

Secure File Sharing System Using AWS Serverless Services

Cloud Computing & Security – Final Project Report

Abstract

This project designs and implements a secure file-sharing system using the facilities of cloud services. The general idea of the project is to let users securely upload files and share them with temporary links to access instead of storing them in an openly available way. The system is developed using AWS serverless services such as AWS Lambda, Amazon S3, and AWS IAM.

The uploaded file is stored in a private Amazon S3 bucket, along with generating a presigned URL with limited expiry time. This model design keeps the security implications in consideration and prevents unauthorized access. The project covers cloud computing and cloud security perspectives. Testing results verify that the system functions correctly and securely.

1. Introduction

1.1 Problem Description

File sharing is a common phenomenon in modern applications, but in many cases, files are stored in public cloud storage. In cases where the links for files are distributed inadvertently, anyone can gain access to the files. This poses a significant threat to security in cases where private files are involved.

1.2 Motivation

The motivation of this project is to design a simple and secure method of sharing files using cloud services. Instead of public access, temporary links with expiry times are used. It is applicable for the secure sharing of reports, documents, and other confidential data.

1.3 Project Focus

The project focuses on:

- Cloud Computing
- Cloud Security

1.4 Objectives

- Build a working cloud-based file sharing system

- Use serverless cloud services
- Apply access control and encryption techniques
- Test and validate the system

2. Cloud Architecture

2.1 Cloud Provider

Amazon Web Services (AWS) is used as the cloud provider.

2.2 Cloud Services Used

- AWS Lambda – Backend serverless logic
- Amazon S3 – Secure object storage
- AWS IAM – Role-based access control
- Amazon CloudWatch – Logging and monitoring

2.3 Architecture Overview

The client sends a POST request to the URL of the Lambda Function. The validation of the request is carried out, and then the image is uploaded to a private Amazon S3 bucket with the use of IAM roles by the AWS Lambda function. After a successful upload, a presigned URL with expiry is sent to the user

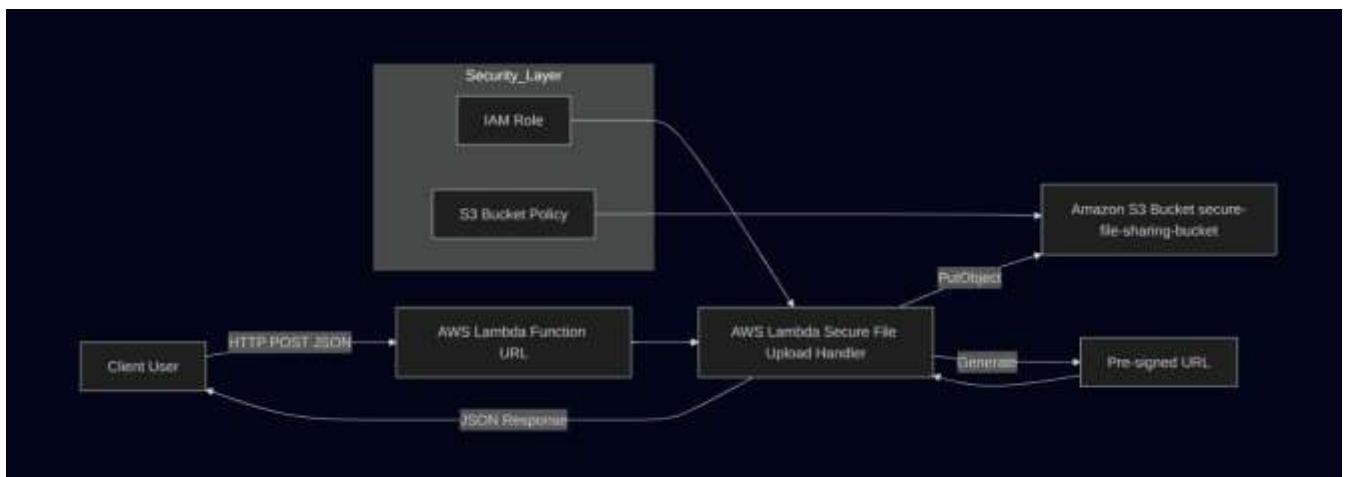


Figure 1: Overall cloud architecture diagram (Lambda – S3 – IAM flow)

3. Core Implementation

3.1 Main Components

AWS Lambda

- Written in Python
- Handles request validation
- Uploads files to Amazon S3
- Generates presigned download URLs

Amazon S3

- Private bucket
- Public access blocked
- Stores uploaded files securely

AWS IAM

- Lambda execution role
- Least-privilege permissions
- No access keys stored in code

3.2 System Workflow

1. User sends a POST request with filename and content
2. AWS Lambda receives the request
3. Input data is validated
4. File is uploaded to the Amazon S3 bucket
5. Presigned URL is generated
6. Response is returned to the user

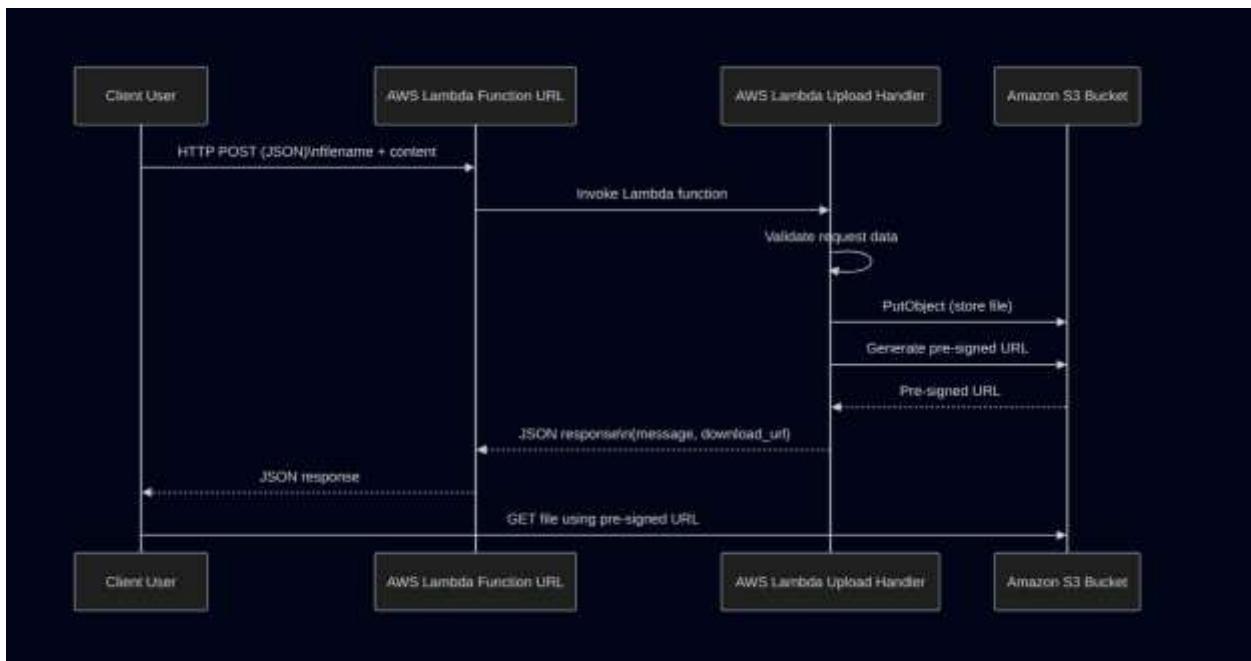


Figure 2: Request flow from client to Lambda and S3

3.3 Example Request

```
curl -X POST <lambda_function_url> \
-H "Content-Type: application/json" \
-d '{
  "filename": "test.txt",
  "content": "this is my real aws lambda upload test"
}'
```

3.4 Example Response

```
{
  "message": "file uploaded successfully",
  "download_url": "https://s3-presigned-url",
  "expires_in_seconds": 300
}
```

Figure 3: Terminal output showing successful file upload

4. Security Analysis

4.1 IAM and Access Control

- AWS Lambda uses an IAM role instead of access keys
 - Permissions are limited to required Amazon S3 actions
 - Unauthorized access is blocked

4.2 Secure Storage

- Amazon S3 bucket is private
 - Public access is disabled
 - Objects are accessed only through presigned URLs

4.3 Secure Communication

- HTTPS is used for the Lambda Function URL
 - Communication is encrypted by default

4.4 Risks and Mitigation

Risk	Mitigation Strategy
Public file access	Private Amazon S3 bucket
Unauthorized users	IAM role-based permissions
Leaked links	Short expiry presigned URLs
Credential exposure	No hardcoded secrets

5. Monitoring and Logging

Amazon CloudWatch is used to monitor AWS Lambda execution. Logs include request details, execution time, memory usage, and error messages. This helps in debugging, performance analysis, and system evaluation.

6. Evaluation and Testing

6.1 Functional Testing

- Valid requests upload files successfully
- Missing content returns an error response
- Invalid requests are handled correctly

6.2 Security Testing

- Direct Amazon S3 access without permission is denied
- Expired presigned URLs do not allow file access

6.3 Performance Testing

AWS Lambda execution time is low, and memory usage remains within limits. The serverless model enables automatic scaling without manual configuration.

7. Challenges and Lessons Learned

7.1 Challenges

- Understanding AWS IAM permissions
- Handling Lambda event input correctly
- Debugging JSON-related errors

7.2 Lessons Learned

- Serverless architecture reduces infrastructure management effort
- IAM plays a critical role in cloud security
- Presigned URLs are effective for secure file sharing

8. Conclusion

The secure file sharing system was successfully implemented using AWS serverless services. The system allows users to upload files securely and access them through time-limited presigned URLs. Cloud security principles such as IAM, encryption, and private storage are effectively applied. This project demonstrates a practical real-world use case of cloud computing and security and fulfills all project requirements.

9. References

1. AWS Lambda Documentation
2. Amazon S3 Security Best Practices
3. AWS IAM User Guide
4. AWS Presigned URL Documentation

10. Appendix

10.1 Deployment Steps

1. Create an Amazon S3 bucket with public access blocked
2. Create an IAM role for AWS Lambda
3. Deploy the Lambda function code
4. Enable the Lambda Function URL
5. Test the system using curl commands

secure-file-sharing-bucket-omar [Info](#)

Objects Metadata Properties Permissions Metrics Management Access Points

Objects (4)

[Delete](#) [Actions ▾](#) [Create folder](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	3f8d4295-69a7-4b07-aa16-94755cdceb9.txt	txt	January 9, 2026, 21:17:28 (UTC+0:30)	89.0 B	Standard
<input type="checkbox"/>	ad821514-e5d6-4d3a-8d9c-905db969542c.txt	txt	January 9, 2026, 21:07:59 (UTC+0:30)	28.0 B	Standard
<input type="checkbox"/>	d2e4dbeb-e3b7-4fa2-#83c-1c5d81b288a7.txt	txt	January 9, 2026, 21:12:18 (UTC+0:30)	89.0 B	Standard
<input type="checkbox"/>	f30a0146-3b42-4fc1-96bc-bb0f422eb49a.txt	txt	January 9, 2026, 21:11:42 (UTC+0:30)	89.0 B	Standard

Figure 4: Amazon S3 bucket showing uploaded objects