Security Analysis of Serverless Functions

sdmay24-26 (Dillon H, Cameron H, Michael G, Samuel P, Trent W)
Client and Advisor: Dr. Berk Gulmezoglu

Background

- What is a serverless function?
- What are some use cases for Serverless Functions?

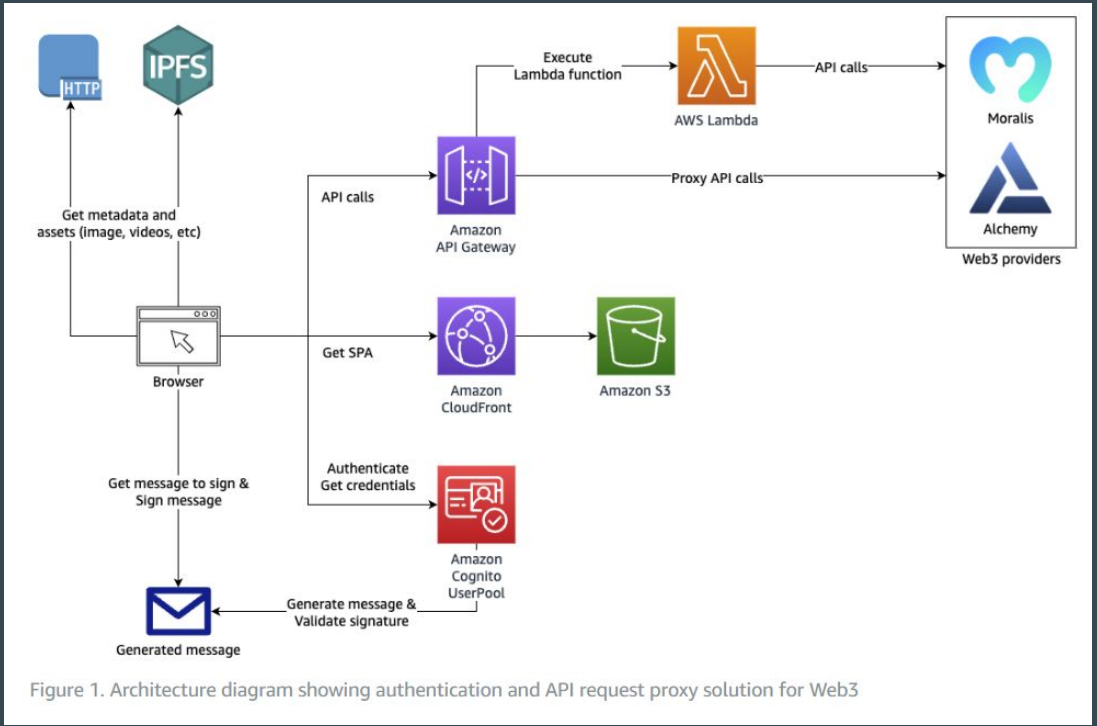
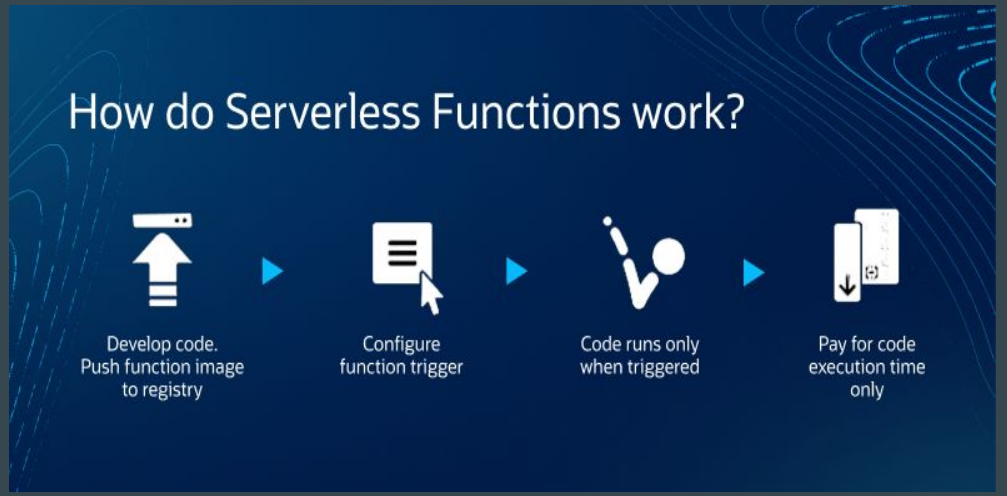


Figure 1. Architecture diagram showing authentication and API request proxy solution for Web3

https://aws.amazon.com/blogs/architecture/dapp-authentication-with-amazon-cognito-and-web3-proxy-with-amazon-api-gateway/

Background

- How do Serverless Functions work
  - Code Deployment
  - Event Trigger
  - Resource Allocation
  - Code Execution
  - Response
  - Resource Deallocation
- Why are these functions advantageous?
  - Cost efficient
  - Scalability
  - Developer Productivity
  - Reduced Overhead



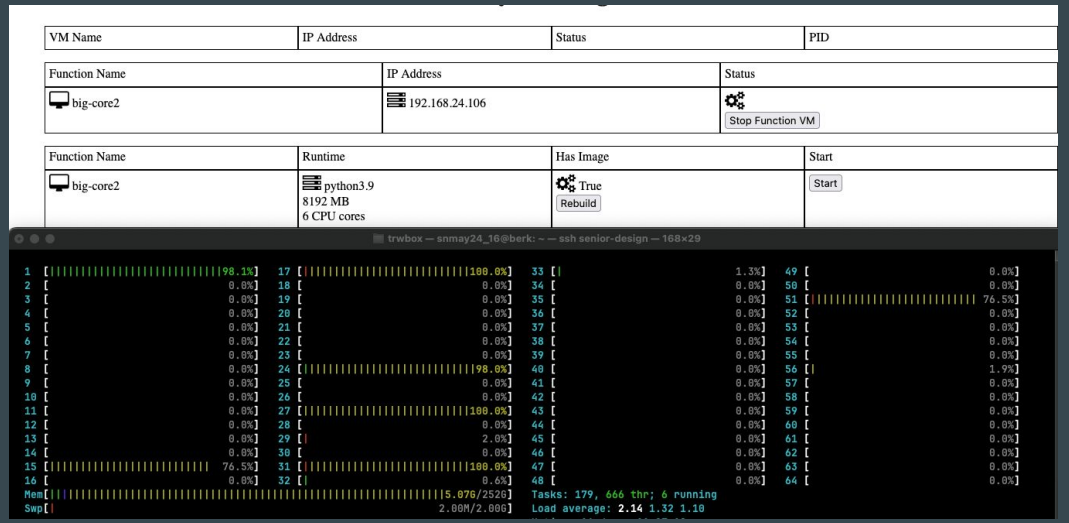
https://developer.oracle.com/learn/technical-articles/serverless-functions

Project Requirements

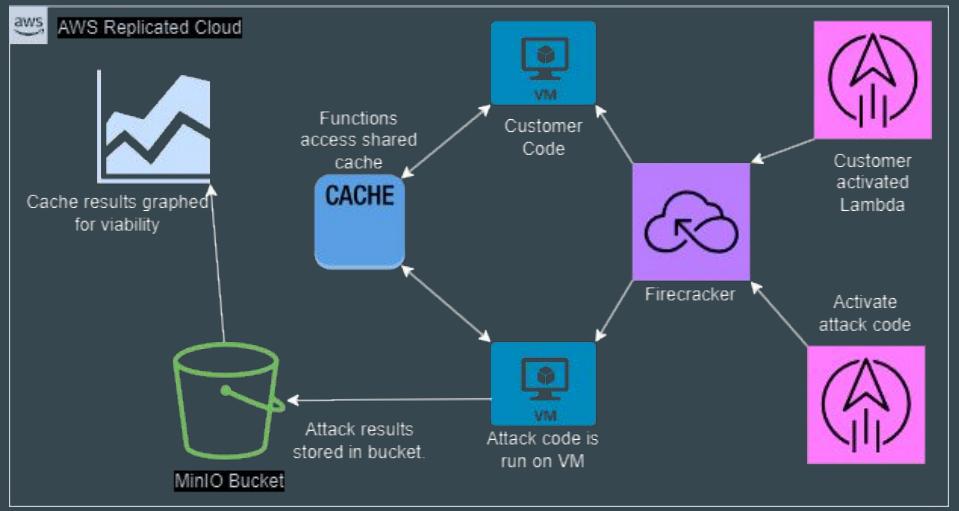
- Functional
  - Documentation, vulnerabilities
- Resources
  - Server, environment setup, Lambdas
- Legal requirements
- Performance
  - Memory, processing, minimum output constraints
- Maintainability
- Testing requirements
  - Environment should mimic AWS as much as possible
  - Consistency of attack code output

Project Design - Backend

- Automated Lambda Creation
- Provide additional AWS services
- Routing of function requests
- Dynamic Networking
- Changeable MicroVM settings
- Uses firecracker for quick VM startup
- Automated enumeration of lambda functions
- Dynamically builds runtime image
- Starting and stopping MicroVMs from an API



Project Design - Frontend



- Lambda
  - Attack code
  - “Customer” functions
- Data collection and transfer
- Graphing - troubleshooting
  - Helped identify viable data to be used in a machine learning model
- Issues faced
  - Multiple languages with different timing mechanisms
  - Customer cache footprinting
    - Running Lambda functions on the same cores/CPU

Results - Testbench

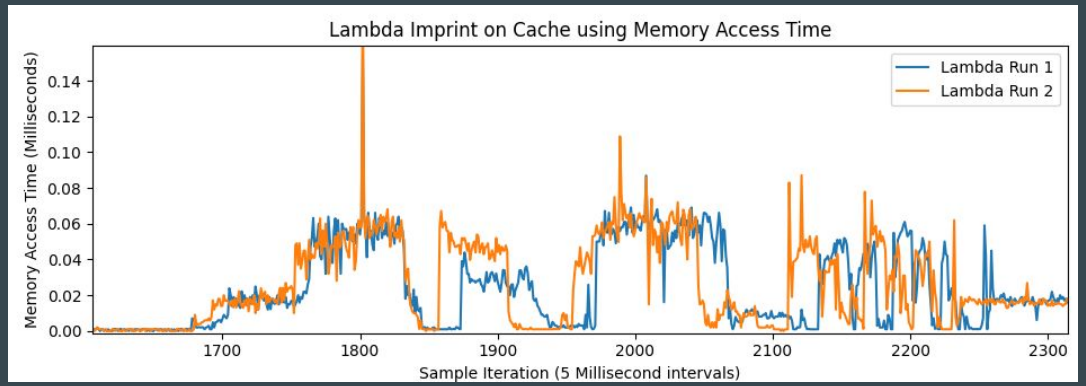
Initial Testing Phase

- Attack code viability
- Useful imprint on cache
- Repeatable

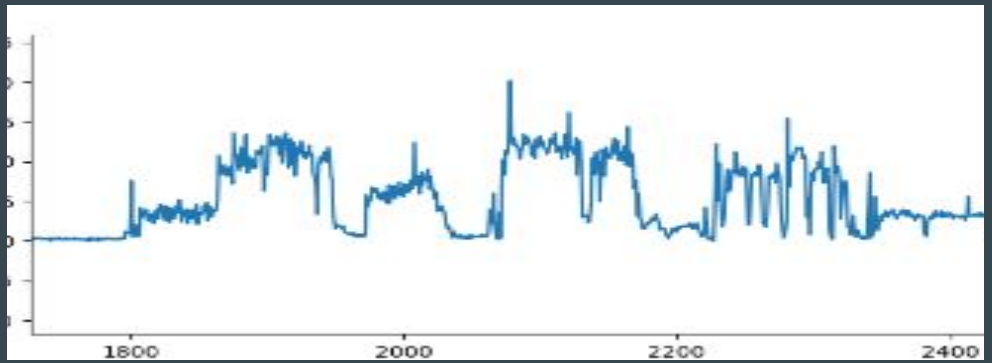
Attack Code as Lambda

- Similar results
- Still using testbench directly

Running Attack Code on Server Directly



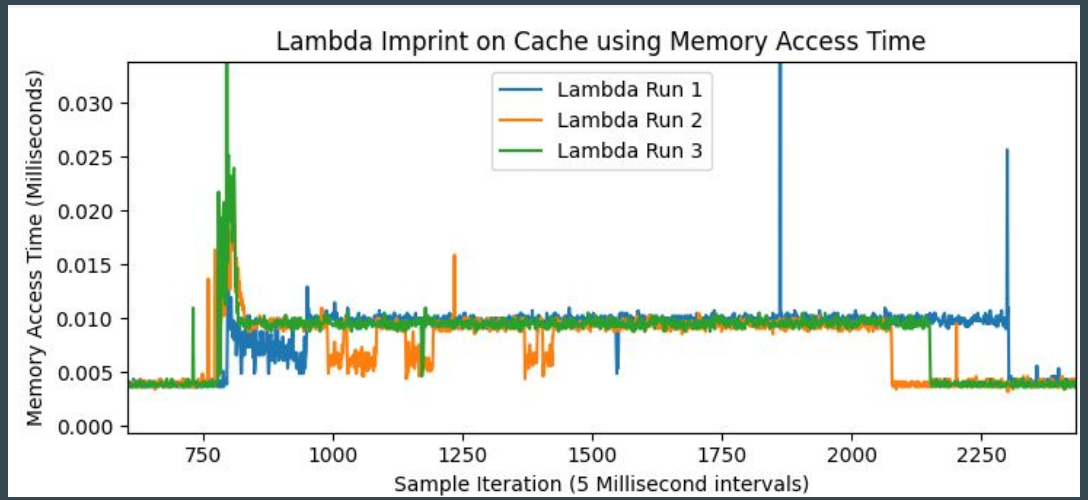
Running Attack Code as a Lambda



Results - Webserver

Attack Code and a Lambda

- Balance between number of cores and imprint
- Finding similarities
- Finding viable candidates for lambda
- Identifiable?
- Clear Start and Stop?
- Same Scale?



Subsequent Actions

- Train and test a ML model for classification of functions
- Testing attack code in real environment (AWS)
- Investigation into minimal viable capture amount
  - How many cores produce viable data?
- Expand interpreted languages used

Conclusion / Demo