Cloud Security Scanner and validator

Carter Cutsforth

4590X Final Report

Project Type: Implementation, done solo

Codebase Viewable at: https://github.com/VirgCuts/4590X-project

### 1. Problem Definition

Cloud services can be a handful for developers who are new to them, especially on the security front as a lot of resources offered by aws can be made very vulnerable. Many developers then need help in securing basic services offered by aws such as having properly set up bucket servers or IAM users with proper permissions for their role. The aim of this project was to address this problem by implementing a validation tool that can scan an aws environment for common security vulnerabilities on aws services, specifically IAM users, S3 buckets, and EC2 instances and provide feedback and remediation to help a developer fully protect their systems. The tool is also designed under the assumption that a developer has already deployed aws services and can configure a role with permissions (read-only rights and bedrock access) such that the project can scan through their environment and provide guidance. The end goal was to create an easy-to-use and lightweight program which can be used in post-deployment auditing to help ensure the security of a cloud resource.

### 2. Conceptual Design

For the design of the project it is split into 4 main components.

1.AWS Resource Scanners (EC2,IAM,S3)

Uses the AWS SDK to scan each service and identify potential misconfigurations (public port access, overprivileged role).

2.Issue Object

Acts as a container for possible vulnerabilities

Contains

* Severity (Low/Medium/High/Critical)
* Title
* Description
* Recommendation
* Remediation Steps

3.Security Assistant

Provides User Interface and compiles all issues from each scanner.

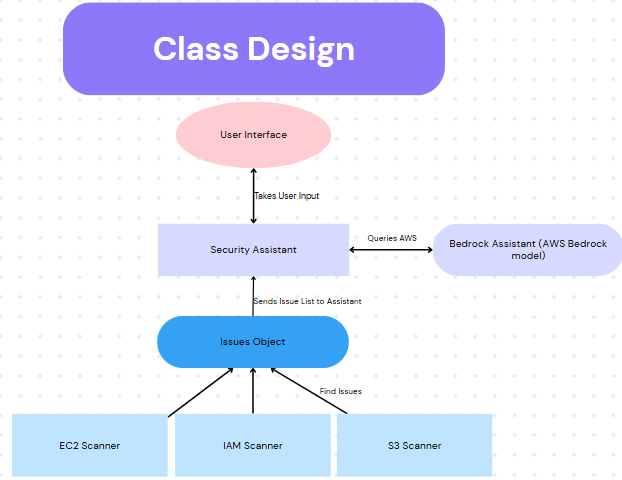
Interface provides information on all issues including remediation help which then

queries bedrock assistant for answers

4.Bedrock Assistant

AI Model from aws, for this project I used Amazon Nova Micro to help in g generating remediation steps and security information.

Called by Security Assistant for remediation help and for general queries to AI



Basic Example of Design and interactions between systems

### 3. Implementation Description

Languages and Tools Used

Java 17+

AWS SDK for java v2

Maven (build/run project)

Amazon Bedrock (Nova Micro Model)

Major Classes:

-S3Scanner.java, EC2SecurityGroupScanner.java, IAMSCanner.java

Scan and compile AWS resource issues

-SecurityAssistant.java

Handles issue collection, user interface, invokes BedrockAssistant

-BedrockAssistant.java

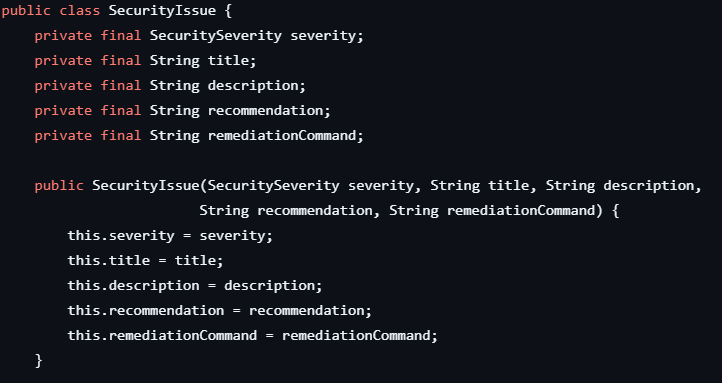
Connects to bedrock model, queries model with prompt for ai generated

remediation and recommendations.

Data Structures:

-SecurityIssue.java

Data class that makes up all parts of an issue



Example of Security Issue structure

Build Process:

Simple with Maven for handling dependencies and compilation or project

Maven allowed compilation into a single JAR file for ease of use aswell

### 4. User Guide

Step by step recommended guide

1. Install Java (17+)
2. Create and configure AWS IAM User, needs Read-only access to S3, IAM ,EC2, plus Bedrock use access and Amazon Nova Micro Access
3. aws configure, enter in access tokens, region, and set output format to JSON
4. Run project with associated .jar file: java -jar aws-validator-1.0-SNAPSHOT.jar
5. If this fails to run, Maven may be required to run, in which case it should work, otherwise code is located on my GitHub VirgCuts under the 4590X Project (link in case you need to view code https://github.com/VirgCuts/4590X-project).

### 5. Self-Evaluation

The project has successfully achieved all the goals I set out to accomplish with its core design. The system can detect and identify the type of misconfigurations in aws instances, explain the problems clearly and succinctly, and provide remediation help and advice. The program is also quite user-friendly with a lightweight interface and needing minimal interactive between aws and the program. The project is also very open to extensibility, as all that would need to be done is adding more scanners or vulnerability finders to preexisting scanners, which can easily be handled by the assistant. There are a couple of problems, however. 1st is that the bedrock assistant being generated can be quite unhelpful and too generic in responses to properly help a user. Second, the system only supports 3 aws services and is not fully extensive on the entirety of each service, so there is much room for improvement.

### 6. References

1. **Amazon Web Services Documentation** *AWS Documentation*, Amazon Web Services,<https://docs.aws.amazon.com/>, Accessed May 2025.
2. **Getting Started with AWS SDK for Java** *AWS SDK Developer Guide*, Amazon Web Services,<https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started-tutorial.html>, Accessed May 2025.
3. **Getting Started with AWS Bedrock** *AWS Bedrock Tutorial*, YouTube,<https://www.youtube.com/watch?v=E1-mUfpeRu0&t=108s>, Accessed May 2025.

### **7. Citations & Code Acknowledgements**

Code interfacing with AWS services reuses snippets from AWS SDK documentation.

Bedrock integration is adapted from YouTube tutorial plus bedrock documentation.

### **8. Group Contributions**

This project was completed as a **solo project**. All research, design, coding, and writing were done by Carter Cutsforth.