. Shahin, M. A. Babar, M. Zahedi and L. Zhu, "Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges," *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Toronto, ON, Canada, 2017, pp. 111-120, doi: 10.1109/ESEM.2017.18.

Ulsch, M.D. (no date) *Cyber threat!: How to manage the growing risk of cyber attacks*, *O'Reilly Online Learning*. Wiley. Available at: https://www.oreilly.com/library/view/cyber-threat-how/9781118935958/#:~:text=Book%20description&text=Cyber%20Threat!,align%20to%20maintain%20information%20integrity. (Accessed: 2022).

Tavbulatova, Z.K. *et al.* (2020) *IOPscience*, *Journal of Physics: Conference Series*. IOP Publishing. Available at: https://iopscience.iop.org/article/10.1088/1742-6596/1582/1/012085 (Accessed: 2022).

Li , Y. and Liu, Q. (2021) *A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments*, *Energy Reports*. Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S2352484721007289 (Accessed: 2022).

Mongescu, I. (no date) *Fugue State Of Cloud Security 2021*, *Scribd*. Scribd. Available at: https://www.scribd.com/document/633558018/Fugue-State-of-Cloud-Security-2021# (Accessed: 2022).

National Cyber Security Strategy 2019-2024 (no date) *ReadkonG.com*. Available at: https://www.readkong.com/page/national-cyber-security-strategy-2019-2024-7843807 (Accessed: 2022).

What is devops and how does it work? (no date) *Synopsys*. Available at: https://www.synopsys.com/glossary/what-is-devops.html (Accessed: 2022).

Deo, N. (2021) *3 advantages and challenges of infrastructure as code (IAC)*, *CloudBolt Software*. Available at: https://www.cloudbolt.io/blog/3-advantages-and-challenges-of-infrastructure-as-code-iac/ (Accessed: October 15, 2022).

Unknown (2022) *What is infrastructure as code (IAC)? Red Hat - We make open-source technologies for the enterprise*. Available at: https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac#:~:text=Infrastructure%20as%20Code%20(IaC)%20is,to%20edit%20and%20distribute%20configurations (Accessed: October 18, 2022).

Unknown (2019) *What is cloud infrastructure? Red Hat - We make open-source technologies for the enterprise*. Red Hat. Available at: https://www.redhat.com/en/topics/cloud-computing/what-is-cloud-infrastructure#:~:text=Cloud%20infrastructure%20is%20a%20term,needed%20to%20build%20a%20cloud. (Accessed: October 21, 2022).

Unknown (no date) *What is Amazon S3?* AWS. Available at: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html (Accessed: November 24, 2022).

Fiser, D. (2020) *Infrastructure as code: Security risks and how to avoid them*, *Security News*. Trend Micro. Available at: https://www.trendmicro.com/vinfo/us/security/news/virtualization-and-cloud/infrastructure-as-code-security-risks-and-how-to-avoid-them#:~:text=Vulnerabilities%20could%20allow%20attackers%20to,or%20leave%20openings%20for%20attacks (Accessed: November 6, 2022).

Braak, C. (no date) *AWS CloudFormation FAQs*, *Amazon*. Amazon. Available at: https://aws.amazon.com/cloudformation/faqs/#:~:text=AWS%20CloudFormation%20is%20a%20service,an%20orderly%20and%20predictable%20fashion (Accessed: November 6, 2022).

Kokje, A. (2018) *AWS CloudFormation Basics and tutorial*, *Medium*. Medium. Available at: https://amolkokje.medium.com/aws-cloudformation-basics-and-tutorial-6a60d4de958c (Accessed: November 20, 2022).

Unknown (2022) *What is AWS CloudFormation? Contino*, *Contino*. Contino. Available at: https://www.contino.io/insights/aws-cloudformation (Accessed: November 4, 2022).

Jawale, C. (2020) *Securing Infrastructure as code*, *Opcito*. Opcito. Available at: https://www.opcito.com/blogs/securing-infrastructure-as-code (Accessed: November 21, 2022).

Bellairs, R. (2020) *What is static code analysis? Static Analysis Overview*, *Perforce Software*. Perforce. Available at: https://www.perforce.com/blog/sca/what-static-analysis (Accessed: November 21, 2022).

AWS CloudFormation (no date) *Amazon*. Amazon. Available at: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/best-practices.html (Accessed: November 23, 2022).

Edviser (2020) *Admin*, *Skill Monk*. Available at: https://blog.skillmonks.com/9-top-aws-jobs-you-can-get-with-aws-certification/ (Accessed: December 14, 2022).

Engdahl, S. (2022) *Blogs*, *Amazon*. Greenhaven Press/Gale. Available at: https://aws.amazon.com/blogs/architecture/using-devops-automation-to-deploy-lambda-apis-across-accounts-and-environments/ (Accessed: December 16, 2022).

Engineering, S. (2020) *Cloud governance with CFRipper*, *Medium*. Medium. Available at: https://medium.com/@SkyscannerEng/cloud-governance-with-cfripper-8890d7413c98 (Accessed: December 16, 2022).

SonarQube (2022) *Wikipedia*. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/SonarQube (Accessed: December 21, 2022).

Chen, L. (2017) *Continuous delivery: Overcoming adoption challenges*, *Journal of Systems and Software*. Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S0164121217300353 (Accessed: November 15, 2022).

Gnanasambandam, C., Libarikian, A. and Turkeli, C. (2022) *The saas factor: Six ways to drive growth by building new SAAS businesses*, *McKinsey & Company*. McKinsey & Company. Available at: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-saas-factor-six-ways-to-drive-growth-by-building-new-saas-businesses#/ (Accessed: 2022).

Figure References

Figure 1 - BasuMallick, C.T. (2022) *What is infrastructure as code? meaning, working, and benefits*, *Spiceworks*. Available at: https://www.spiceworks.com/tech/cloud/articles/what-is-infrastructure-as-code/ (Accessed: 2022).

Figure 2 - Patil Sr. Assistant Editor, P. and BasuMallick Technical Writer opens a new window opens a new , C. (2022) *What is cloud computing? definition, benefits, types, and Trends*, *Spiceworks*. Available at: https://www.spiceworks.com/tech/cloud/articles/what-is-cloud-computing/ (Accessed: 2022).

Figure 3 - *Pros and cons of different cloud computing models* (2021) *Heptabit by Sedmi odjel*. Available at: https://www.heptabit.at/blog/cloud-migration/pros-and-cons-of-different-cloud-computing-models (Accessed: 2022).

Figure 4 - Vailshery, L.S. (2022) *Software as a service (SAAS) market worldwide 2008-2020*, *Statista*. Available at: https://www.statista.com/statistics/510333/worldwide-public-cloud-software-as-a-service/ (Accessed: 2022).

Figure 5 - Almashaqbeh, A. (2021) *Cloud deployment models*, *Cloud Computing Gate*. Available at: https://cloudcomputinggate.com/cloud-deployment-models/ (Accessed: 2022).

Figure 7 - Kumar |, R. (2022) *AWS market share 2023: How far it rules the cloud industry?*, *WPOven Blog*. Available at: https://www.wpoven.com/blog/aws-market-share/ (Accessed: 2022).

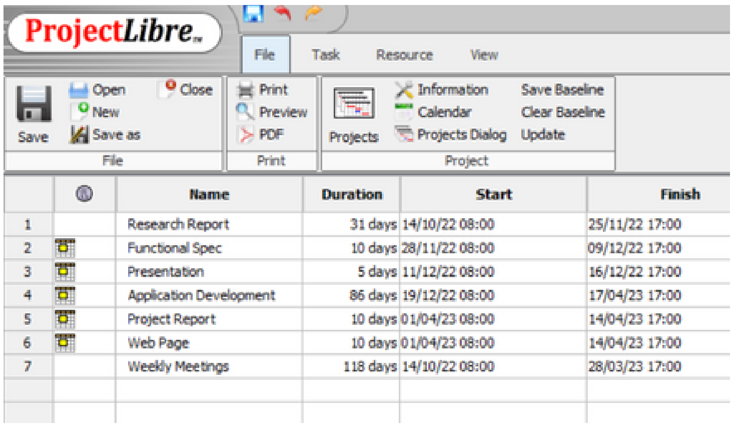
Figure 8 - *Top 10 AWS Services* (no date) *OpsMatters*. Available at: https://opsmatters.com/latest-posts-tag/cloud?page=174 (Accessed: 2022).

Figure 9 - Braak, C. (no date) *On cloudformation: Nuclei of condensation, haziness, dimensions of cloudparticles*, *Amazon*. Amazon. Available at: https://aws.amazon.com/cloudformation/ (Accessed: 2022).

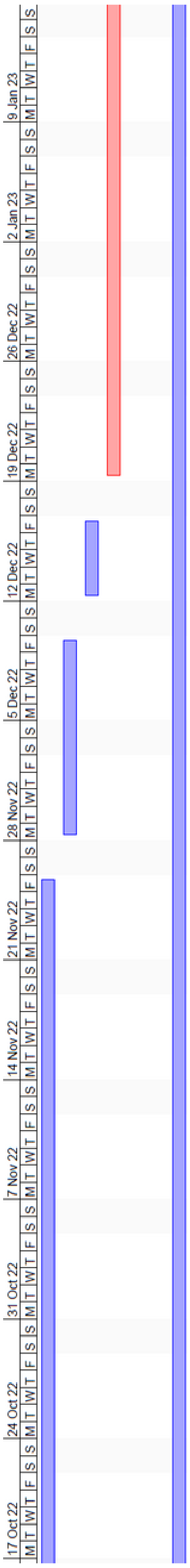
Figure 11 - Continuous integration vs continuous delivery vs continuous deployment #CICD (2017) *Dev Tips Curator*. Available at: https://devtipscurator.wordpress.com/2017/02/03/continuous-integration-vs-continuous-delivery-vs-continuous-deployment/ (Accessed: 2022).

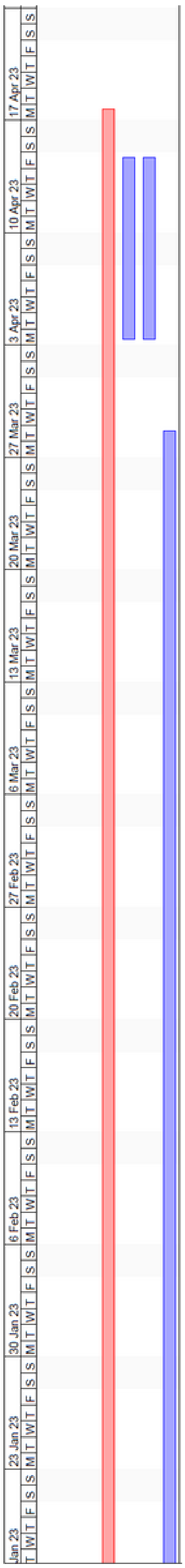
Figure 13 - Staff, V.B. (2021) *Fugue: 36% of organizations have suffered a serious cloud leak or breach in the last year*, *VentureBeat*. VentureBeat. Available at: https://venturebeat.com/data-infrastructure/fugue-36-percent-orgs-suffered-serious-cloud-breach-in-last-year/ (Accessed: 2022).

Project Timelines



		Name	Duration	Start	Finish
1		Research Report	31 days	14/10/22 08:00	25/11/22 17:00
2		Functional Spec	10 days	28/11/22 08:00	09/12/22 17:00
3		Presentation	5 days	11/12/22 08:00	16/12/22 17:00
4		Application Development	86 days	19/12/22 08:00	17/04/23 17:00
5		Project Report	10 days	01/04/23 08:00	14/04/23 17:00
6		Web Page	10 days	01/04/23 08:00	14/04/23 17:00
7		Weekly Meetings	118 days	14/10/22 08:00	28/03/23 17:00


17 Oct 22
24 Oct 22
31 Oct 22
7 Nov 22
14 Nov 22
21 Nov 22
28 Nov 22
5 Dec 22
12 Dec 22
19 Dec 22
26 Dec 22
2 Jan 23
9 Jan 23


Jan 23
23 Jan 23
30 Jan 23
6 Feb 23
13 Feb 23
20 Feb 23
27 Feb 23
6 Mar 23
13 Mar 23
20 Mar 23
27 Mar 23
3 Apr 23
10 Apr 23
17 Apr 23