



SECURITY TESTING OF INFRASTRUCTURE AS CODE

PRESENTED BY
Conor McKenna

SUPERVISOR
Hisain Elshaafi



Declaration of Plagiarism

I declare, this document in this submission in its entirety is my own work except for where duty acknowledged. I have cited the sources of all the quotations, paragraphs, summaries of information, tables, diagrams, or other material. This includes software and other electronic media that is integral property rights may reside. I have provided the complete biography at the end of my document detailing all the works and resources used in the presentation of this submission. I am aware that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offense.

Student: Conor McKenna


Tutor: Hisain Elshaafi

Institution: South East Technological University

Title: Security Testing of Infrastructure as Code

Submission Date: 17th April 2023

Date: 17th April 2023

 Recoverable Signature

X Conor McKenna

Conor McKenna

Mr

2. FUNCTIONAL SPECIFICATION

Table of Contents

Introduction.....	1
Application Purpose.....	1
Target Groups.....	2
Architecture Diagram.....	3
Use Cases.....	4
Non-Functional FURPS.....	5
Functionality.....	5
Usability.....	5
Reliability.....	5
Performance.....	5
Supportability.....	6
Metrics Plans for the Project.....	6
Design & Description of Application.....	7
Similar Applications Open-Source or Commercial.....	7
CFRipper.....	7
SonarQube.....	8
Codacy.....	9
CloudSpolit.....	9

Introduction

The research report has examined Cloud Infrastructure and Cloud-based web services that are used by many modern-day corporations. The primary focus is on Amazon Web Services and the services that they provide to their clients. This chapter will report on web services that are utilised in creating an infrastructure in Amazon and detail how some services can be exploited. An explanation will detail the language used in this project and the reasons for conducting this project around this specific language. Priority is to scan both YAML and JSON files that can run these scripts and also run through an AWS script together with the headers and topics needed to create an AWS script. By understanding how a script or template is created the exploits and vulnerabilities can be comprehended and snags which have occurred in previous scripts can be readily identified and resolved. This process will provide a better understanding of how each script is formatted and how these scripts can be made more secure.

This project aims to provide AWS clients with a tool that can inspect and provide feedback on AWS CloudFormation scripts from a cyber security point of view. This tool could then be implemented onto the AWS services and used alongside all AWS CloudFormation scripts by placing the tool towards the start of the Continuous Delivery (CD) pipeline.

This project aims to create a static code analyser to enable the scanning of AWS Cloud Formation scripts before execution. If successful, this will improve scripts and minimise the potential exploits that can occur in YAML or JSON scripts. Java is the programming language of choice to code this tool and together with a graphic user interface will allow users to operate this tool with ease, to provide useful security feedback to the AWS developer and help them improve their scripts. The aim is that the tool will perform to specifications and be successful in minimizing false positives that may occur when the script is scanned. Throughout the development of the tool, it is hoped that new issues can be identified and more importantly solutions found to resolve these issues.

Application Purpose

The purpose of this tool is to allow AWS CloudFormation scripts to be scanned and reviewed before the script is launched. Whether it is a script to create a whole Virtual Infrastructure or just a script that will create S3 buckets for EC2 instances. It should scan the scripts (YAML or JSON), process the data, and then provide feedback to the AWS architecture or AWS developers on improving their script from a security point of view. Potential problems that the script will identify to the user include, are ports open, IPs are not secure or vpc set up incorrectly etc. If the tool is successful in running, the tool will be integrated into the AWS CD pipeline. Ultimately then all cloud/AWS architecture can use this tool and consider the feedback analysis returned. Once the tool is adopted and implemented, constant revision and tweaks will be made to ensure that any vulnerabilities arising can be resolved. Ideally, the main functions aimed at identifying and minimising when running this tool are false positive and false negative readings.

Target Groups

The target group of users of this static code analysis tool is predominately AWS Developers. Since the tool is focused primarily on AWS CloudFormation scripts it would be targeted towards Amazon software developers, allowing the testing of scripts before publishing pre-built scripts. However, there is also scope for the tool to be utilised by other software developers who use AWS as part of their own business or any cloud computing employee.

AWS architecture can also use this tool when developing their scripts, while another potential customer for this tool could be cyber security analysts who have an interest in cloud computing. This group can view the tool, categorize what is happening when the tool is running and more importantly give feedback and advise how the tool can be improved upon. While this tool is framed as a static code analyser for AWS CloudFormation scripts, other cloud developers may replicate the concept but instead create a tool that would specifically target e.g. windows PowerShell scripts etc.



Figure 1 - Beneficial users of this tool

Architecture Diagram

The diagram below visually presents the procedure of how scripts are created within a VPC Amazon environment. The first step in the diagram, the IAM user is creating a YAML script following which the script is pushed into the AWS code and released into the CD pipeline. In the AWS pipeline the code then gets saved following which it is deployed within the pipeline. When it gets to the AWS CodeBuild it builds the code and then sends the code to be saved and stored in an S3 bucket which will be encrypted so the code cannot be accessed by any unauthorized AWS user. When running the tool, the end user will be notified to encrypt the S3 buckets so ensure that all AWS S3 buckets are unencrypted by default. Once the code is stored in the S3 bucket the code can be deployed. Ideally, the tool is best implemented during the AWS CodeCommit section within the AWS pipeline. Before the code gets deployed and before the code can be built, every step should be taken to ensure that the code is a vulnerability-free script and causes no major threat to the infrastructure the user is trying to create.

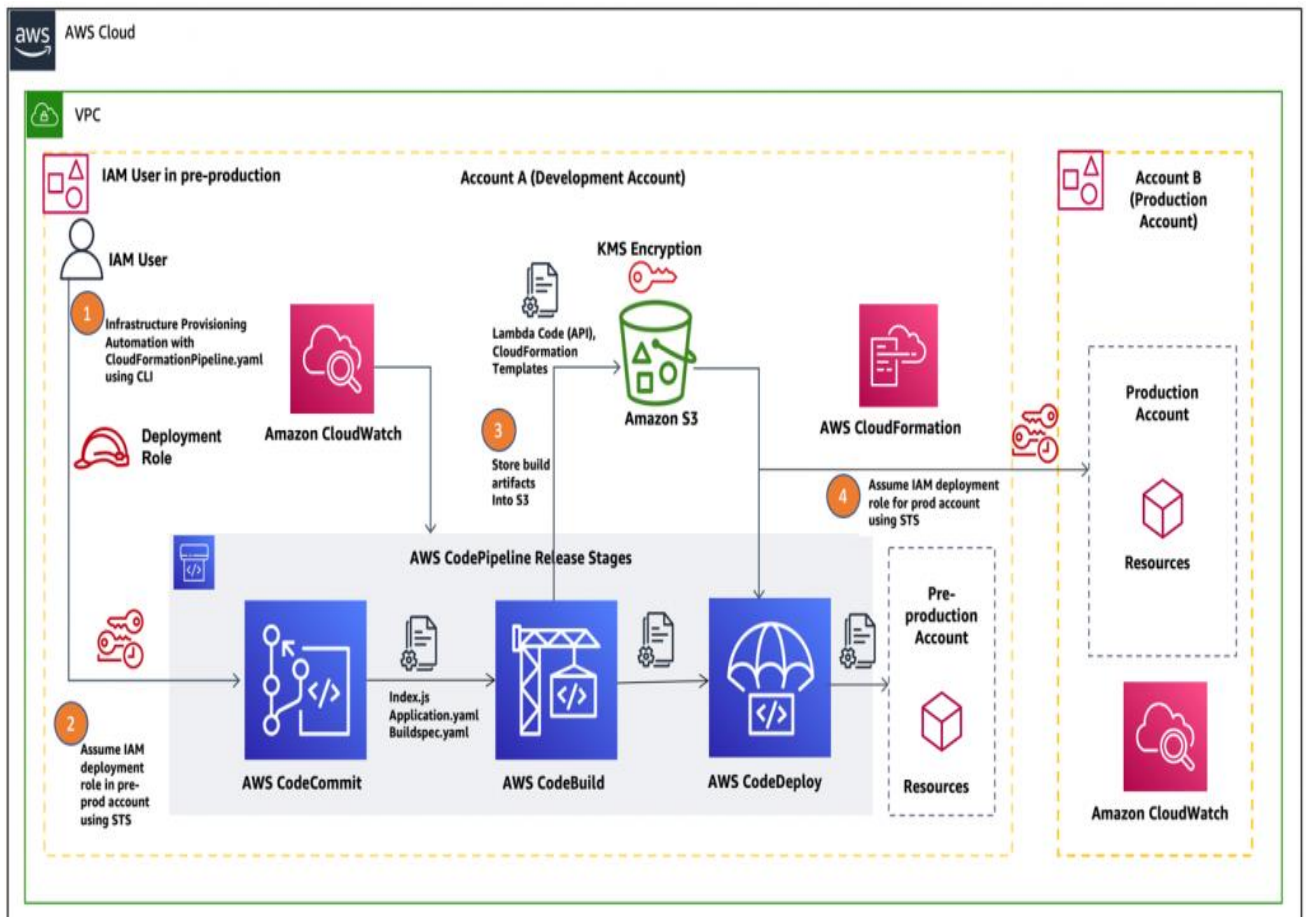


Figure 2 - AWS Code Deployment

Use Cases

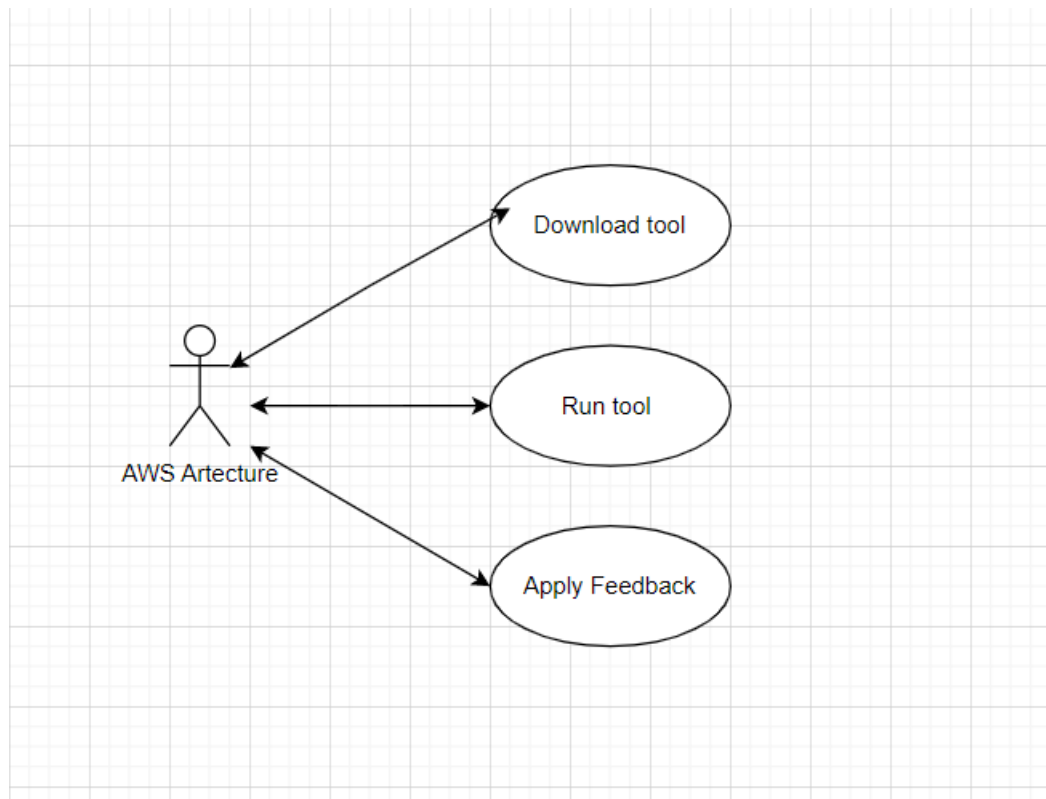


Figure 3 - Use Case

The 'Use Case' diagram highlights the process and graphically depicts the users possible interactions with the tool. The basic steps, while minimal, are identified as:-

1. Download the tool.
2. Scan AWS scripts that are ready to be executed and check for potential vulnerabilities.
3. Provide feedback to the user.

This process will provide users with warnings over their code and more importantly identify what steps are needed to secure the code. This practice should differentiate between critical errors within the script versus warning errors that, while will not be harmful to the infrastructure the user wishes to setup, are errors that can always be improved upon.

Non-Functional FURPS

Functionality

The functionality of this tool is designed to allow AWS architectures and developers to take this tool and scan their own AWS CloudFormation scripts and be given feedback on how to secure the script in a better manner, from a cyber security point of view and facilitate running in a smoother environment. The tool should be available to those who used AWS Cloud scripts for their environments, providing safety and reassurance to the developers working on these scripts. Once the tool has been created and validated, it is hoped that the tool can be implemented on every script that needs to be run and will prove to be a valuable asset, providing additional security to all companies.

Usability

One of the primary efforts of this project is to ensure that the final tool easy to use and it is recognised as a viable asset rather than an impediment for the end user. The goal is to provide the AWS users with security feedback before launching their scripts and that the user interface is easy to navigate through and visually appealing. If all these components are met, then the project will be deemed successful.

Reliability

Consistency and reliability will be hallmarks of this tool, when hopefully adopted by every AWS user, while creating AWS CloudFormation scripts. When the script is scanned, the security feedback to the client should be clear and concise and not bombard the end user with unnecessary comments or errors, i.e. false positives that are not beneficial to the user's script. Feedback is aimed not to overwhelm the AWS architecture or script developer but to give assurance and reliability, so that the AWS architecture feel the tool is a vital necessity. Important also that users can be assured that regular updates will ensure that the tool will always remain bug-free and will scan as many scripts as the developer wishes with the same speed and intensity after each use.

Performance

The performance of the tool is of prime importance when launching a new tool. The tool needs to be beneficial, quick and easy to use for the developer. If the tool is an inconvenience and slow when scanning the script, multiple issues will arise. The aim of this project is that this tool should be fast to identify and provide the end-user with the corrections for scripts and without detecting false positives. For profit focused, fast-paced working environments, time is money and consequently tools and applications which can prove to be both economical and beneficial will be selected for use.

Supportability

When it comes to supportability this tool will scan all AWS CloudFormation scripts and be integrated into the AWS webpage so, all scripts created will be examined before the script is published by the developer or published on the AWS webpage to the public. With the tool set open-sourced, the opportunity to capture feedback from clients and/or AWS users will prove positive to inform of any problems which may arise when trying to use this static code analyzer. Through this, maintenance and upgrades can be implemented to ensure the tool is updated, supported and working to its optimal capacity. The main goal of this product is to make sure the clients are satisfied with how the tool runs and more importantly the benefits which accrue to the clients. If the clients are happy with the tool then they will have confidence and continue to use the tool on all types of scripts.

Metrics Plans for the Project

The main goal for this project is to get a static code analyser to scan AWS CloudFormation scripts or templates and provide security feedback to the code AWS architecture or Developers setting up the scripts. One of the primary functions of this tool is to minimise the security flaws that may be present in an AWS CloudFormation script. If successful, it will provide security feedback to the developers regarding the script; will recommend better security practices to the developers working on the script and confirm when the script is safe to launch. It will also notify the user of any potential misconfiguration or insecure patterns that may be present within the script. An additional objective for this project is to integrate this tool into the AWS system so that it can be used by all AWS architecture that uses CloudFormation scripts & templates.

Design & Description of Application

The application will be developed in Java as mentioned above and the once the application is launched the user will be prompted on what AWS service their file is (S3 for example). Once selected they should be able select their file and then a scan will be done of the users script. The script type must be JSON or Yaml file formatted and any other file extension will not be deployed. The results should be returned to the user in a simple prompt box. The ease of use is a key factor we hope to achieve in this project. The application will be uploaded to github so that if any potential clients want to use this open-sourced tool they can download the files to their own devices and run the tool.

Similar Applications Open-Source or Commercial

CFRipper

Research produced at the foundation stages of this project revealed that a similar tool has already been created. Skyscanner Engineering, a Scottish metasearch engine and travel agency have developed a tool known as CFRipper. This tool, developed using python programming language, was designed to prevent vulnerabilities from getting into production infrastructure through AWS CloudFormation scripts. With CFRipper, all code is available with public access on GitHub as they wanted to make CFRipper an open-source tool. Although other static code analysers exist, this is the closest product that offers an opportunity to scan and check scripts in AWS CloudFormation.

```
$ docker run -v /tmp/templates/:/templates -t stelligent/cfn_nag --help
Usage: cfn_nag [options] <cloudformation template path ...>|<cloudformation template in STDIN>

0.5.20

$ docker run -v /tmp/templates/:/templates -t stelligent/cfn_nag /templates/template_example.json

-----
/templates/template_example.json
-----
| FAIL F38
|
| Resources: ["rootRole"]
| Line Numbers: [5]
|
| IAM role should not allow * resource with PassRole action on its permissions policy
-----
| FAIL F3
|
| Resources: ["rootRole"]
| Line Numbers: [5]
|
| IAM role should not allow * action on its permissions policy
-----
| WARN W11
|
| Resources: ["rootRole"]
| Line Numbers: [5]
|
| IAM role should not allow * resource on its permissions policy

Failures count: 2
Warnings count: 1
```

Figure 4 - CFRipper Menu

```
17 from cfripper.model.rule_processor import Rule
18
19
20 class S3BucketPolicyWildcardActionRule(Rule):
21
22     ...REASON = "S3 Bucket policy {} should not allow * action"
23     ...MONITOR_MODE = False
24
25     ...def invoke(self, resources):
26     ...     ...for resource in resources.get("AWS::S3::BucketPolicy", []):
27     ...     ...     ...if resource.policy_document.wildcard_allowed_actions(pattern=r"^(\w*:{0,1}\*$)":
28     ...     ...     ...     ...self.add_failure(
29     ...     ...     ...     ...     ...type(self).__name__,
30     ...     ...     ...     ...     ...self.REASON.format(resource.logical_id),
31     ...     ...     ...     ...     ...)
32
```

Figure 5 – CFRipper Bucket Policy

The above screenshot demonstrates what the application looks like when the tool is run, i.e. the tool is scanning a JSON script. A nice feature CFRipper offers is that it tells the AWS architecture which aspects of the script are the problem. It tells the user which aspects of the file failed and is a critical error in the script. It also tells the user of potential vulnerabilities, that are not as serious as warnings, which are present in the script. This is a nice function and one which will be ideally replicated in this project. Although while the CFRipper tool is accessed through the command line, the hope of this project is to implement the tool with a user interface, so it is easier to read and navigate.

SonarQube

SonarQube is an open-source static code analyzer that works to minimise security defects in code and completes it in a quick manner. Its automated system was designed to detect bugs and code errors that may have appeared in the code. Although the tool focused upon in this project has been designed to work on AWS CloudFormation scripts, SonarQube is used to examine code errors in multiple programming languages. The goal of SonarQube is to inspect segments of codes which may have issues and vulnerabilities present. SonarQube also monitors code quality which is a feature to ideally be replicated in this project.

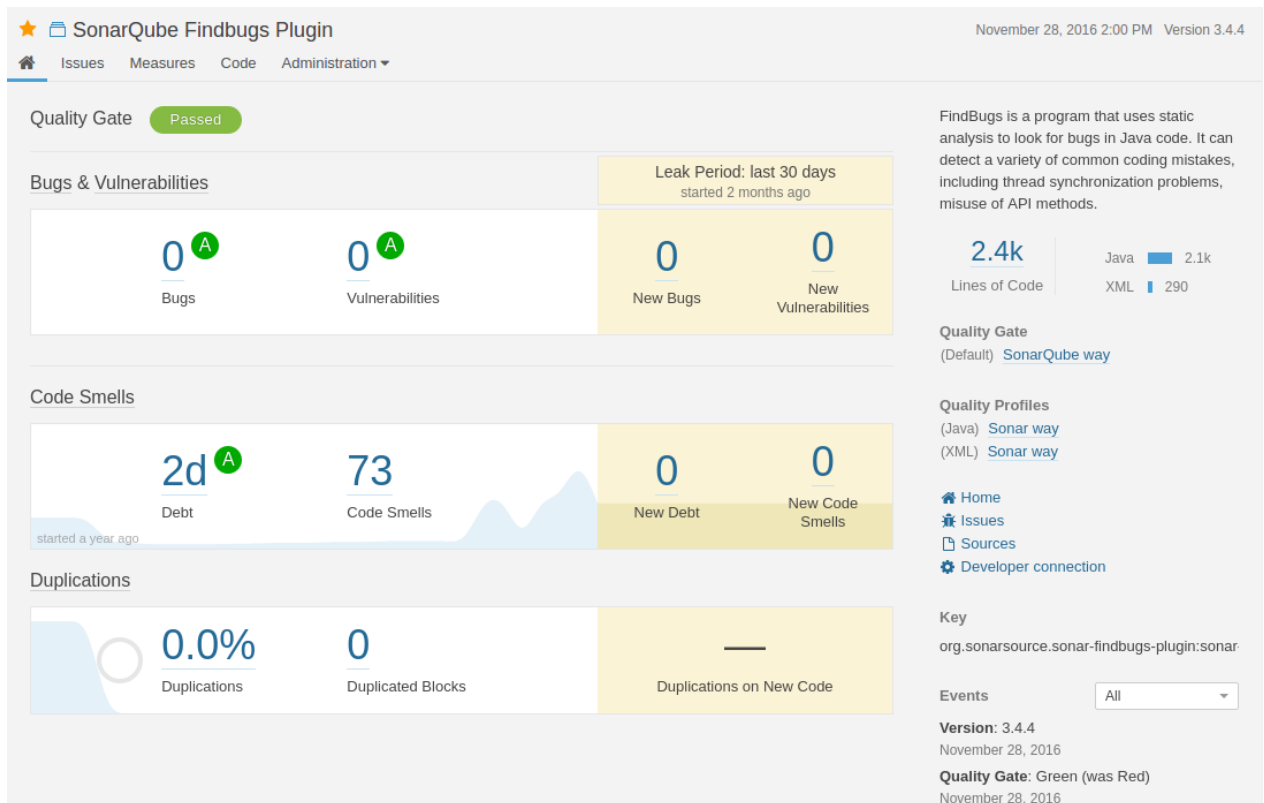


Figure 6 - SonarQube

Codacy

Codacy is a code quality checker that maintains and monitors code. This project will duplicate similar functionality as the tool can be imbedded and work alongside AWS CloudFormation scripts. By viewing and running the scripts on Codacy, an understanding can be gleaned on how this tool can support AWS CloudFormation scripts.

CloudSpolit

CloudSpolit is an open-source AWS scanning tool. It checks against set predefined rules that are best practice for AWS security. The tool identifies misconfigurations, insecure patterns and more critical vulnerabilities. It analyses AWS services like EC2 instances, S3 buckets etc. Its automatic security checks and integration of the tool into systems are a key benefit to having and using this tool.