

# Hacking Serverless Applications

A Treasure Map for  
Uncharted Waters

A pair of dark, reflective sunglasses is shown against a green background. The sunglasses are positioned diagonally, with the left lens in the foreground and the right lens behind it.

**Matteo Rosi** | Cloud Native Security Researcher  
[matteo.rosi@contrastsecurity.com](mailto:matteo.rosi@contrastsecurity.com)

# Whoami

 ottimo

 @ottimorosi

 matteorosi

 ottimo.me

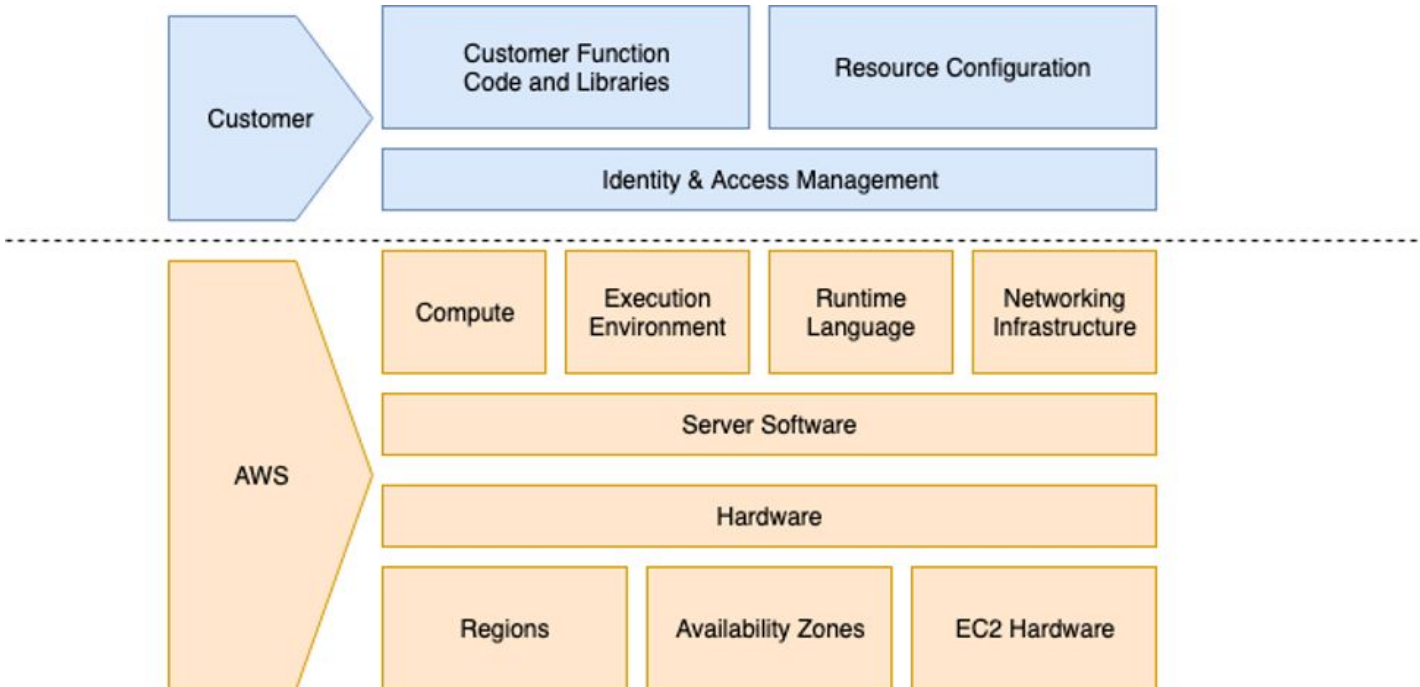
 matteo.rosi@contrastsecurity.com



Security Researcher @ Contrast Security

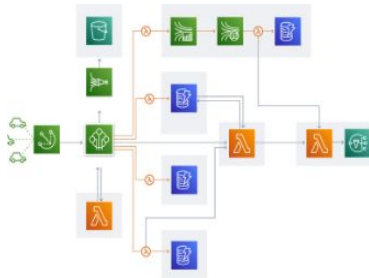
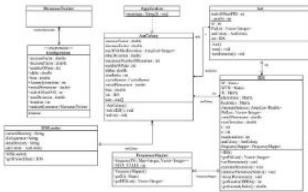
Head of Cyber Defense & CISO Deputy @ Telepass spa  
SOC Manager @ 7Layers srl

# AWS Serverless Shared Responsibility Model

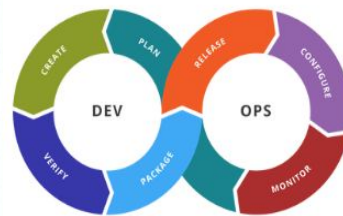
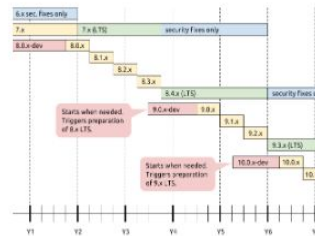


# Serverless for DevSecOps

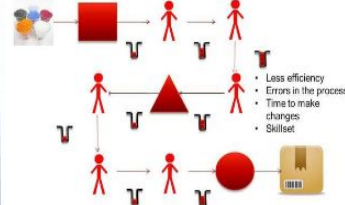
## Architecture



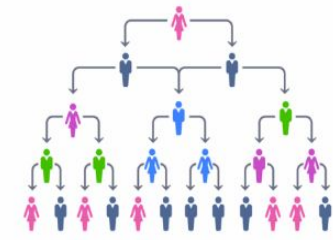
## Cycles



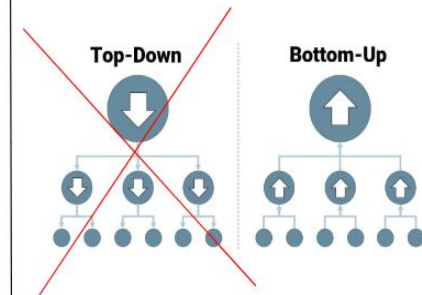
## Process



## Decision

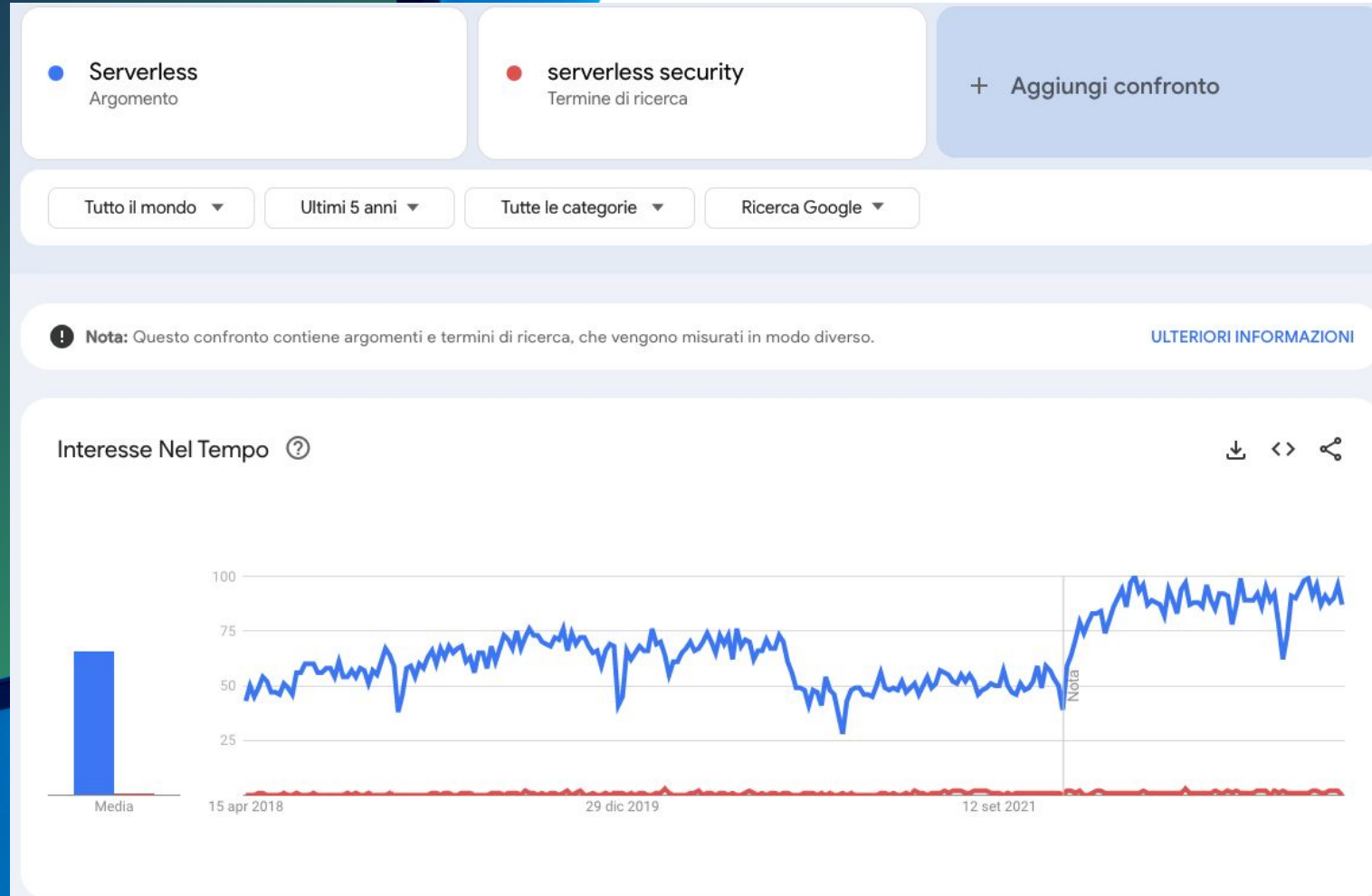


© Jacob Morgan ([thefutureorganization.com](http://thefutureorganization.com/))



# Serverless

## Trends



# Serverless Architecture

CONTRAST

Applications

Servers

Libraries

Vulnerabilities

Serverless

000000000123

Provider: aws

Region: us-west-1

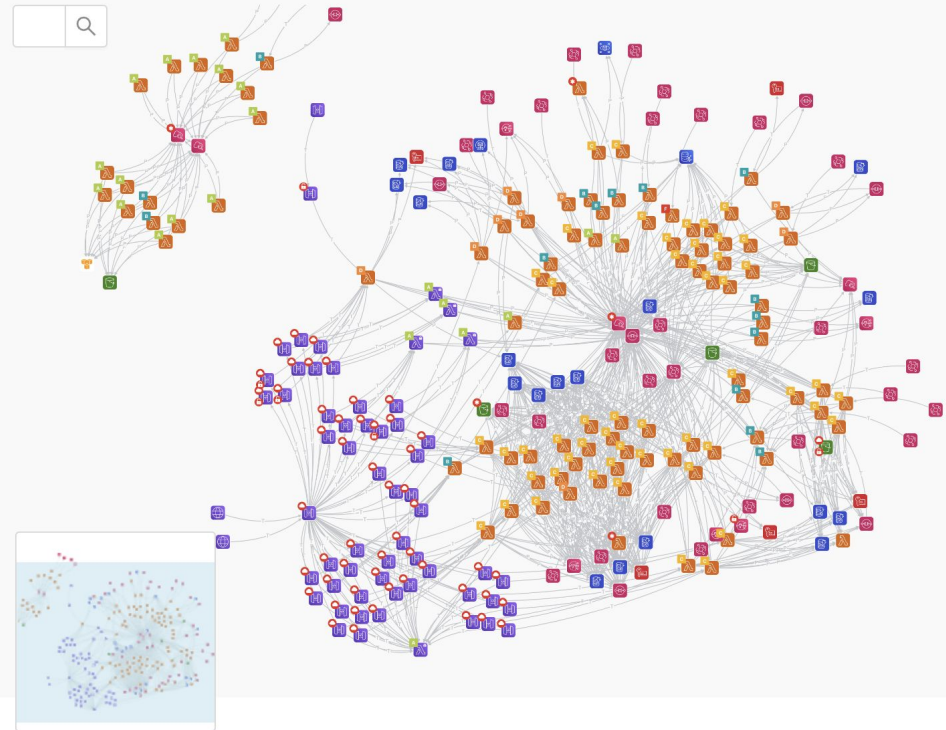
Functions

Scans

Vulnerabilities

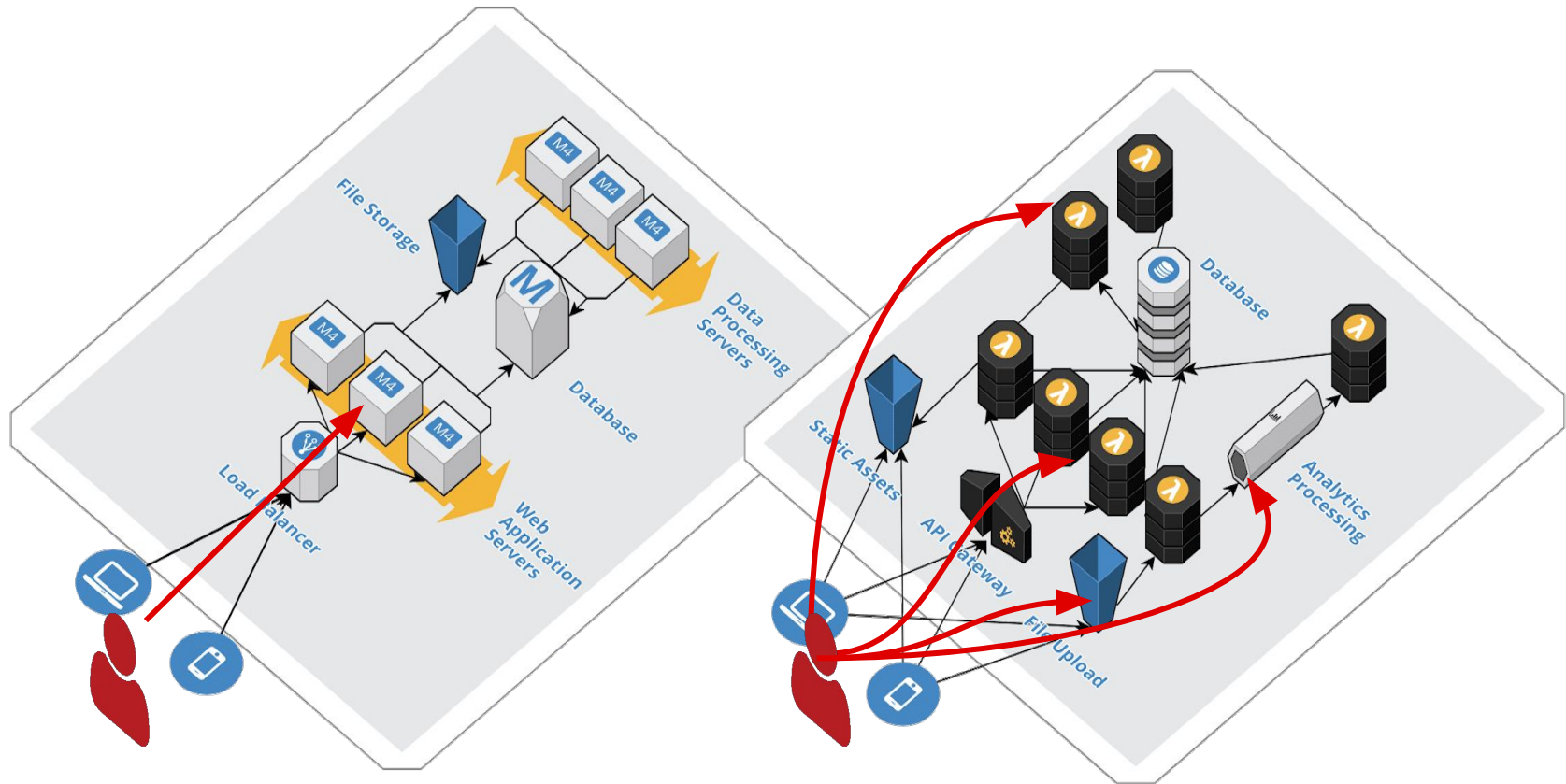
Graph

Settings

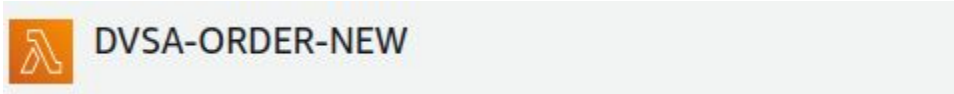




# Loss of Perimeter



# Resource-Based IAM



```
def lambda_handler(event, context):
    orderId = str(uuid.uuid4())
    itemList = event["items"]
    status = 100

    userId = event["user"]
    address = "{}"
    ts = int(time.time())
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table(os.environ["ORDERS_T"])
    response = table.put_item(
        Item={
    })

    if response['ResponseMetadata']['HTTPStatus'] == 200:
        res = {"status": "ok", "msg": "Order created"}
    else:
        res = {"status": "err", "msg": "Order not created"}

    return res
```

## Execution role

Role name

[serverlessrepo-DVSA-OrderNewFunctionRole-N65M2RQ1B6QS](#)

## Resource summary



Amazon DynamoDB  
1 action, 2 resources

To view the resources and actions that your function has permission to access, choose a service.

By action

By resource

Action

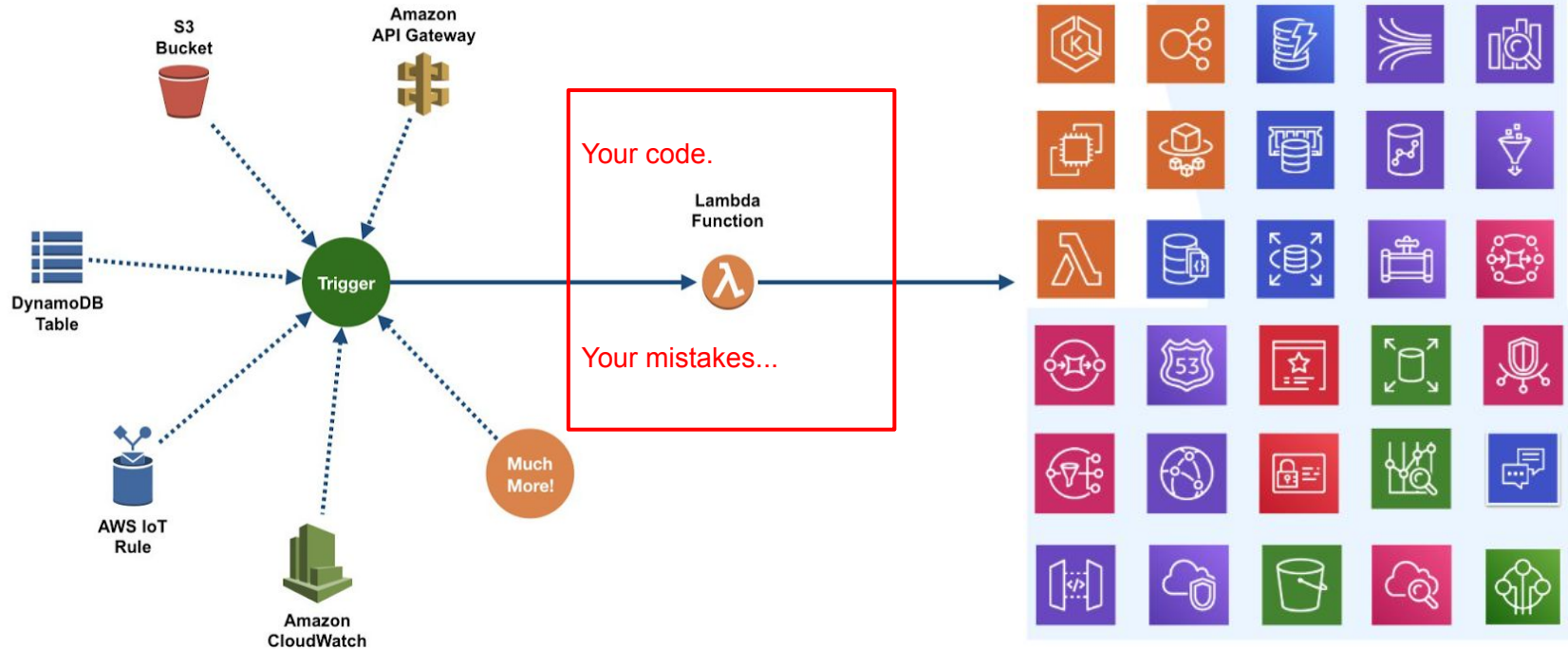
Resources

dynamodb:PutItem

Allow: arn:aws:dynamodb:us-east-1:402181209224:table/



# Event-Driven Architecture



1. Triggered via events
2. CSP spins up a container with code
3. Container terminates after code executes\*\*

# Serverless Risks

- Event Injection
- Broken Authentication
- Sensitive Data Exposure
- Over-Privileged Functions
- Vulnerable Dependencies
- Insufficient Logging & Monitoring
- Open Resources
- DoW / DoS
- Insecure Shared Space
- Insecure Secret Management



```
4ppsec@TMELAMED-C02DX3Q2ML85:~  
$ contrast lambda --function-name cn-customer-dynamic-via-s3 --profile pycon --region us-east-1  
✓ Fetching configuration and policies for Lambda Function "cn-customer-dynamic-via-s3"  
✓ Sending Lambda Function scan request to Contrast  
✓ Scan requested successfully  
✓ Scan Finished  
----- Scan completed 40.83s -----  
  
Critical | Least Privilege Violations The Attached Role arn:aws:iam::234681846983:role/cn-customer-dev-DynamicLambdaRole-NQNB9139T25Y has an overly permissive policy that may violate the principle of least privilege. For more information on AWS least privilege: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege  
Recommendation: Replace the existing policies with the following  
[  
  {  
    "PolicyName": "LambdaExecutionPolicy",  
    "PolicyDocument": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Sid": "1",  
          "Effect": "Allow",  
          "Action": [  
            "logs:CreateLogGroup",  
            "logs:CreateLogStream",  
            "logs:PutLogEvents"  
          ],  
          "Resource": [  
            "arn:aws:logs:us-east-1:234681846983:log-group:/aws/lambda/*:*:*"  
          ]  
        },  
        {  
          "Sid": "3",  
          "Effect": "Allow",  
          "Action": [  
            "s3:GetObject"  
          ],  
          "Resource": [  
            "*"   
          ]  
        }  
      ]  
    }  
  ]  
]  
  
High | Vulnerable dependency Django:2.2.27 has 2 known CVEs  
CVE-2022-28346, CVE-2022-28347
```

# CodeSec by Contrast

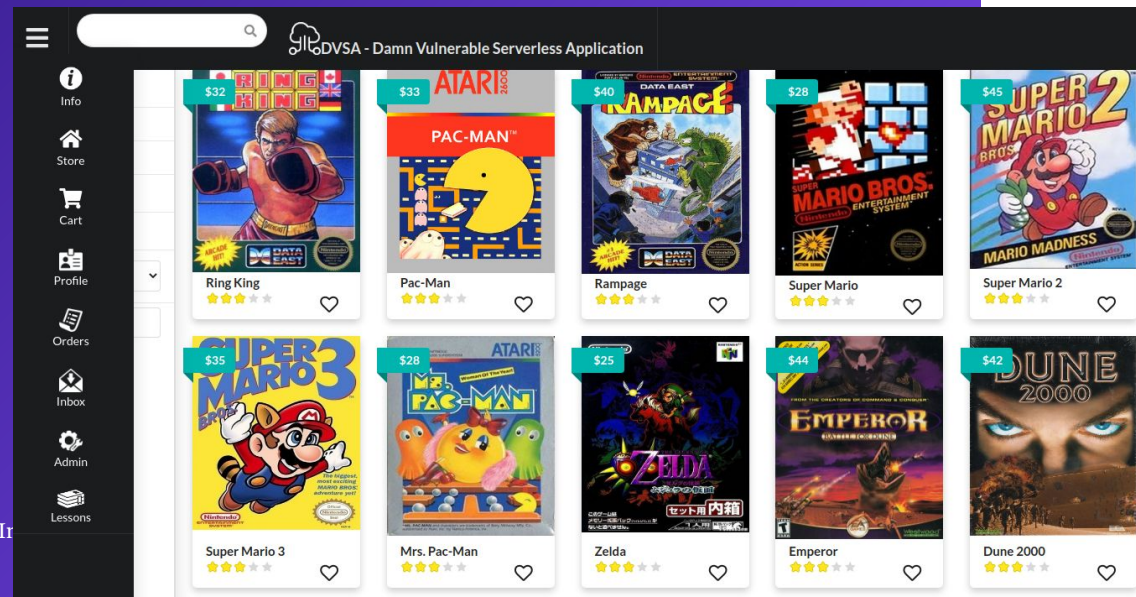
<https://www.contrastsecurity.com/developer>





<https://owasp.org/www-project-dvsa/>  
[github.com/owasp/dvsa](https://github.com/owasp/dvsa)  
Twitter: @DVSAowasp

**! NOT in PRODUCTION !**



# Security Aspects

AWS Lambda

- Read-only environment, except for /tmp
- Not wired to the internet\*
- Data is temporary\*\*
- Code reside in environment
- Keys are available as environment variables



# Demo 1

Lambda Runtime: Java

Framework involved: Spring Cloud Function

Vulnerability: CVE-2022-22963

Trigger: API Gateway



SpEL: CVE-2022-22963



# Demo 2

Lambda Runtime: Python

Library involved: PyYAML

Vulnerability: CVE-2020-14343

Trigger: S3 Bucket ObjectCreated

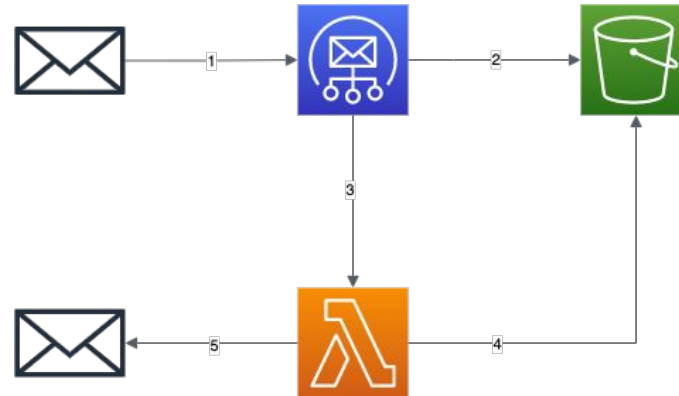
Insecure Deserialization



Lambda Runtime: Python  
Trigger: AWS SES (email)

# Demo 3

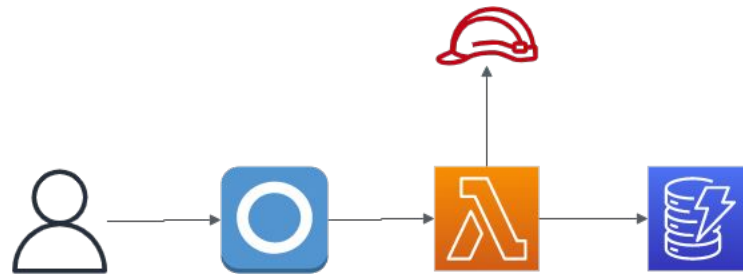
SES Injection



Lambda Runtime: Python  
Trigger: Alexa Skill

# Demo 4

Alexa Injection



# Thank you



**Hacking Serverless Applications: A Treasure Map for  
Uncharted Waters**

Company Confidential. Internal Use Only. ©2023 Contrast Security, Inc.

