



CYART

[inquiry@cyart.io](mailto:inquiry@cyart.io)

[www.cyart.io](http://www.cyart.io)

---

## **Cloud Privilege Abuse Simulation Lab**



## Table of contents

1. Lab Objective	3
2. Tools	3
3. Environment Setup	3
4. Lab Execution	4
5. Summary	8

## List of Figures

Figure 3.1 Shows LocalStack being setup	3
Figure 4.1 Shows over-privileged role being created	4
Figure 4.2 Shows admin policy being added to over-privileged role	5
Figure 4.3 Shows role being verified	5
Figure 4.4 Shows credentials of over-privilege role	6
Figure 4.5 Shows exporting creds	6
Figure 4.6 Shows a test IAM role being created	7
Figure 4.7 Shows recon commands for certificate_transparency	7
Figure 4.8 Shows deleting users	8

## List of Tables

Table 2.1 Shows Tools used	3
----------------------------	---

## 1. Lab Objective

- Simulate privilege abuse in a cloud environment using a controlled lab setup.
- Demonstrate the risks of over-privileged IAM roles.
- Gain administrative access by exploiting misconfigured permissions in a safe environment.

## 2. Tools

<i>Tool</i>	<i>Purpose</i>
LocalStack	Simulated AWS cloud environment for safe testing
AWS CLI	Execute IAM actions and simulate privilege abuse
Bash	Environment for running commands and managing variables

Table 2.1 Shows Tools used

## 3. Environment Setup

- Start LocalStack and set up LocalStack

```

└─$ localstack start -d

LocalStack

- LocalStack CLI: 4.8.0
- Profile: default
- App: https://app.localstack.cloud

[17:17:01] starting LocalStack in Docker mode
           preparing environment
           configuring container
           starting container
[17:17:03] detaching
           localstack.py:532
           bootstrap.py:1315
           bootstrap.py:1324
           bootstrap.py:1334
           bootstrap.py:1338

(venv)-(kali@vbox)-[~]
└─$ aws configure
AWS Access Key ID [*****access]: fakeaccess
AWS Secret Access Key [*****ress]: fakeaccess
Default region name [us-east-1]: us-east-1
Default output format [json]: json

(venv)-(kali@vbox)-[~]
└─$ export AWS_ENDPOINT_URL=http://localhost:4566

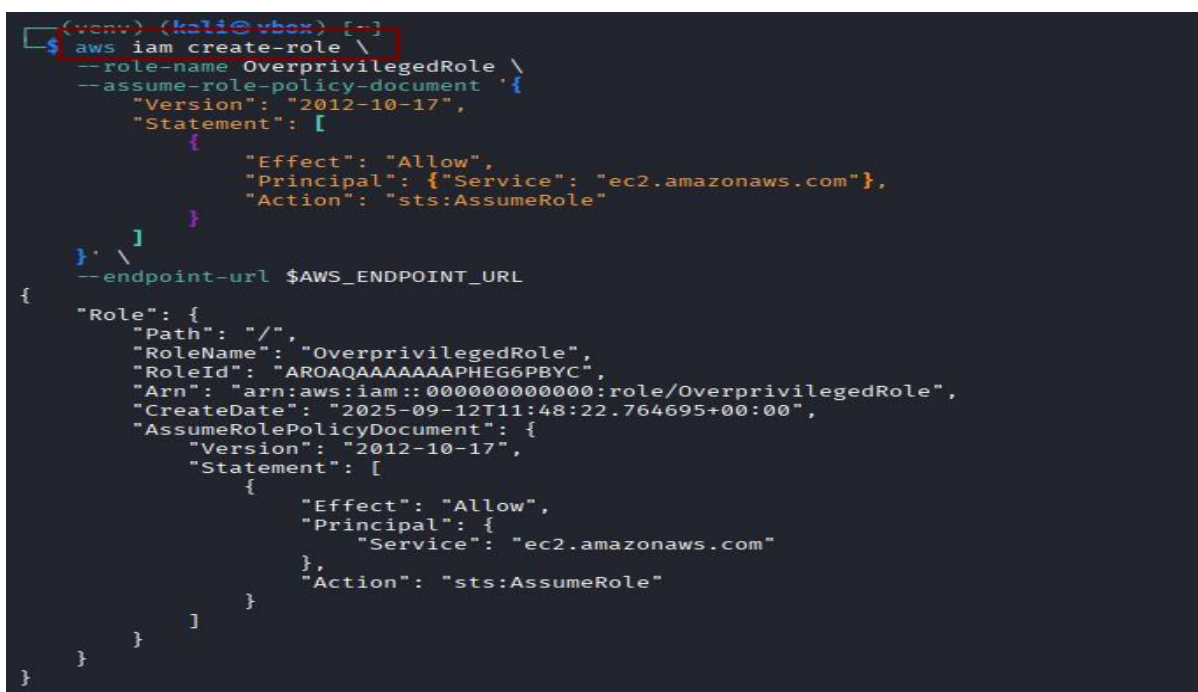
```

Figure 3.1 Shows LocalStack being setup

## 4. Lab Execution

*Step 1: Create an Overprivileged Role*

```
aws iam create-role \  
    --role-name OverprivilegedRole \  
    --assume-role-policy-document '{  
        "Version": "2012-10-17",  
        "Statement": [  
            {"Effect": "Allow", "Principal": {"Service":  
"ec2.amazonaws.com"}, "Action": "sts:AssumeRole"}  
        ]  
    }'  
    --endpoint-url $AWS_ENDPOINT_URL
```

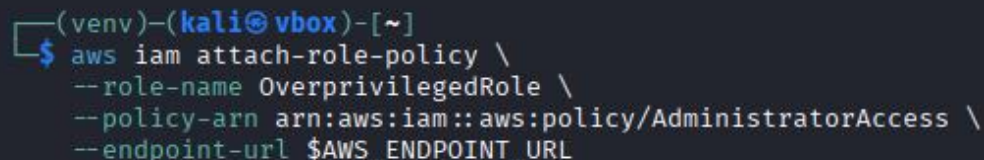


```
(venv) (kali@vbox) [~]  
$ aws iam create-role \  
  --role-name OverprivilegedRole \  
  --assume-role-policy-document '{  
    "Version": "2012-10-17",  
    "Statement": [  
      {  
        "Effect": "Allow",  
        "Principal": {"Service": "ec2.amazonaws.com"},  
        "Action": "sts:AssumeRole"  
      }  
    ]  
  }' \  
  --endpoint-url $AWS_ENDPOINT_URL  
{  
  "Role": {  
    "Path": "/",  
    "RoleName": "OverprivilegedRole",  
    "RoleId": "AROQAQAAAAAAPHG6PBYC",  
    "Arn": "arn:aws:iam::000000000000:role/OverprivilegedRole",  
    "CreateDate": "2025-09-12T11:48:22.764695+00:00",  
    "AssumeRolePolicyDocument": {  
      "Version": "2012-10-17",  
      "Statement": [  
        {  
          "Effect": "Allow",  
          "Principal": {  
            "Service": "ec2.amazonaws.com"  
          },  
          "Action": "sts:AssumeRole"  
        }  
      ]  
    }  
  }  
}
```

*Figure 4.1 Shows over-privileged role being created*

**Step 2:** Attach Admin Policy to it

```
aws iam attach-role-policy \  
--role-name OverprivilegedRole \  
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \  
--endpoint-url $AWS_ENDPOINT_URL
```

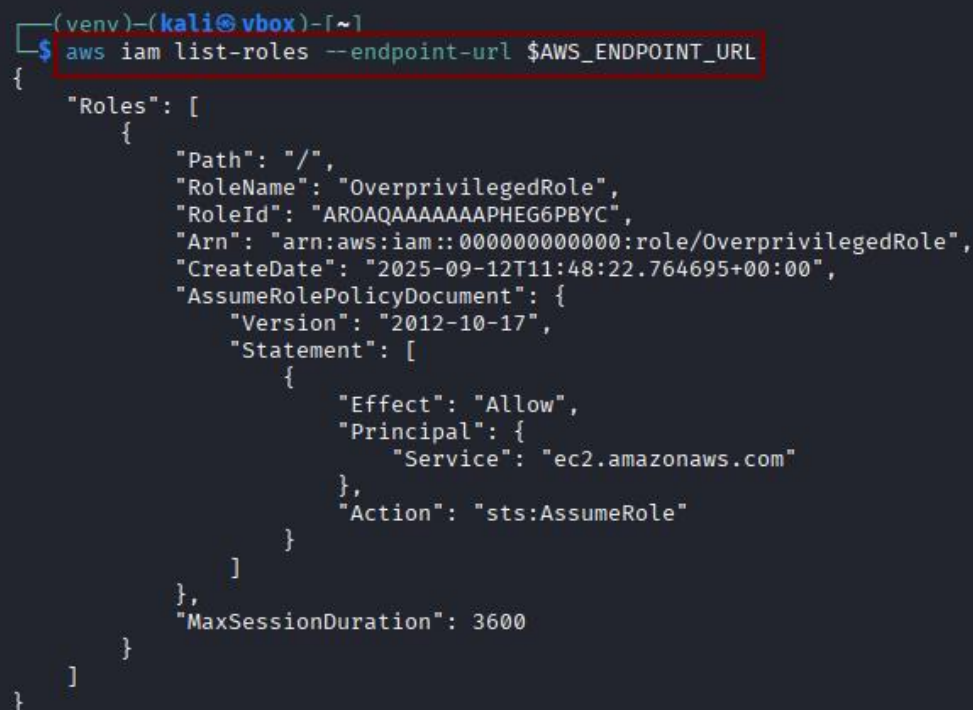


```
(venv)-(kali@vbox)-[~]  
$ aws iam attach-role-policy \  
--role-name OverprivilegedRole \  
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \  
--endpoint-url $AWS_ENDPOINT_URL
```

*Figure 4.2 Shows admin policy being added to over-privileged role*

**Step 3:** Verify Role

```
aws iam list-roles --endpoint-url $AWS_ENDPOINT_URL
```

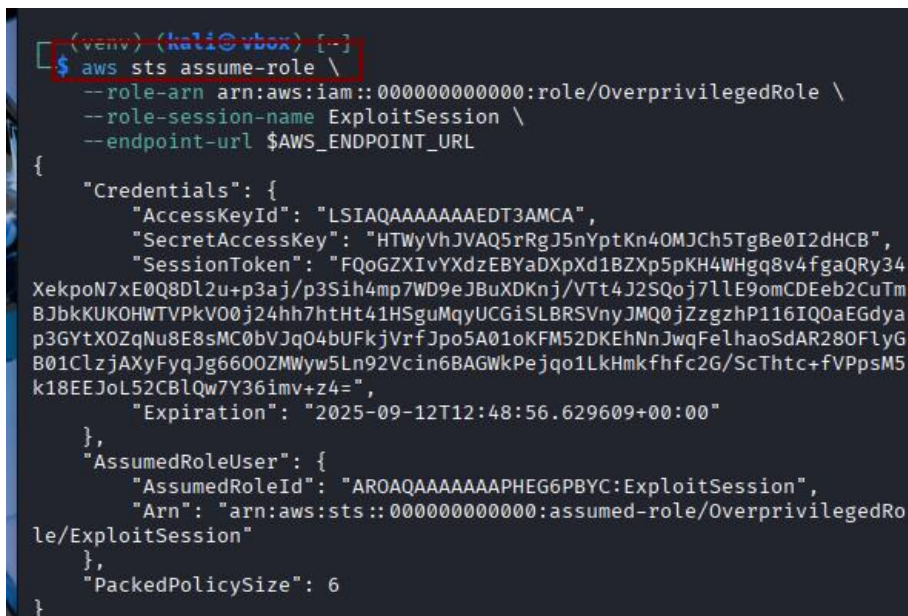


```
(venv)-(kali@vbox)-[~]  
$ aws iam list-roles --endpoint-url $AWS_ENDPOINT_URL  
{  
  "Roles": [  
    {  
      "Path": "/",  
      "RoleName": "OverprivilegedRole",  
      "RoleId": "AROQAQAAAAAAPHG6PBYC",  
      "Arn": "arn:aws:iam::000000000000:role/OverprivilegedRole",  
      "CreateDate": "2025-09-12T11:48:22.764695+00:00",  
      "AssumeRolePolicyDocument": {  
        "Version": "2012-10-17",  
        "Statement": [  
          {  
            "Effect": "Allow",  
            "Principal": {  
              "Service": "ec2.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
          }  
        ]  
      },  
      "MaxSessionDuration": 3600  
    }  
  ]  
}
```

*Figure 4.3 Shows role being verified*

### Step 3: Assume Overprivileged Role

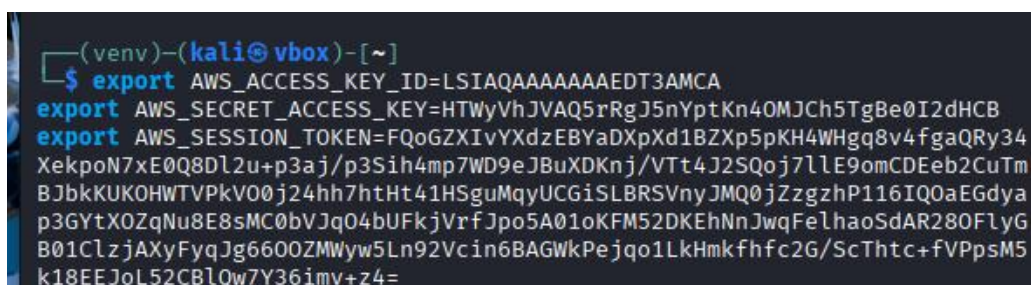
```
aws sts assume-role \
    --role-arn arn:aws:iam::000000000000:role/OverprivilegedRole \
    --role-session-name ExploitSession \
    --endpoint-url $AWS_ENDPOINT_URL
```



```
(venv) (kali@vbox) [~]
$ aws sts assume-role \
    --role-arn arn:aws:iam::000000000000:role/OverprivilegedRole \
    --role-session-name ExploitSession \
    --endpoint-url $AWS_ENDPOINT_URL
{
  "Credentials": {
    "AccessKeyId": "LSIAQAAAAAAEDT3AMCA",
    "SecretAccessKey": "HTWyVhJVAQ5rRgJ5nYptKn40MJCh5TgBe0I2dHCB",
    "SessionToken": "FQoGZXIvYXZlEBYadXpXd1BZXp5pKH4WHgq8v4fgaQRy34XekpoN7xE0Q8Dl2u+p3aj/p3Sih4mp7WD9eJBuXDKnj/VTt4J2SQoj7lLE9omCDEeb2CuTmBJbkKUKOHWTVPkV00j24hh7htHt41HSguMqyUCGiSLBRsvnyJMQ0jZzgzhP116IQ0aEGdya p3GYtX0ZqNu8E8sMC0bVJq04bUFkjVrfJpo5A01oKFM52DKEhNnJwqFelhaoSdAR280FlyGB01ClzjAXyFyqJg6600ZMwyw5Ln92Vcin6BAGWkPejqo1LkHmkfhfc2G/ScThtc+fVPpsM5k18EEJoL52CBlQw7Y36imv+z4=",
    "Expiration": "2025-09-12T12:48:56.629609+00:00"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AROQAQAAAAAAPHEG6PBYC:ExploitSession",
    "Arn": "arn:aws:sts::000000000000:assumed-role/OverprivilegedRole/ExploitSession"
  },
  "PackedPolicySize": 6
}
```

Figure 4.4 Shows credentials of over-privilege role

### Step 4: Export the Temporary Credentials from above step



```
(venv) (kali@vbox) [~]
$ export AWS_ACCESS_KEY_ID=LSIAQAAAAAAEDT3AMCA
export AWS_SECRET_ACCESS_KEY=HTWyVhJVAQ5rRgJ5nYptKn40MJCh5TgBe0I2dHCB
export AWS_SESSION_TOKEN=FQoGZXIvYXZlEBYadXpXd1BZXp5pKH4WHgq8v4fgaQRy34XekpoN7xE0Q8Dl2u+p3aj/p3Sih4mp7WD9eJBuXDKnj/VTt4J2SQoj7lLE9omCDEeb2CuTmBJbkKUKOHWTVPkV00j24hh7htHt41HSguMqyUCGiSLBRsvnyJMQ0jZzgzhP116IQ0aEGdya p3GYtX0ZqNu8E8sMC0bVJq04bUFkjVrfJpo5A01oKFM52DKEhNnJwqFelhaoSdAR280FlyGB01ClzjAXyFyqJg6600ZMwyw5Ln92Vcin6BAGWkPejqo1LkHmkfhfc2G/ScThtc+fVPpsM5k18EEJoL52CBlQw7Y36imv+z4=
```

Figure 4.5 Shows exporting creds



### Step 5: Test Admin Privileges

Check for IAM Users

```
aws iam list-users --endpoint-url $AWS_ENDPOINT_URL
```

Create IAM User if not present

```
aws iam create-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
```

```
(venv)-(kali@vbox)-[~]
$ aws iam list-users --endpoint-url $AWS_ENDPOINT_URL
{
  "Users": []
}

(venv)-(kali@vbox)-[~]
$ aws iam create-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
{
  "User": {
    "Path": "/",
    "UserName": "TestUser",
    "UserId": "xe4uv4oyn4hi6ez5ptbv",
    "Arn": "arn:aws:iam::000000000000:user/TestUser",
    "CreateDate": "2025-09-12T11:50:46.033586+00:00"
  }
}
```

Figure 4.6 Shows a test IAM role being created

### Step 6: Attach Admin Policy to Role

```
aws iam attach-role-policy \  
--role-name TestUser \  
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \  
--endpoint-url $AWS_ENDPOINT_URL
```

```
(venv)-(kali@vbox)-[~]
$ aws iam attach-user-policy \
  --user-name TestUser \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
  --endpoint-url $AWS_ENDPOINT_URL
```

Figure 4.7 Shows recon commands for certificate\_transparency

---

**Step 7: Verify Policies and cleanup**

```
aws iam list-attached-role-policies --role-name TestUserRole --endpoint-url $AWS_ENDPOINT_URL
```

**Detach Policies**

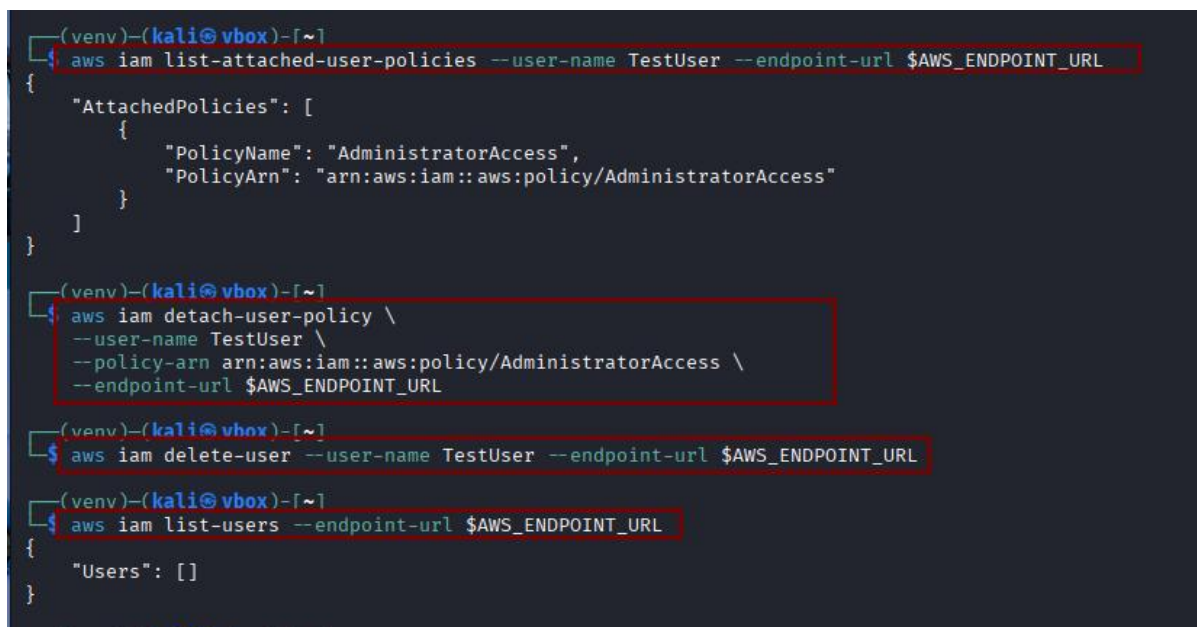
```
aws iam detach-user-policy --user-name TestUser --policy-arn arn:aws:iam::aws:policy/AdministratorAccess --endpoint-url $AWS_ENDPOINT_URL
```

**Delete IAM user**

```
aws iam delete-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
```

**Delete roles**

```
aws iam delete-role --role-name TestUser --endpoint-url $AWS_ENDPOINT_URL
```



```
(venv)-(kali@vbox)-[~]
$ aws iam list-attached-user-policies --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    }
  ]
}

(venv)-(kali@vbox)-[~]
$ aws iam detach-user-policy \
  --user-name TestUser \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
  --endpoint-url $AWS_ENDPOINT_URL

(venv)-(kali@vbox)-[~]
$ aws iam delete-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL

(venv)-(kali@vbox)-[~]
$ aws iam list-users --endpoint-url $AWS_ENDPOINT_URL
{
  "Users": []
}
```

Figure 4.8 Shows deleting users

## 5. Summary

- Using LocalStack and AWS CLI, we successfully simulated cloud privilege abuse:
- Created an over-privileged IAM role.
- Assumed the role to gain temporary admin credentials.
- Performed admin actions: created users and roles, attached policies.
- Demonstrated how misconfigured permissions can lead to full administrative access.