



Securing a Serverless Server Via Amazon Web Services

Florida International University
Knight Foundation School of Computing and Information Sciences

Team Members:

Marcos Yauner - *Lead*
Sebastian Pina
Nicholas Sadiku
Carlos Leal

Mentor: James Beswick & *Amazon Web Services Support*

Instructor: Dr. Masoud Sadjadi

Table of Contents

Table of Contents.....	Page 1
Abstract.....	Page 2
Introduction.....	Page 3
Current Website Deployment.....	Page 3
Proposed Website Deployment.....	Page 4
User Stories.....	Page 4
Implemented User Stories.....	Page 4
Pending User Stories.....	Page 5
Project Plan.....	Page 5
Hardware and Software Resources.....	Page 5
Sprint Plans.....	Page 6-7
System Design.....	Page 7
Installation Guide.....	Page 8
Risk Assessment.....	Page 9
Vulnerabilities.....	Page 9
Threat Identification.....	Page 9
Risk Rating.....	Page 10
Risk Controls.....	Page 10-11
Shortcomings.....	Page 11
Glossary.....	Page 11
References.....	Page 12

Abstract

This project aims to implement a robust security framework for securing a serverless server on Amazon Web Services (AWS). The project will use various AWS security services and tools to ensure the confidentiality, integrity, and availability of the serverless architecture. The project will begin by setting up strong identity and access management controls using AWS IAM and Amazon Cognito. It will then configure AWS security services such as AWS Lambda, Amazon API Gateway, and AWS WAF to protect against common security threats such as DDoS attacks, SQL injection, and cross-site scripting. Additionally, the project will implement encryption at rest and in transit using AWS KMS and SSL/TLS certificates. Finally, the project will set up monitoring and logging using AWS CloudTrail, AWS Config, and Amazon CloudWatch to detect any security incidents and ensure compliance with security policies and regulations. By the end of the project, the serverless server on AWS will have a robust security framework that can withstand various security threats and provide a secure environment for hosting applications and data.

Introduction

AWS stands for Amazon Web Services, and it is a cloud computing platform that provides a wide range of services and tools to help businesses and individuals build and manage their IT infrastructure. It is a subsidiary of Amazon.com Inc. and is one of the most popular cloud computing platforms used globally.

AWS provides a flexible and scalable solution for businesses to run their applications and services on the cloud, without the need for costly and complex on-premises infrastructure. With AWS, customers can easily deploy and scale their applications to meet their changing business requirements, pay only for what they use, and benefit from the high reliability, security, and performance of the AWS platform.

Current Website Deployment

Traditional servers present various challenges when it comes to hosting and maintaining a website. Setting up the server requires extensive knowledge in hardware, software, and networking, which can be a complex and time-consuming process. Maintaining the server to ensure smooth operation requires continuous monitoring, updates, and security patches. If any issues arise, troubleshooting and problem-solving may require specialized knowledge.

Another challenge when using traditional servers is scaling to handle increased traffic. This can be particularly difficult if traffic spikes occur unexpectedly, requiring additional servers to be installed and configured. Securing a server requires a strong understanding of security best practices and the implementation of security controls such as firewalls and encryption.

Proposed Website Deployment

AWS offers a variety of services, including computing, storage, database, analytics, machine learning, security, and more. These services are provided through a global network of data centers that are spread across different regions and availability zones. Customers can choose from different types of services based on their needs, such as virtual servers (EC2), object storage (S3), relational databases (RDS), and many others.

AWS provides a flexible and scalable solution for businesses to run their applications and services on the cloud, without the need for costly and complex on-premises infrastructure. With AWS, customers can easily deploy and scale their applications to meet their changing business requirements, pay only for what they use, and benefit from the high reliability, security, and performance of the AWS platform.

User Stories

In the section that follows, you will be presented with the User Stories that were implemented during the creation of the Securing a Serverless Server project. The development of the project's features was based on these user stories. These User Stories will also be taken into account for future development.

Implemented Stories

- Ride wait time is to be included on the Ride description page.
- Limiting EC2 Administrator Full Access policy in IAM.
- Momentarily displaying the uploaded picture, from the add photo feature, in the ride description page to reduce temporary memory storage.
- Set object parameters in the Lambda App.

Pending User Stories

- Creating a second backend directory, in cloud9, to separate objects and git-repository parameters
- A vivid, bold user interface.
- Importing a second git-repository with different image and object files.

Project Plan

The following section will go over the planning that the team undertook to bring the project idea into creation. This project involved an AWS lecture course and several apps from the Amazon Web Services platform which required sprints to be planned according in order to meet the required timelines. This section will also go over the components and AWS services used.

Hardware and Software Resources

Under the AWS umbrella, the following apps were used:

- Desktops/Laptops
- Discord App (Communication)
- Amazon Web Services Management Console
- Cloud9
- IAM
- CodeCommit
- Lambda
- S3
- DynamoDB
- AWS Amplify
- AWS Cognito
- JavaScript
- JSON

Sprints Plan

Sprint 1

- Upon speaking with the product owner via email, it was decided to maintain focus on the AWS knowledge checks and progress through the modules
- Worked on AWS Modules 1 -3. We proceeded with going over all relative lecture videos and knowledge checks.
- Module 3 Lab implementing what we learned and getting some practical applications

Sprint 2

- After discussion, the velocity of the team was estimated to be one module per week in Module 4 as mentioned before.
- Per AWS module, we will be reviewing two lectures per day. We will work on learning about the AWS security groups, network ACLs, and load balancers.
- Our main goal is to review the Set up of Public and Private Subnets in internet protocols today.
- We had slight difficulties completing the Module 5 lab but were able to complete it.

Sprint 3

- Commence module 6 and review the Logging and Monitoring introduction and importance lecture video.
- The team came to an agreement to complete all module labs by March 5 and to start working on the AWS project on March 6, to have a full month of research and work on the project.

Sprint 4

- The team went over possible project proposals that were AWS Security themed. We will work on the development of it all March in collaboration with the professor's assistance.
- The team will be divided into two smaller teams. One focused on building the AWS serverless server and the other getting a head start on the documentation of the poster and product

Sprint 5

- Upon speaking with the product owner via email, it was decided to maintain focus on the AWS knowledge checks and progress through the modules
- We discussed the new timeline schedule since the deliverables are fast approaching

Sprint 6

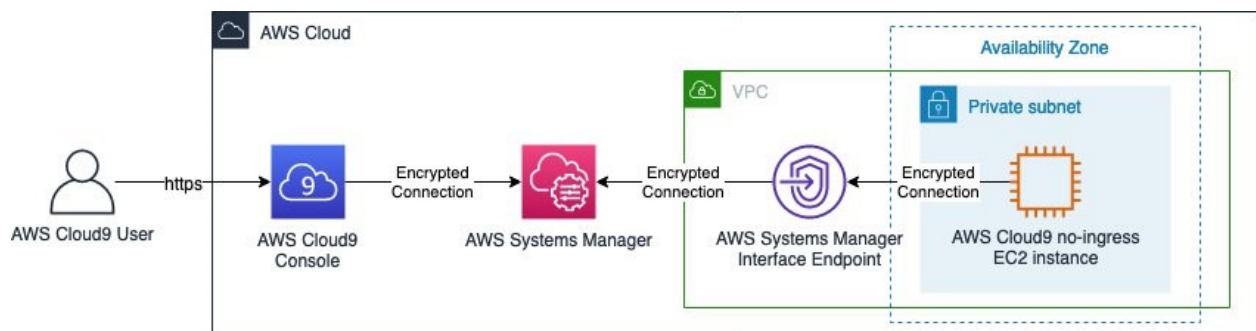
- The Team will continue its two-team commitment in one completing the project documentation and the AWS serverless server development.
- We will focus this week on completing the poster and PowerPoint by Thursday in order to have ample time to work on AWS.
- The poster and User Manual are complete

Sprint 7

- Completed the posters and the team finished their individual ones, we proceeded to complete the Demo video, user manual, and installation guide.
- We will be focusing on the intro video, project documentation, PowerPoint, and Senior presentation.

System Design

The system design and requirements involve creating a development environment in Cloud9, using CodeCommit for version control, configuring the serverless backend using Amplify, and setting up the DynamoDB table, Lambda function, and API Gateway API for the serverless backend.

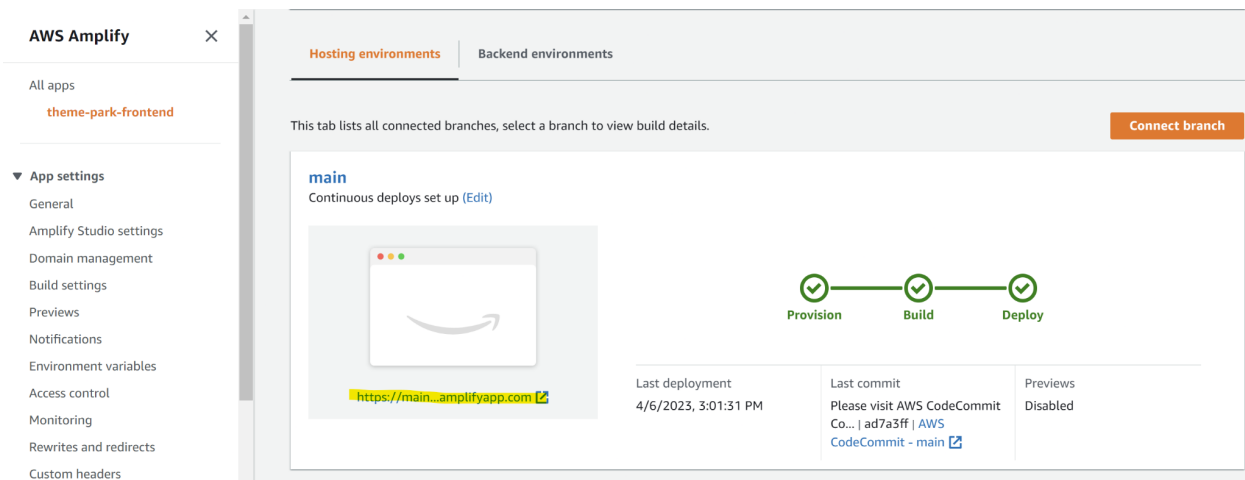


Installation Guide

No Installation Needed

1. Since we created a Serverless web application, there is nothing to install; just visiting this website will allow you to access it:

<https://main.d2p8utow75bnsx.amplifyapp.com/#/>



2. Also below is the provided QR Code for quick access to the deployed website.

QR Code:



Risk Assessment

For our Theme Park website deployed via AWS Amplify and its associated apps, the below is the risk analysis from what our Serverless Server may encounter:

Vulnerabilities:

1. Injection Attacks: Attackers could inject malicious code into the application, especially if the application is not properly validating user inputs.
2. Denial of Service (DoS) Attacks: Attackers could launch a DoS attack on the serverless application, causing it to become unresponsive or unavailable.
3. Authentication and Authorization Issues: Insufficient authentication and authorization controls could allow unauthorized access to sensitive data and functions within the application.
4. Insecure Serverless Deployment: If the serverless application is not properly configured, it could expose sensitive data or functions to unauthorized users.

Threat Identification:

1. External Attackers: Hackers, malicious actors, or competitors attempting to disrupt the website or steal information.
2. Internal Attackers: Rogue employees, contractors or other insiders with access to the system who may have malicious intent.
3. Accidental misuse: Unintentional errors by authorized users, such as misconfigured functions or overloading the system, can cause significant damage.

Risk Rating:

1. Injection Attacks: High risk
2. Denial of Service (DoS) Attacks: High risk
3. Authentication and Authorization Issues: Medium risk
4. Insecure Serverless Deployment: Medium risk

Risk Controls:

Below are the following parameters set with the deployed theme park website, as well as measures taken that will be implemented in response to the vulnerabilities described in the Vulnerability section.

Vulnerability Controls

1. Injection Attacks: Mitigate the risk of injection attacks by implementing input validation and sanitization techniques, and using tools like AWS WAF or third-party firewalls.
2. Denial of Service (DoS) Attacks: Prevent or mitigate DoS attacks by implementing rate-limiting, throttling, and monitoring of traffic. Also, consider using AWS Shield or third-party DDoS protection solutions.
3. Authentication and Authorization Issues: Reduce the risk of unauthorized access by implementing MFA, strong password policies, and using AWS IAM to control access to resources and functions.
4. Insecure Serverless Deployment: Secure the serverless application by following security best practices such as encrypting data at rest and in transit, ensuring function isolation, and conducting regular vulnerability assessments.

Implemented Controls

1. IAM: Assign User roles and policies to comply with C.I.A best practice.
2. Cloudtrail: To trace and monitor movement across AWS.
3. Lambda: Assign Functions and API Gateways with set parameters.
4. S3 Bucket: Store our objects with set parameters revoking public access.
5. Amplify: Securely deploy our app with automatic security updates, encryption, IAM, Application-level security and compliance.

SHORTCOMINGS

There weren't any major shortcomings with our project. As a team, we were able to resolve all the tasks given to complete the project. Even though we had several shortcomings at the beginning of the development phase, such as miss management of user policies, with a couple of errors we then had to restart our project. But those errors helped the team understand the User Policy fundamentals and IAM roles of AWS better. We then proceeded to develop the serverless backend a second round with no errors this time around.

Glossary

Cloud9 IDE: Cloud service with resizable compute capacity.

CodeCommit: Secure Git repository for code management.

AWS Amplify: Platform for scalable cloud apps.

AWS S3: Scalable and durable object storage.

AWS Lambda: Serverless code execution.

API Gateway: API development, deployment, and management.

AWS DynamoDB: Fully-managed NoSQL database.

AWS Cognito: Managed authentication and user management.

References

Beswick, J. (n.d.). Innovator island. Innovator Island. Retrieved April 6, 2023, from <https://www.eventbox.dev/published/lesson/innovator-island/index.html>

Beswick, J. (2021, April 26). Build a serverless web app for a theme park: Episode 1 - AWS virtual workshop. YouTube. Retrieved April 6, 2023, from <https://www.youtube.com/watch?v=GhZpSYQ6F9M&t=2804s>

Team, T. S. (2022, June 8). Total cost of ownership of servers. Sherweb. Retrieved April 6, 2023, from <https://www.sherweb.com/blog/cloud-server/total-cost-of-ownership-of-servers-iaas-vs-on-premises/>