Guarding Serverless Applications with Kalium

Deepak Sirone Jegan*, Liang Wang+, Siddhant Bhagat#, Michael Swift*

* University of Wisconsin - Madison
+ Princeton University
# Microsoft

MADS&P
Security and Privacy Research Group
at UW-Madison
Outline

• **Serverless Computing**
• Security in Serverless Computing
• Our Approach: Kalium
• Evaluation
• Conclusion
Serverless Computing

1. Upcoming application deployment model in the cloud
2. Decompose large applications into stateless functions
3. Billing per-invocation of function
1. A path is a sequence of functions executed in the application
2. Functions can be invoked from other functions or external services: Purple Arrows
3. A function can send messages to external services: Green Arrows
4. Output of last function in a path is returned to the application logic
Outline

• Serverless Computing
• Security in Serverless Computing
• Our Approach: Kalium
• Evaluation
• Conclusion
Threat Model

1. Serverless Infrastructure is secure

2. At least one serverless function has **Remote Code Execution** bugs: Data input to application logic is untrusted
Attack Scenario 1

Application Logic

Validate Transaction
Validate User
Read Balance
Compute Balance
Write Balance
Write User

Function Invocation
Message to Ext. Service

Confidential Information Should not Flow to Unintended External Locations

User      Balance
Bug              $10
Attacker Controller Server

{Bug: $10}

Data Store
Attack Scenario 2

<table>
<thead>
<tr>
<th>Function Invocation</th>
<th>Message to Ext. Service</th>
</tr>
</thead>
</table>

Data Written to External Storage Should be the Result of a Valid Path
Insight for Attack Detection

Internal complex Control Flow is now observable at the network level

Prior Work including information flow control and web application firewalls do not consider order of functions in a path
Outline

• Serverless Computing
• Security in Serverless Computing
• Our Approach: Kalium
• Evaluation
• Conclusion
Kalium - Overview

Idea: All executed paths in serverless application should be valid

Kalium: System to track paths and its validity in serverless applications

Application Profiling Stage: Build expected valid paths of each function and whole application

Enforce the valid paths: Augment the function runtime to intercept messages generated by a function

Global Controller tracks current path in whole application
Kalium – Serverless Control Flow

Define Application Control Flow Graph and Function Control Flow Graph

- Application Control Flow Graph: Graph depicting order of function invocations in application
- Function Control Flow Graph: Graph depicting order of messages sent during execution of function

<table>
<thead>
<tr>
<th>Function Invocation</th>
<th>Message to Ext. Service</th>
</tr>
</thead>
</table>

Diagram:
- User interacts with Application Logic
- 
- /deposit
- /add_user
- Validate Transaction
- Read Balance
- Compute Balance
- Write Balance
- Error Logger
- Message Queue
- Data Store
- Function Invocation
- Message to Ext. Service
Application Control Flow

Application Control Flow Graph: Graph depicting order of function invocations in application

Nodes are the functions in the application

Edges between functions are labeled with URLs of destination function

<table>
<thead>
<tr>
<th>Function Invocation</th>
<th>Message to Ext. Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purple</td>
<td>Green</td>
</tr>
</tbody>
</table>

Diagram showing function invocations and data flow including:
- API Gateway
- Message Queue
- Data Store
- Functions: Validate Transaction, Read Balance, Compute Balance, Write Balance, Write User
- Utensils: Error Logger

User actions:
- /deposit
- /add_user

Application Logic:
- Validate Transaction
- Validate User
- Read Balance
- Compute Balance
- Write Balance
- Write User
Function Control Flow

Function Control Flow Graph: Graph depicting order of messages sent during execution of function

Nodes are internal function states before sending a message

Edges between nodes are labeled with URLs of destination external services

Each function is assumed to end in exactly one application sub-path

```python
def read_balance(transaction):
    user_balance, err = getUserBalance(dataStore, transaction.user)
    if err:
        sendErr(queue, err)
    else:
        callComputeBalance(transaction, user_balance)
```
Kalium - Implementation

Intercept function messages at the network syscall level with augmented gVisor

Once a function finishes execution, it checks with the global controller whether to allow outgoing edge

A global controller maintains the position of the current function on application CFG
Outline

• Serverless Computing
• Security in Serverless Computing
• Our Approach: Kalium
• Evaluation
• Conclusion
Evaluation – Attack Scenario 1 and 2

Evaluation

- Attack Scenario 1 and 2
  - Function Invocation
  - Message to Ext. Service

Application Logic
- Validate Transaction
- Validate User
- Read Balance
- Compute Balance
- Write Balance
- Write User

Controller
- Invalid Transaction
- Error Logger
- Data Store

Message Queue
- Compute Balance
- Write Balance
- Write User

Controller Server
- User
- Balance
- Bug
- $10

Function Invocation
- /deposit
- /add_user

Invalid Transaction
- gVisor
- Controller

Message to Ext. Service
- Invalid Transaction
- gVisor
- Controller
Kalium – Performance Evaluation

Benchmarks: Valve Benchmarks

Comparison: Valve (IFC) [WWW ’20] on gVisor, Trapeze (IFC) [OOPSLA ’18] on gVisor

Geomean: 1.25, 1.40 and 2.90 across all benchmark functions/paths for Kalium, Valve and Trapeze resp.
Conclusion

• Enforcing Control Flow is important for Serverless Application Security

• We present Kalium a Control Flow Integrity framework for Serverless Apps

• Kalium has reasonable performance overhead for enforcing Control Flow Integrity

https://github.com/multifacet/kalium_artifact

Questions?

Deepak Sirone: dsirone@cs.wisc.edu