CYBERSECURITY CONVERGENCE LAB

kips 한국정보처리학회 Korea Information Processing Society

INCHEON NATIONAL UNIVERSITY

Bambda:
A Framework for Preventing Evasion Attacks in Serverless Environments

Changhee Shin

Department of Computer Science & Engineering

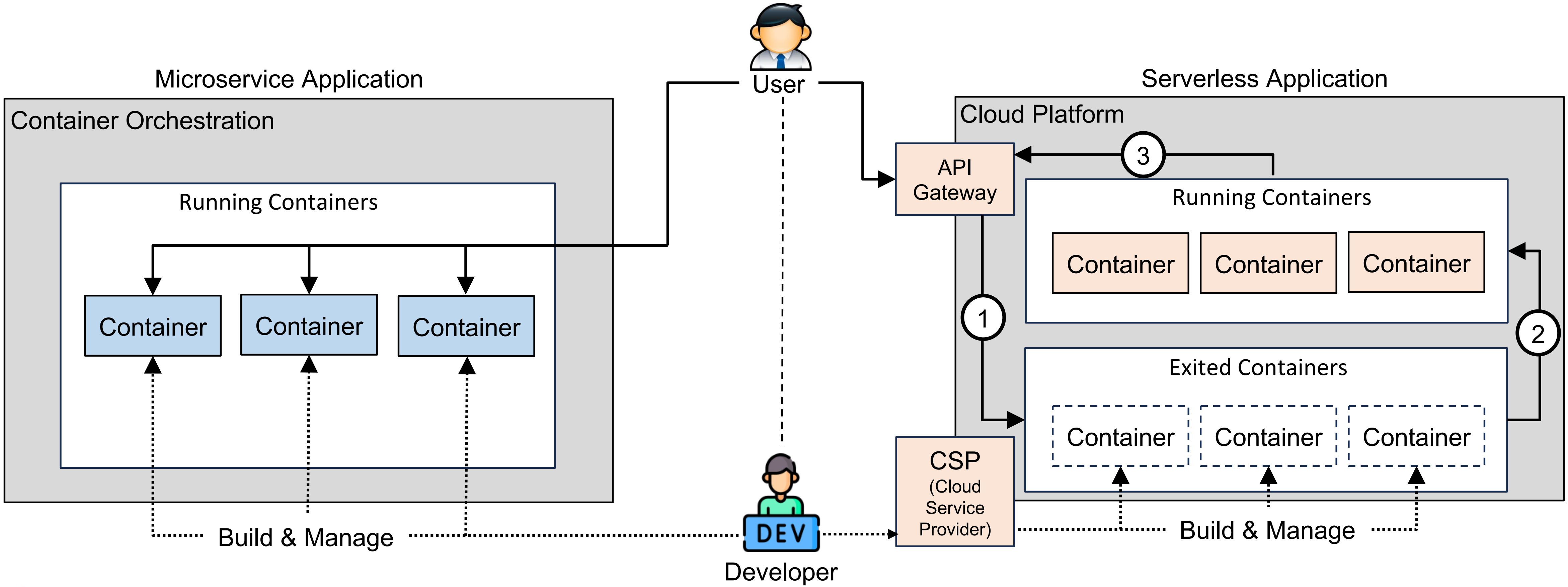
Incheon National University

Index

- Background
- Problem Statements
- Bambda Design
- Evaluation
- Conclusion

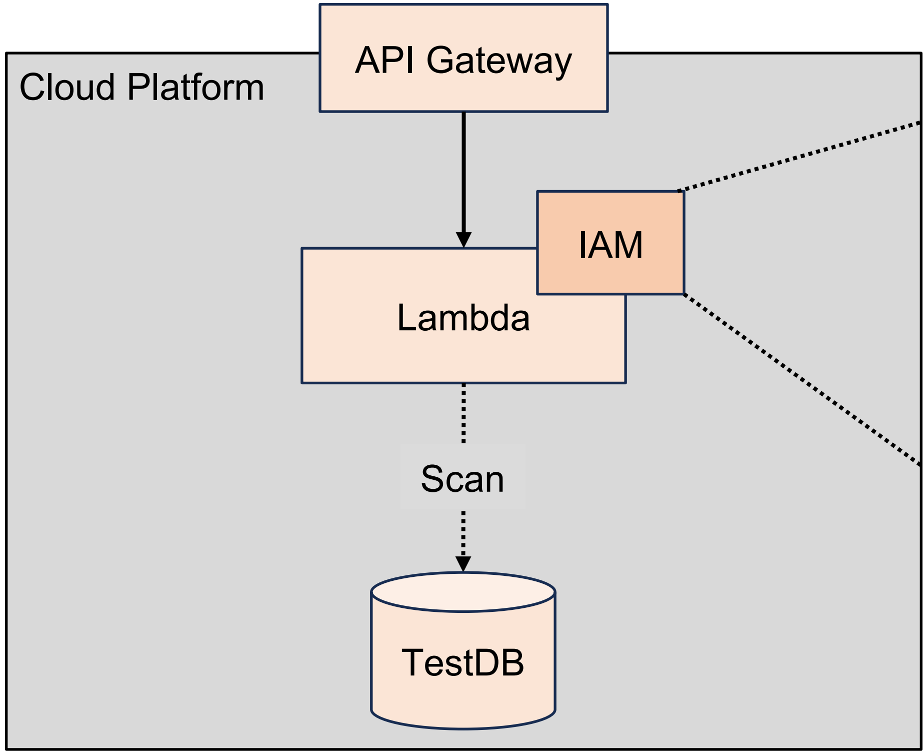
Background : Serverless Computing

- With Microservices, deploy and manage containers
- With Serverless, deploy applications without building and managing containers



Background : IAM(Identity and Access Management)

- Access control policy to resources and services provided by the cloud platform

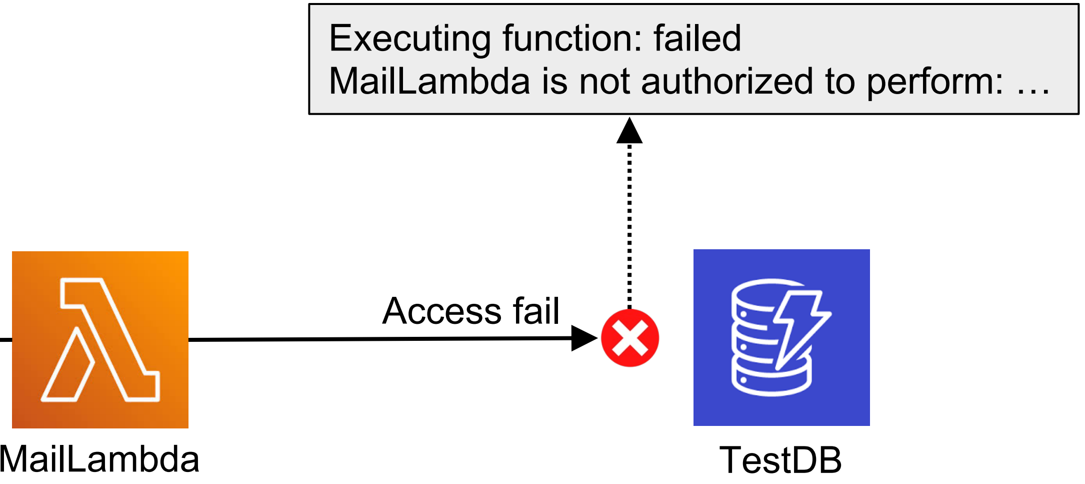


```
provider:
    name: aws
    runtime: nodejs20.x
    region: ap-northeast-2
    iam:
        role:
          statements:
            - Effect: Allow
              Action:
                  - dynamodb: Scan
              Resource: "…/TestDB"
```

Problem Statements

- MailLambda doesn’t have authority to access TestDB

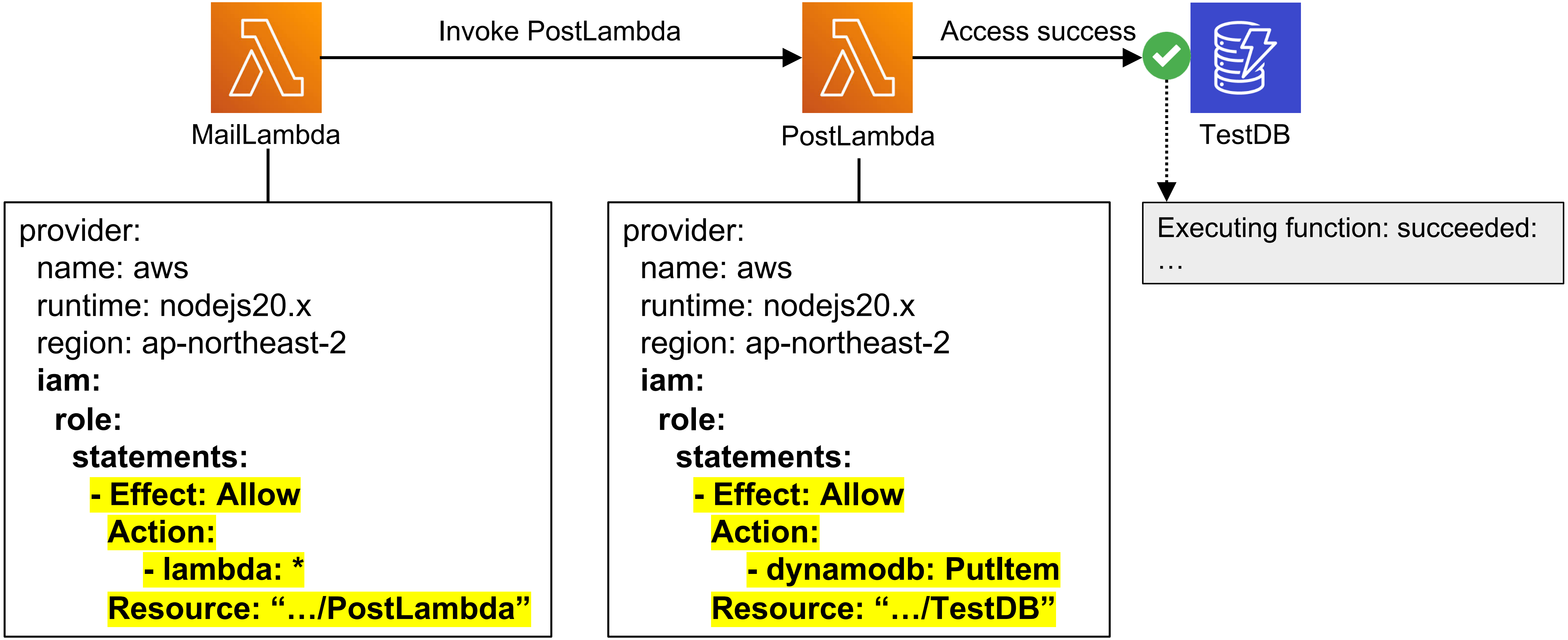
```
provider:
  name: aws
  runtime: nodejs20.x
  region: ap-northeast-2
  iam:
    role:
      statements:
        - Effect: Allow
          Action:
              - lambda: *
          Resource: “…/PostLambda”
        - Effect: Allow
          …
```



Executing function: failed
MailLambda is not authorized to perform: …

Problem Statements

- But, MailLambda has authority to invoke PostLambda
  - MailLambda can access TestDB using IAM vulnerability

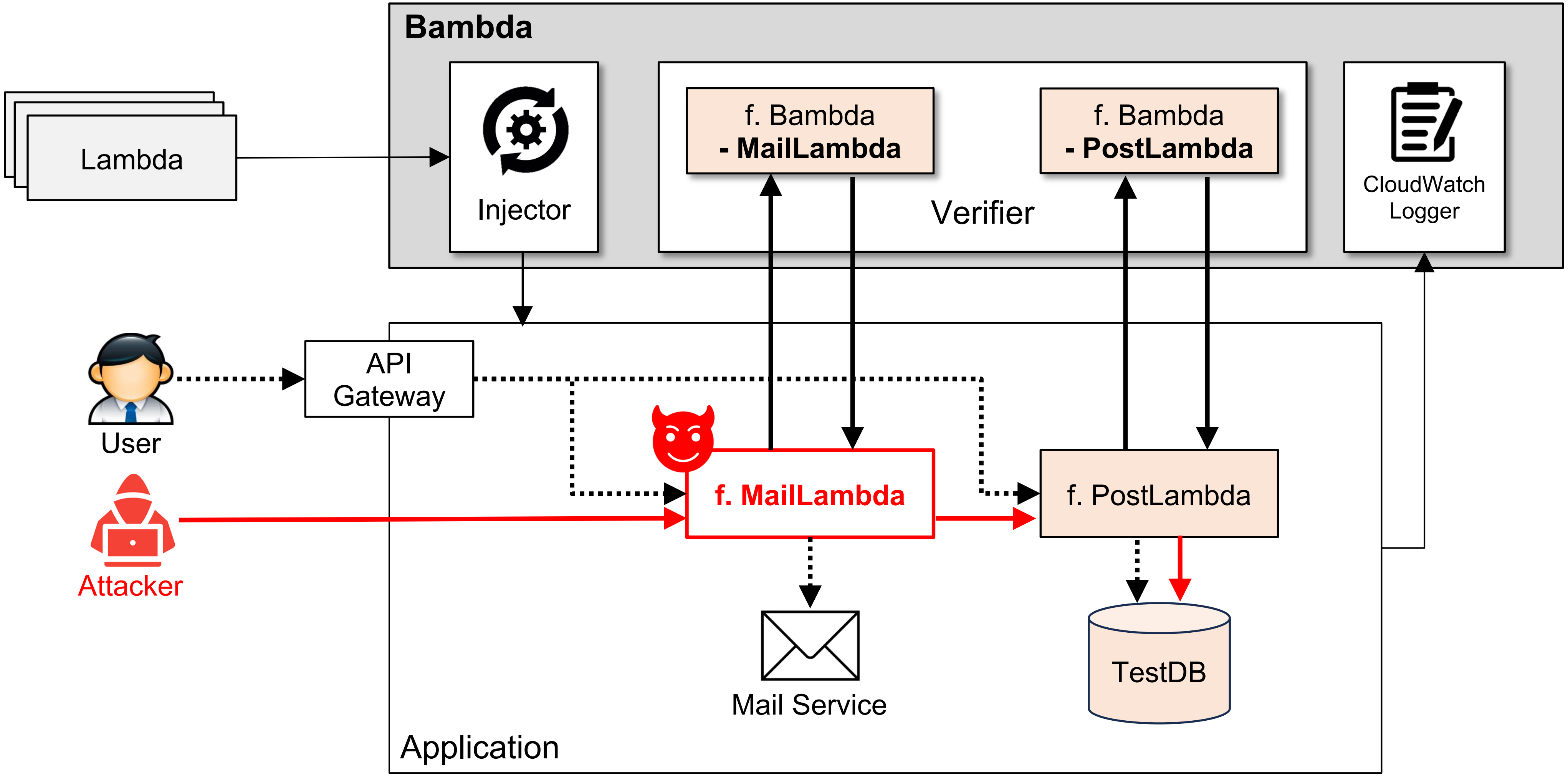


```
provider:
  name: aws
  runtime: nodejs20.x
  region: ap-northeast-2
  iam:
    role:
      statements:
        - Effect: Allow
          Action:
              - lambda: *
          Resource: "…/PostLambda"
```

```
provider:
  name: aws
  runtime: nodejs20.x
  region: ap-northeast-2
  iam:
    role:
      statements:
        - Effect: Allow
          Action:
              - dynamodb: PutItem
          Resource: "…/TestDB"
```

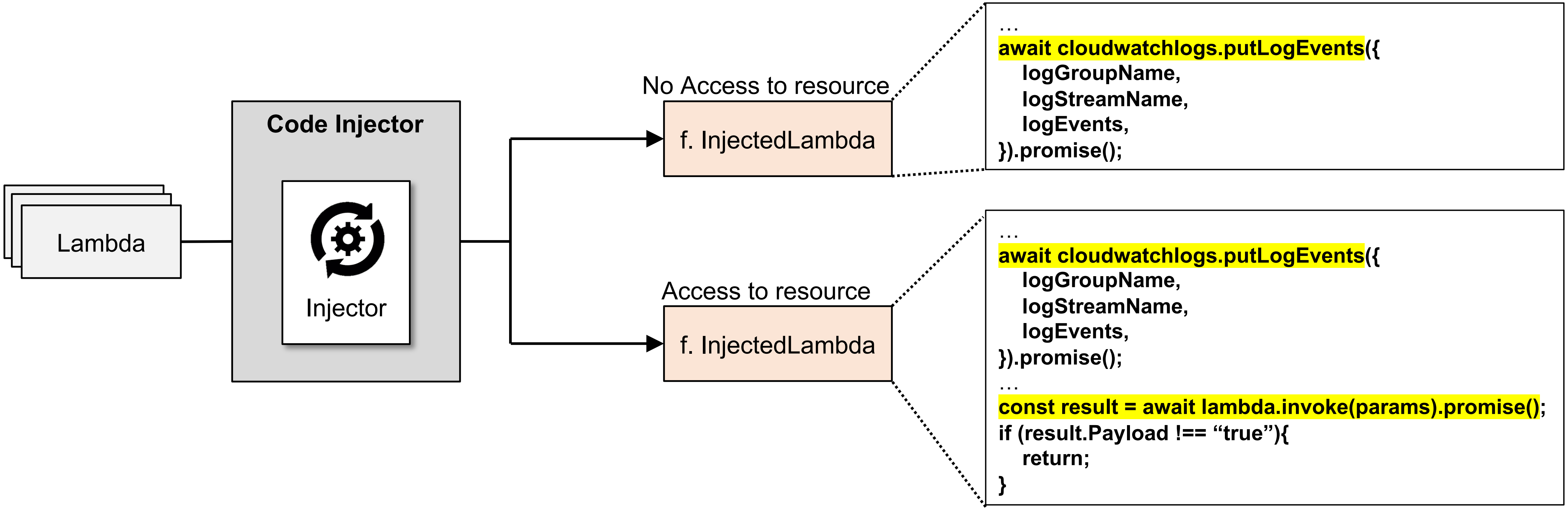
```
Executing function: succeeded:
…
```

Bambda Design : Overall Architecture



Bambda Design : Code Injector

- Inject code for real-time logging functionality
- If lambda has access to a resource, additionally inject code for verification by Bambda

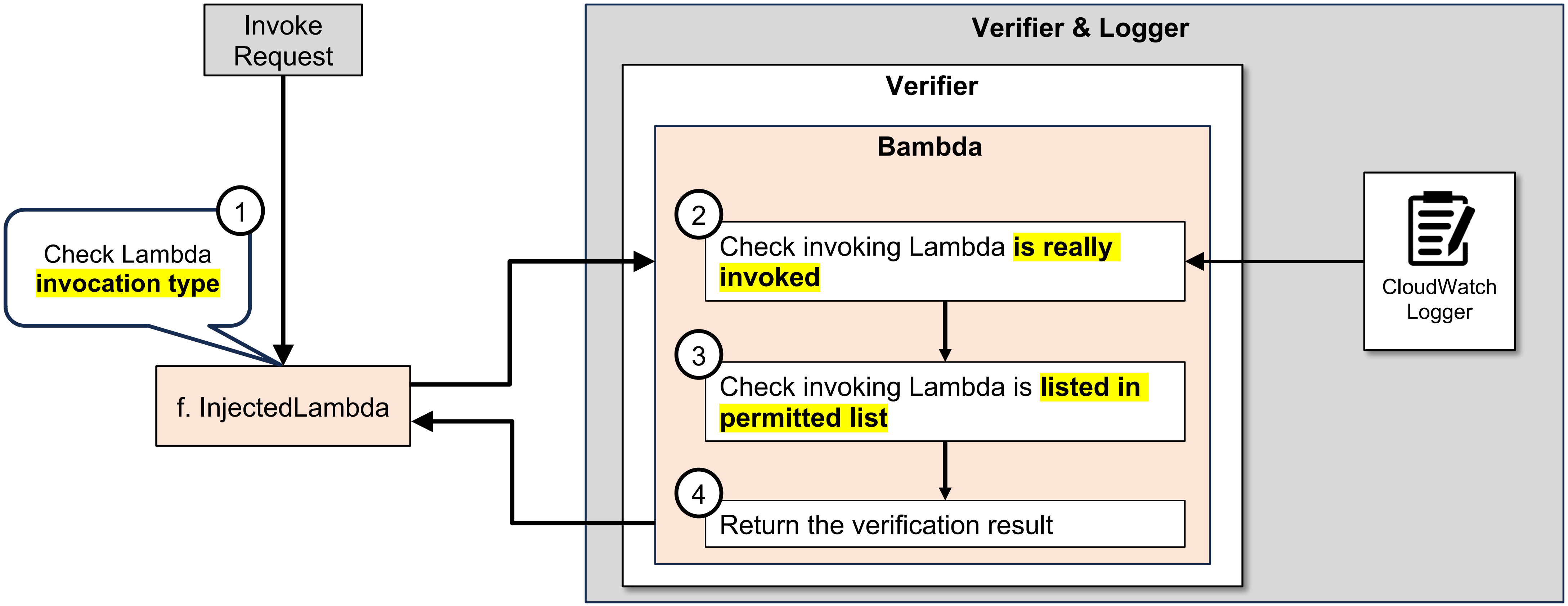


```
…
await cloudwatchlogs.putLogEvents({
    logGroupName,
    logStreamName,
    logEvents,
}).promise();
```

```
…
await cloudwatchlogs.putLogEvents({
    logGroupName,
    logStreamName,
    logEvents,
}).promise();
…
const result = await lambda.invoke(params).promise();
if (result.Payload !== “true”){
    return;
}
```

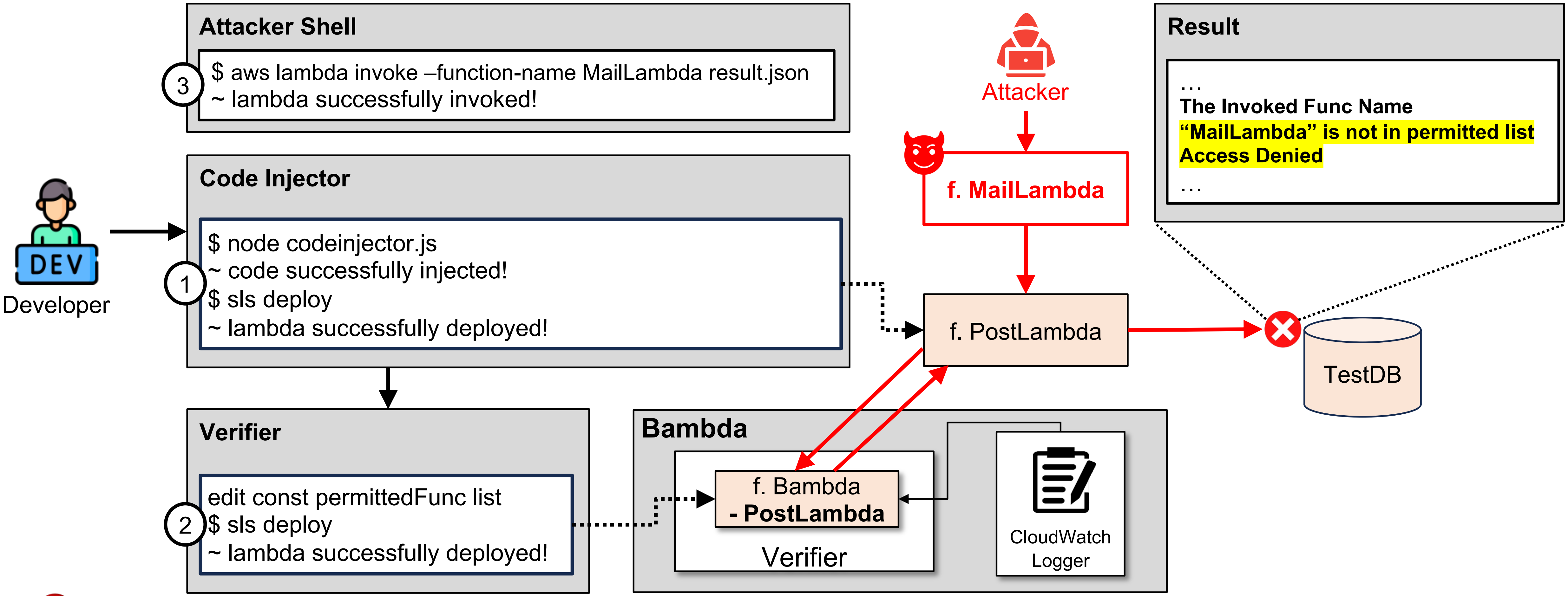
Bambda Design : Verifier

- Verify whether the function invocation request is authorized


Invoke Request
1
Check Lambda **invocation type**
f. InjectedLambda
Verifier & Logger
Verifier
Bambda
2
Check invoking Lambda **is really invoked**
3
Check invoking Lambda is **listed in permitted list**
4
Return the verification result
CloudWatch Logger

Evaluation

- Verified that Bambda successfully detects evasion attack



```
3 $ aws lambda invoke –function-name MailLambda result.json
~ lambda successfully invoked!
```

```
$ node codeinjector.js
~ code successfully injected!
1 $ sls deploy
~ lambda successfully deployed!
```

```
edit const permittedFunc list
2 $ sls deploy
~ lambda successfully deployed!
```

```
…
The Invoked Func Name
"MailLambda" is not in permitted list
Access Denied
…
```

Conclusion and Future Work

- In this paper, we proposed ***a dynamic evasion detection framework*** in a serverless environment

- Demonstrate the ability to ***prevent evasion attacks*** by using automated plugin-based Lambdas in a serverless environment

- In future work, focus on building a framework that can automatically defend ***against evasion attacks using events***