locate $USER

nu11p0inter

_nu11p0inter

talmelamed

tal.melamed@qu.edu

AppSec.IT

tal@protego.io
Agenda

Base Camp
Top Risks
What's next?
The Evolution of the Cloud
Customers going serverless
Serverless Basics

- **Serverless Infrastructure**: Scales to zero, Don’t pay for idle
- **Serverless Architecture**: Single purpose micro-services, Stateless / ephemeral
- **Serverless Operations**: Minimal DevOps, Rapid feature velocity
Safety First!
Single Purpose Container

- Container spins up when required
- Triggered via events
- Read-only environment, except for /tmp
- Terminates when code execution
- Not wired to the internet*
- Data is temporary**
- Code reside in environment
- Keys are available as environment variables
Current project state:
  - Interpretation of Top 10
  - Open Data Call: http://tiny.cc/slscall

Goal: Serverless-tailored Top 10
Event Injection

- Multiple, uncontrolled entry points
- Traditional injections (cmdi, no/sqli, etc)
- Per-language Code Injection
- New Injections (MQTT, Email, Pub/Sub)
- Depends on the vulnerable function permissions
Demo
Injection Entry Points

- REST APIs
- 3rd-Party Applications
- Cloud Storage
- Authentication Services
- Logs and Events
- IoT
- Email
- SNS
- Code Injection Entry Points
Best Practices

● Never trust, pass or make any assumptions regarding input and its validity from any resource

● Use positive or “whitelist” input validation when possible
  ○ Api Gateway allow configuring json model for requests

● Consider all event types and entry points into the system

● Run functions with the least privileges required to perform the task to reduce attack surface

● Use a commercial runtime defense solution to protect functions on execution time
Opportunity

Unauthorized Service Call | cognito-idp.adminUpdateUserAttributes
Critical | Ι Blocked | Thursday, September 19th 2019, 7:27:13

What is this?
Protego has detected an attempt to call an unauthorized service api, which is not part of the function's approved access list.

What could it be?
Someone tried to use the function to access other cloud resources in the account. A successful attack could result in sensitive data leakage, modification of data or even a complete cloud resource takeover, depending on the functions' permissions.

What was detected?
Service: cognito-idp
Action: adminUpdateUserAttributes
Broken Authentication

- Functions are Stateless
- Multiple entry points, services, events and triggers
- No continuous flow
Internal function, exposed to attacker via SES
Best Practices

● Use cognito whenever possible

● Access tokens can include custom data that will be signed by Cognito

● Perform input validation and run with “Least Privileges”

● If necessary, store a “state” OOB
Sensitive Data Exposure

- Same as any other cloud-based data
- Common serverless scenarios:
  - Data under /tmp
  - Sensitive data in environment variable
  - Sensitive data in an open bucket
  - Source code is also in the environment
Stealing function keys

```
lambda@aws:~ env
AWS LAMBDA FUNCTION VERSION=$LATEST
AWS SESSION TOKEN=FQoGZXIvYXdzEAYaDB369Izam15zE1TKJCLqAdogoBF+p5olZmmlx5WSAYD9wV4bUuyMEz3jnft/thpZ2j0NYJ81KgLQyUytg3pPS7k0wdo6t
nBMLGbhNLvUkJ9oW90ecg/yTjdUJtCCAt3J4sLVnAhr+dQe4FmLjdeP2Fjxtf8cJyF600Xb/Hn1M9XLYrZ3GwAyTQrO60cb92JvElghSl9GN4D9am+LnfSza9aQ7
SGSS5Xn4rZjylygBtJss5IL6g2YWhr+rx9/1V0TV6VojF50e5LuqB360iuimWVQPMWvezhYqKbu2L10S85Dp/Rb+b42L/xB1CHAAksj+dJ0E+6C10m02fBQ==
AWS LAMBDA LOG GROUP NAME=aws/lambda/get-lambda-passwd
LAMBDA_TASK_ROOT=/var/task
LD_LIBRARY_PATH=/lib64:/usr/lib64:/var/runtime:/var/runtime/lib:/var/task:/var/task/lib:/opt/lib
AWS LAMBDA LOG_STREAM_NAME=2018/11/25/[LATEST]943526074b0e4e52a2285a136ed7ie3
AWS EXECUTION ENV=AWS_Lambda_python2.7
AWS_XRAY_DAEMON_ADDRESS=169.254.79.2:2000
AWS LAMBDA FUNCTION NAME=get-lambda-passwd
PATH=/usr/local/bin:/bin:/usr/bin:/opt/bin
AWS DEFAULT_REGION=us-east-1
PWD=/var/task
AWS SECRET ACCESS KEY=BVx8n3NKTMOYyjwip2fvyV0ALPeYjh9gyz46xW5s
LAMBDA_RUNTIME_DIR=/var/runtime
LANG=en_US.UTF-8
AWS REGION=us-east-1
third_party_api_key=A0JCAJh0d5uFyQjilLCZAc7aPM0a7QzatypYc0t7rLGb9u0sv3WfM7t4cC+XJCNn9IP1a0dNUAAAaDWoBgbkqhki9w0BBwagWzBZAgEAMF
QQCSq55s13BCEHATaeBgIghk8ZQMEAS4wEQQrMKZUEpjv0PWlOAAAqE0gGcCteG0jB09qWnl4RpWsfB3Qj6Ca0jT7wLQAkFeJvJhR6i3+y2Qc=
TZ=UTC
AWS ACCESS_KEY_ID=ASIAY03RCHMAPBPJAMF0
SHELVL=1
.AWS/XRAY DAEMON ADDRESS=169.254.79.2
.AWS/XRAY DAEMON PORT=2000
PYTHONPATH=/var/runtime
.X AMZN TRACE ID=Root=1-5bf1412c-5836f8d93f798321d37a68A;Parent=51898c3d4bf32405;Sampled=0
AWS SECURITY TRACE KEY=Fq0GZXIvYXdzEAYaDB369Izam15zE1TKJCLqAdogoBFXp5olZmmlx5WSAYD9wV4bUuyMEz3jnft/thpZ2j0NYJ81KgLQyUytg3pPS7k0wdo6t
nBMLGbhNLvUkJ9oW90ecg/yTjdUJtCCAt3J4sLVnAhr+dQe4FmLjdeP2Fjxtf8cJyF600Xb/Hn1M9XLYrZ3GwAyTQrO60cb92JvElghSl9GN4D9am+LnfSza9aQ7
SGSS5Xn4rZjylygBtJss5IL6g2YWhr+rx9/1V0TV6VojF50e5LuqB360iuimWVQPMWvezhYqKbu2L10S85Dp/Rb+b42L/xB1CHAAksj+dJ0E+6C10m02fBQ==
AWS_XRAY_CONTEXT_MISSING=LOG_ERROR
HANDLER=lambda function.lambda_handler
AWS LAMBDA FUNCTION MEMORY_SIZE=256
_.=../bin/env
```
Demo
Best Practices

- Whenever possible, delete /tmp after use
- Use KMS to encrypt environment variable
- Secure your Buckets configuration
- Run as “Least Privilege” to reduce access to sensitive data
- Use AWS Macie to identify sensitive data
Over-Privileged Functions

- Over privileged functions
- More than 90% are mis-configured
- Impact of other vulnerabilities depends on the permission given to the function
  - In extreme cases - full cloud account takeover
```javascript
var s3 = new AWS.S3({apiVersion: '2006-03-01'});
var params = {Bucket: 'myBucket', Key: imageFileName};
var file = require('fs').createWriteStream('/tmp/file.jpg');
s3.getObject(params).createReadStream().pipe(file);
```
Best Practices

- Review each resource and apply least privileges
- Automate!
Opportunity

Permissive Role

Critical | 16 days ago

What is this?
This lambda function is using a role which has redundant permissions, which are not required by the function.
Setting unnecessary permissions increases the function attack surface, which can be leveraged by attackers to leak sensitive data and could even lead into complete resource takeover.

Why was this detected?
The function has the following redundant permissions:

cognito-idp:
  cognito-idp:*

lambda:
  lambda:*

logs:
  logs:*

sts:
  sts:*

Version: '2012-10-17'
Statement:
- Sid: ProtegoGenerated07b335de
  Effect: Allow
  Action:
    - 'cognito-idp:AdminGetUser'
    - 'logs:CreateLogGroup'
  Resource:
    - '*'

- Sid: ProtegoGenerated19ee761
  Effect: Allow
  Action:
    - 'lambda:InvokeFunction'
  Resource:
    - 'arn:aws:lambda::067972962417:function:'

- Sid: ProtegoGenerated85bb4c1
  Effect: Allow
  Action:
    - 'logs:CreateLogStream'
    - 'logs:PutLogEvents'
  Resource:
Vulnerable Dependencies

- Using dependencies which are insecure
- Very common
- Functions may have 100 lines of code, but they bring everything with them
Best Practices

- Scan your dependencies before deploying into production
  - Open-source, 3rd-party
- Use secure versions, replace library or apply patch
Opportunity

Vulnerable Dependency
High | 11 hours ago

What is this?
This lambda function uses a library that has known vulnerabilities.

Why was this detected?
Following is a list of libraries used by this function along with their known vulnerabilities:

* node-serialize (version: 0.0.4):
  - An issue was discovered in the node-serialize package 0.0.4 for Node.js. Untrusted data passed into the unserialized() function can be exploited to achieve arbitrary code execution by passing a JavaScript Object with an Immediately Invoked Function Expression (IIFE).

Which resources are involved?
Insufficient Logging & Monitoring

- More difficult than traditional web applications
- We don’t own the infrastructure - where to deploy?
- Logs exist, but we need to know how and what to extract.
- Even if we do:
  - with 1M invocations - how can we learn anything?
Stateless & Ephemeral
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
serverless.fail
github.com/owasp/dvsa
@DVSAowasp
Thanks!

Q & A