Go Serverless: Secure Cloud via Serverless Design Patterns

Sanghyun Hong°, Abhinav Srivastava*, William Shambrook*, Tudor Dumitraș°

°University of Maryland College Park, MD USA
*Frame.io, NY USA
Shared Responsibility Model

Tenants

Cloud Providers

[VMs] [Containers]
Shared Responsibility Model – cont’d

[Security services or applications]
1. Monitor intrusions/misuses
2. Malware/vulnerability scanners
3. Real-time data collection pipelines
   - Data transformation
   - Data stream partition/aggregation
4. Automated incident response

[VMs]          [Containers]
Serverless Architecture

[Key intuitions]
1. Spare the management efforts
2. Reduce the size of attack vectors
Serverless Architecture – cont’d

Many architectures have been proposed. But it is hard to find simple design patterns.
Contributions

1. Present **6** serverless design patterns
2. Describe the key **benefits** and security **applications** for each pattern
3. Propose a threat-intelligence platform that utilizes the **6** patterns
4. Discuss the **limits** of Lambda functions and ways to overcome them
A Taxonomy of Serverless Design Patterns

• **Six** Design Patterns (DPs)
  1. DP1: Periodic invocation pattern
  2. DP2: Event-driven pattern
  3. DP3: Data transformation patterns
  4. DP4: Data streaming patterns
  5. DP5: State machine patterns
  6. DP6: Bundling multiple patterns
DP1: Periodic Invocation Pattern

[Applications]

• **Security service**: monitor continuous compliance status (SOC2, CSA, etc.)
• **Others**: archive the data not accessed for an extended time to cold storage
DP2: Event-driven Pattern

[Applications]

- **Security service**: monitor malicious file-uploads to cloud storage
- **Security service**: monitor incoming network traffics at a load balancer
DP3: Data Transformation Pattern [for ETL pipelines]

[Source: incoming streams]  [Transform: enriched or formatted]

Data stream  Lambda functions  Databases

[Transforms]
- **Security-related**: append the Geo-IP information to incoming network requests
- **Security-related**: append the VM or container information where a request is processed
DP4: Data Streaming Pattern [for ETL pipelines]

[Data Partitioner]

Data source → Streaming data 1
Data source → Streaming data 2

[Data Aggregator]

Data source 1, Data source 2 → Streaming data
Lambda functions

[Applications]

- **Partitioner**: report a security incident to multiple channels (e.g., Slack or PageDuty)
- **Aggregator**: append the Geo-IP information to incoming network requests
DP5: State Machine Pattern

[Applications]
- **Security-related**: stabilize data processing [ETL] pipelines
DP6: Bundling Multiple Pattern

[DP2: Send the events and file data]

[Data Partitioner]

Streaming data

Streaming data

[DP4: Partition the data]
Threat Intelligence Platform

[DP6: Bundling DP2 and DP3, 4, 5 to build data processing pipeline]

[DP2: Collect data when a new event comes]

[DP3, 4: Data streaming/transforms]

[DP5: State machine that extends DP3, 4]

Data Collection Component

Notification & Incident Response Components

[DP1, 2: Monitoring attacks]

Go Serverless: Secure Cloud via Serverless Design Patterns 14
Outline

1. Introduction
2. Six Serverless Design Patterns
3. Threat Intelligence Platform
4. Last Mile Problems
5. Conclusion
Last Mile Problems

• Resource constraints
  1. Time-bound execution
     • Problem: Lambda function have a max. execution time limit
     • Solution: Increase the execution time limit or pass state between executions

  2. Lack of computing power
     • Problem: Lambda is insufficient for CPU intensive workloads
     • Solution: Make computing resources configurable or support GPUs

  3. Disk space
     • Problem: Lambda has limited disk space under the “/tmp” directory
     • Solution: Make disk space configurable or support mounting external disks
Last Mile Problem – cont’d

• Limited functionalities
  1. Event tracing
     • Problem: Lack of tools for monitoring event traces in complex serverless systems
     • Solution: Cloud providers support such tools fully integrated with existing services
  2. Security
     • Problem: No security services fully integrated with lambda functions
     • Solution: Services such as vulnerability scanning of lambda function code
Conclusion

1. Lambda can be used as a core component of security services/applications.
   - Minimizes the management effort compared to VMs or containers
   - Reduces the attack vectors from the tenant’s space

2. We identified the six serverless patterns that utilize lambdas
   - Each pattern has key benefits and can be commonly used in various services/applications
   - Combining multiple patterns allows building large-scale and complex security systems

3. Lambda has several limits to be used in various domains
   - Require to solve resource constraints and to provide more functionalities
   - Open up more research questions in the serverless field
Thank you!

Sanghyun Hong
shhong@cs.umd.edu
Q & A: Is Lambda Secure?

- [MITM: encrypt using AWS KMS]
- [Unauthorized access: enforce access policies]
- [Malware: deploy IDSe]
- [Side-channel: use SGX/SEV]
Q & A: Cost & Scalability Analysis

• Task [that transforms incoming network requests]:
  • Execution time: 100ms - 5min.
  • Allowed latency: 100ms - 500ms.
  • Size: 200 req. logs per minute, where each log has 5k entries [total 1million req.]

• Comparison:
  • Use VMs: 2 EC2 instance [m5.large type] with 2CPUs and 8GB mem.
  • Use lambdas: 256MB mem.

• **Cost [per month]:** $37.74 $\lambda$ / $138.24$ [VMs], (c.f., run $\lambda$ 1min - **$2,162.16 / 138.24$**)
• **Scalability:** lambda is the best for the unpredictable loads, as it only runs when it is invoked.