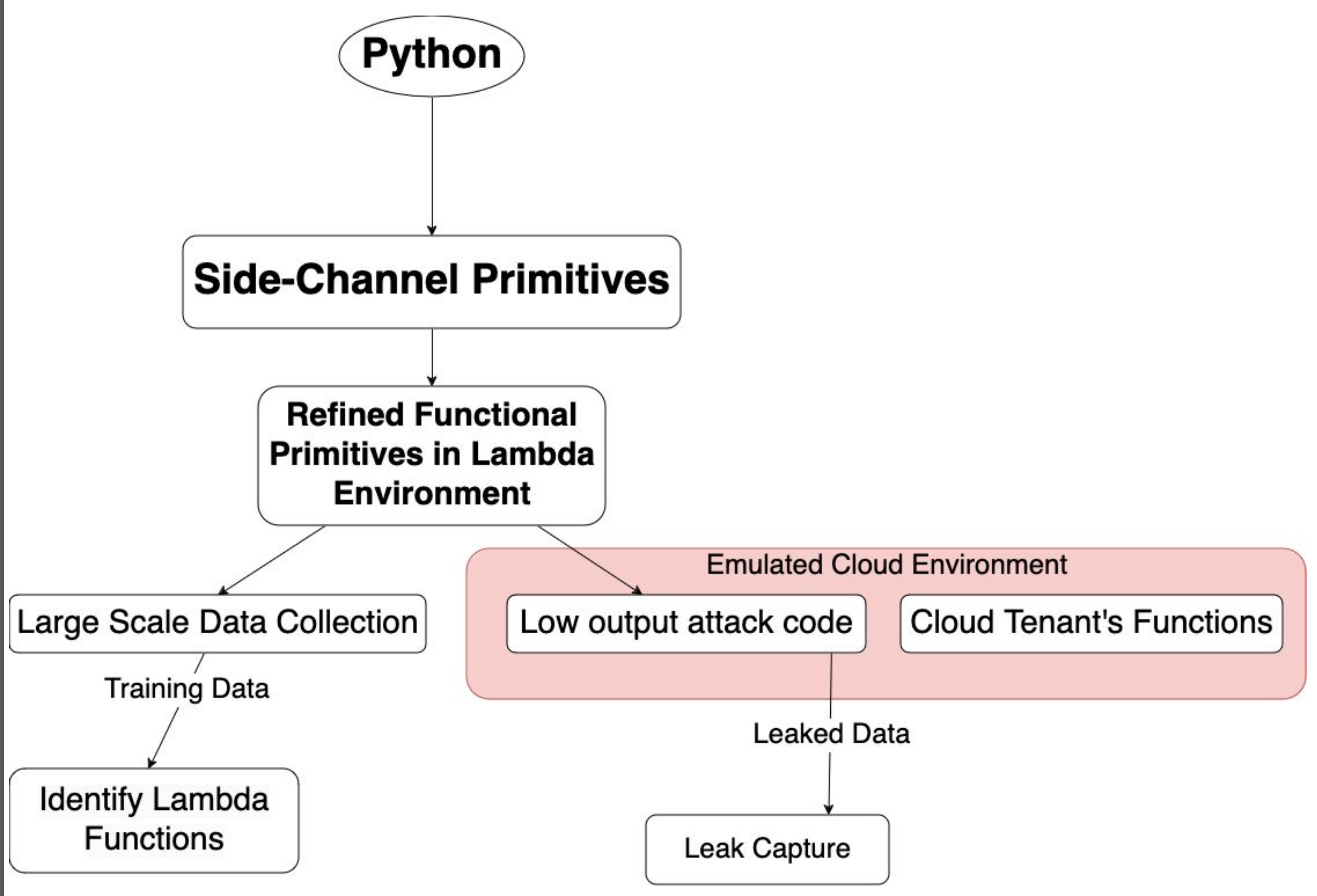
Detailed Security Analysis of Serverless Functions With Interpreted Languages (sdmay24-26)

Client/ Advisor: Berk Gulmezoglu
Team: Trent Walraven, Samuel Potter, Cameron Hurt, Dillon Hacker, Michael Gohr

Introduction

<u>Problem:</u> Serverless functions are widely offered as a service by cloud providers. However, their security is underexplored.
<u>Solution:</u> Create a side-channel attack to identify Lambda functions being run. Then create a security analysis that will highlight our findings and show potential vulnerabilities in serverless functions.
<u>Intended Users:</u> Cybersecurity professionals and technology enthusiasts.
<u>Intended Uses:</u> Any lambda function users. These functions are widely used by both organizations and individuals.

Design Approach


Python
Side-Channel Primitives
Refined Functional Primitives in Lambda Environment
Large Scale Data Collection
Training Data
Identify Lambda Functions
Emulated Cloud Environment
Low output attack code
Cloud Tenant's Functions
Leaked Data
Leak Capture

Technical Details

<u>Concerns:</u>
- Successful replication of Amazon's environment
- Finding suitable Lambda functions
- Viability of side channel attacks with protections already in place

<u>Limitations:</u>
- Do not have access to Amazon's environment
- Limited by the amount of tests we can run on our shared server
- Limitation of small execution footprint of serverless functions

Design Requirements

<u>Functional Requirements:</u>
- Code that is well documented, and able to be understood by anyone looking to translate the code into another SDK for a different cloud provider
- Functions are short-lived, anything defined as a vulnerability needs to actually run in the cloud environment constraints

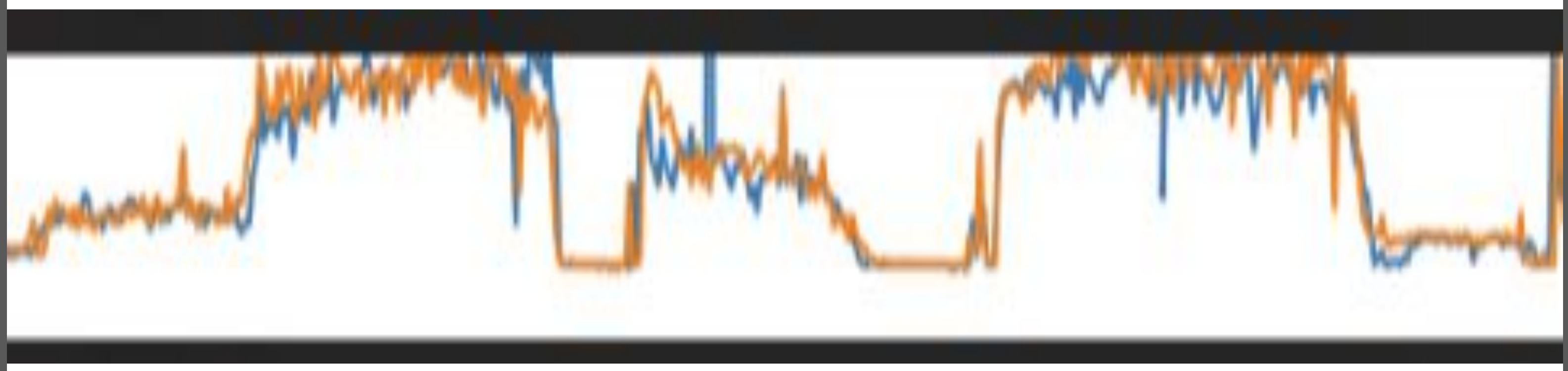
<u>Non-Functional Requirements:</u>
- If there is reasonable concern of a vulnerability existing, go through the proper channels to safely report the issue, and only publish information after a fix is implemented
- Do not put public cloud tenants at risk.

<u>Constraints:</u>
- We do not have access to Amazon's server so we are having to replicate Amazon's servers and test everything locally.
- Constrained to interpreted languages that are accepted by Lambda functions.

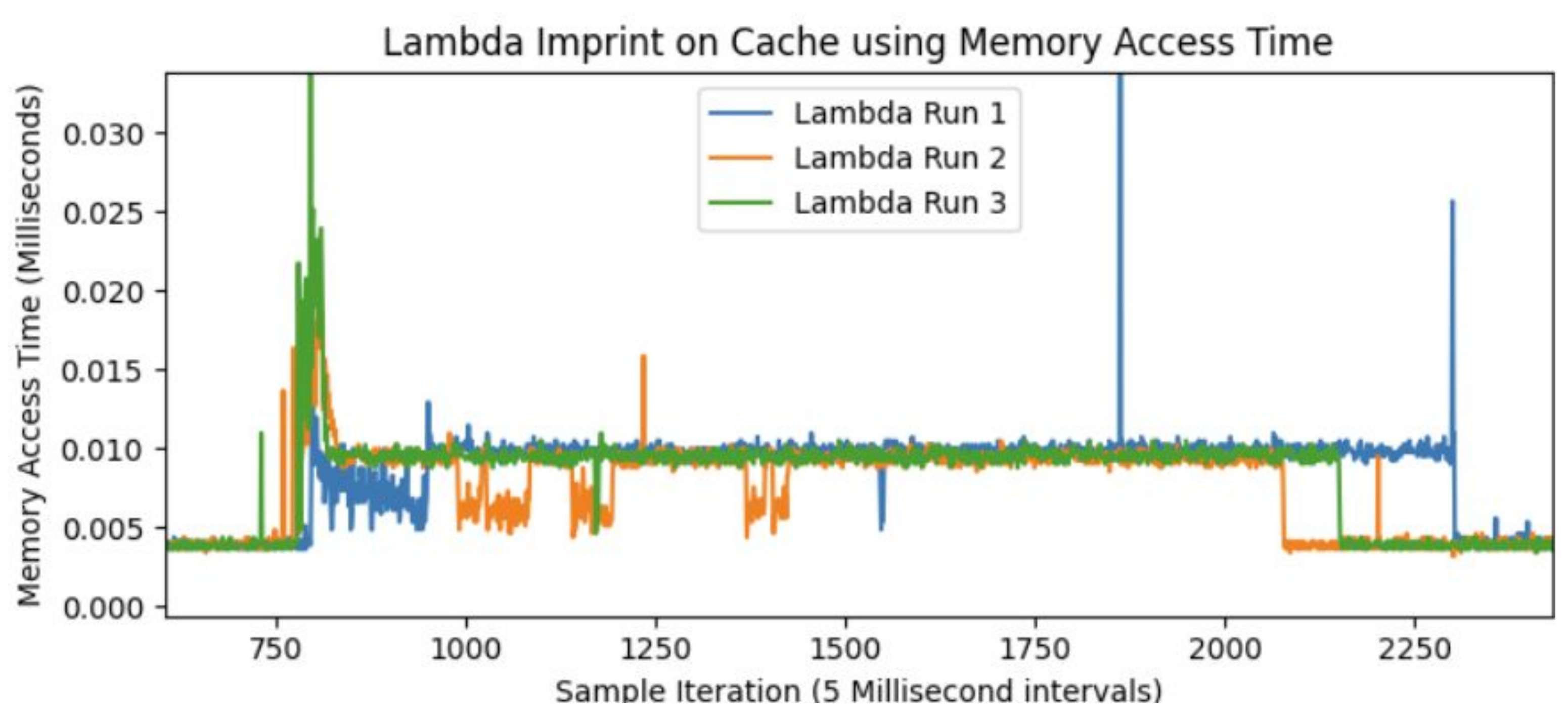
Testing

<u>Environment</u>:
- All testing done outside of the public cloud
- This had some limitations but insured a safe testing environment

<u>Strategy:</u>
- Started testing using a testbench



- Continued testing using our Webserver and related components



<u>Results:</u>
- Results were more consistent when using the testbench
- Multiple high usage functions cause a noticeable impact on the cache