AWS ENUMERATION – PART II (PRACTICAL ENUMERATION)

December 14, 2022 Eduard Agavriloae aws, Cloud Security, Pentest techniques Leave a comment

We hackers love cheat sheets so here are mine for AWS IAM, EC2, S3 Buckets and Lambda Functions. In Part I we showed what approaches you can take for enumerating an AWS

Comment Reblog Subscribe

environment. This time, we'll present you a cheat sheet of commands that will help you in lateral movement, privilege escalation and data exfiltration.

1. IAM (Identity and Access Management)

This is the place that usually gives you the most powerful attack vectors. Giving the wrong permission, setting a lax role trust relationship or having groups with admin privileges are some examples of insecure configurations that I encounter all the time. The only thing left is creating an attack vector.

A cheat sheet for IAM resources:

```
# USERS
# list users
aws iam list-users
# list groups of an user
aws iam list-groups-for-user --user-name $username
# list policies attached to a user
aws iam list-user-policies --user-name $username
aws iam list-attached-user-policies --user-name $username
# list signing certificates of a user
aws iam list-signing-certificates --user-name $username
# list ssh public keys
aws iam list-ssh-public-keys --user-name $username
# get SSH key details
aws iam get-ssh-public-key --user-name $username --encoding PEM --ssh-p
# check mfa devices of users
aws iam list-virtual-mfa-devices
# check if user can login in web console
aws iam get-login-profile --user-name $username
# GROUPS
# list groups for the AWS account
aws iam list-groups
# list group policies
aws iam list-group-policies --group-name $group_name
aws iam list-attached-group-policies --group-name $group_name
# POLICIES
# list policies for AWS account
aws iam list-policies
# filter for customer managed policies
aws iam list-policies --scope Local |
# check policy details
```

Comment Reblog Subscribe

```
aws iam get-policy --policy-arn $policy_arn
# check policy version which will also details the given permission. ve
aws iam get-policy-version --policy-arn $policy_arn --version-id $versi
# check policy for user
aws iam get-user-policy --user-name $username --policy-name $policy_nam
# ROLES
# list roles for AWS account
aws iam list-roles
# check details for role
aws iam get-role --role-name $role_name
# check for policies attached to role
aws iam list-attached-role-policies --role-name $role_name
aws iam list-role-policies --role-name $role_name
# get details for those policies
aws iam get-role-policy --role-name $role --policy-name $policy
```

Now, what can you do with this information? By enumerating this you can determine what you need to do to complete the attack vector. Some examples:

- A role has admin privileges and can be assumed by all the users within the account. Two users do not use MFA. The missing piece of the attack vector would be the compromise of one user account from those two.
- A group has admin privileges. We need to find a way to add an user that's in our control there
- A role for EC2 instances has admin privileges. We need to compromise an EC2 instance that's using this role and exfiltrate the access credentials

We keep things simple for the moment, but we don't actually need IAM resources with admin privileges. Is enough to compromise resources with permissions that can lead to a privilege escalation attack.

All privilege escalation vectors include IAM actions. Here is a list of actions that can directly lead to privesc and should be further analyzed:

- iam:PutGroupPolicy
- iam:PutRolePolicy
- iam:PutUserPolicy

Comment Reblog Subscribe

- iam:AttachGroupPolicy
- iam:AttachRolePolicy
- iam:AttachUserPolicy
- iam:CreatePolicyVersion
- iam:AddUserToGroup
- iam:CreateLoginProfile
- iam:UpdateLoginProfile
- iam:CreateAccessKey

2. S3 Buckets

S3 buckets are interesting and more than often misconfigured. The worst thing is having a public bucket with private data, but there are other things that matter as well. For example, missing encryption at rest or in transit, missing access logging, missing versioning where needed and so on.

The cheat sheet for S3 Bucket enumeration:

```
# if the endpoint is private, then it must be used the --endpoint switc
# aws --endpoint http://$ip:$port s3api list-buckets
# list buckets
aws s3api list-buckets --query "Buckets[].Name"
aws s3 ls
# check bucket location
aws s3api get-bucket-location --bucket $bucket_name
# enumerate bucket objects
aws s3api list-objects-v2 --bucket $bucket_name
aws s3api list-objects --bucket $bucket_name
aws s3 ls $bucket_name
# check object versions
aws s3api list-object-versions --bucket $bucket_name
# check bucket ACLs and object ACLs
aws s3api get-bucket-acl --bucket $bucket_name
aws s3api get-object-acl --bucket $bucket_name --key $file_name
# download objects from the S3 bucket
aws s3 cp s3://$bucket_name/$file_name $local_path
# check bucket policy status
aws s3api get-bucket-policy-status --
```

Comment Reblog Subscribe

```
# check public access for a bucket
aws s3api get-public-access-block --bucket $bucket_name
# check if object listing is allowed for anonymous users
# should get something like directory listing if allowed
curl http://$domain/$bucket_name | xmllint --format -f
# check if ListBucket is explicitly allowed
aws s3api get-bucket-policy --bucket $bucket_name
```

I've seen organizations storing access keys and other credentials in S3 buckets, so having access inside the bucket can be very useful.

A bucket can be made public in multiple ways. One way is to explicitly make it public by not blocking public access from the web console.

Another way, more subtle to errors, is through the use of bucket policy. The bucket can be made public by allowing all principles to perform actions on the bucket.

My favorite way however is when people grant access to "Authenticated" principles, believing that they grant access to the users within the account. In fact, this grants access to any AWS user from the internet, which is almost as making the bucket public.

If you don't have listing permissions over the bucket you can enumerate it as an web application with tools like dirb or gobuster.

You can find bucket names by following the next URL structure: https://bucket-name.s3.amazonaws.com.

Now, for enumerating files, all you have to do is run a normal directory enumeration using a wordlist.

3. EC2

Having access to an EC2 instance can give you the right foothold for moving to an on-premises AD network or target other cloud services.

Comment Reblog Subscribe

In most cases, people deploy EC2 instances and need them to perform some kind of actions, but for that you need permissions. The recommended way to grant permissions to an EC2 instance is through roles. Not every cloud engineer is aware that the access keys can be exfiltrated through the Metadata API.

The worst thing is when the EC2 instance exposes a vulnerable web application to the internet and the access keys can be exfiltrated from there. However, that's not the only thing that matters. What if you can connect or run commands on that EC2 instance? Maybe credentials are stored in the instance's user data. Going even further, maybe the EC2 instance can communicate with critical systems from other VPCs.

Below is a cheat sheet for enumerating the most important aspects of EC2.

```
# EC2
# list instances
aws ec2 describe-instances
# check if they use Metadata API version 1 (easier to exfil access keys
aws ec2 describe-instances --filters Name=metadata-options.http-tokens,
# get user data of instances and look for secrets
aws ec2 describe-instance-attribute --instance-id $id --attribute userD
# list volumes
aws ec2 describe-volumes
# list snapshots (check if anything is public)
aws ec2 describe-snapshots
# list security groups
aws ec2 describe-security-groups
# list security groups that allow SSH from the internet (from-port and
aws ec2 describe-security-groups --filters Name=ip-permission.from-port
# better yet, just check for ingress rules from the internet
aws ec2 describe-security-groups --filters Name=ip-permission.cidr,Valu
# get EC2 instances that are part of a fleet
aws ec2 describe-fleet-instances
# get details about existing fleets
aws ec2 describe-fleets
# list dedicated hosts that can contains multiple EC2 instances
aws ec2 describe-hosts
# list profile association for each instances
aws ec2 describe-iam-instance-profile-associations
# find what role is allocated to a instance profile
aws iam get-instance-profile --instan
# display private AMIs
```

Comment Reblog Subscribe

```
aws ec2 describe-images --filters "Name=is-public,Values=false"
# list names of SSH keys
aws ec2 describe-key-pairs
# retrieve latest console output from an instance
# output differs based on OS
aws ec2 get-console-output --instance-id $id --output text
# takes a screenshot of the terminal and returns it as base64
aws ec2 get-console-screenshot --instance-id $id
# get the admin password for an EC2 instance
# the returned password is encrypted with the key pair specified when c
# if you want the clear text data, include the next parameter: --priv-l
aws ec2 get-password-data --instance-id $id
# VPN
# list client VPN endpoints
aws ec2 describe-client-vpn-endpoints
# list active connections or connections terminated in the last 60 minu
# can return domain users if VPN integrated with AD
aws ec2 describe-client-vpn-connections --client-vpn-endpoint-id $id
# check if anyone can connect, from which AD group and more
aws ec2 describe-client-vpn-authorization-rules --client-vpn-endpoint-i
# list customer VPN gateways
aws ec2 describe-customer-gateways
# list site to site VPN connections
aws ec2 describe-vpn-connections
# Network
# list elastic IPs
aws ec2 describe-addresses
# list gateways
aws ec2 describe-internet-gateways
aws ec2 describe-local-gateways
aws ec2 describe-nat-gateways
aws ec2 describe-transit-gateways
aws ec2 describe-vpn-gateways
# list network interfaces (a lot of useful information)
aws ec2 describe-network-interfaces
# list VPCs
aws ec2 describe-vpcs
# list subnets
aws ec2 describe-subnets
# list network ACLs
# useful when filtering on specific VPCs/subnets (--filters Name=vpc-id
aws ec2 describe-network-acls
# list VPC endpoints
aws ec2 describe-vpc-endpoints
# list allowed connections between VPC pairs
aws ec2 describe-vpc-peering-connections
```

Comment Reblog Subscribe

You can formulate various attack vectors based on the collected information, for example:

- Find EC2 instances with high permissions that expose web applications to the internet. Try to exploit the web applications in order to get the access credentials.
- Search for EC2 instances with SSH open. Try to find the SSH key from S3 buckets, the instance's user data or other AWS services. After connecting to it, exfiltrate access credentials or leverage any peer connection between VPCs in order to access private instances
- Download any public EBS snapshot or start an instance with a public AMI. Look over the files on disk.

4. Lambda Functions

Lambda Functions were always an interesting point of attack because, if a function has an execution role, you can exfiltrate the access credentials from it by reading the environmental variable from "/proc/self/environ".

The best attack with Lambda Functions is to create a new function (or edit an existing one) that would retrieve the access credentials. Make sure you pass it an execution role with high privileges. This is one well known privilege escalation vector in AWS. But it's not always possible and when that's the case, my focus is on checking if any credentials are passed as environmental variables, if a role with high privileges is attached to it and if there are any source code vulnerabilities that can be exploited.

Until recently, the way to make a Lambda Function accessible from the internet was to integrate it with an API Gateway. Now, there is a new method to do that, by enabling function URL. This would generate a link for your function that can be accessed directly from a browser. While it's still not so popular, I believe it increases the security risk of the environment. Finding vulnerabilities from a source code review can be more rewarding for functions that use this feature so make sure to keep an eye on them.

Additionally, you might use this as a backdoor: add a snippet of code in an existing function that would retrieve the access credentials, enable the

Comment Reblog Subscribe

that's it. No need to have the permission of invoking the function.

The Lambda Function cheat sheet:

```
# List functions
aws lambda list-functions
# get a single function
# the --function-name supports both arn and function name on all comman
# you can get a specific version with "--function-name $name:$version"
aws lambda get-function --function-name $name
# list versions of function
aws lambda list-versions-by-function --function-name $name
# get function's code download link
aws lambda get-function --function-name $name --query 'Code.Location'
# get information like the role attached to the function, version in us
aws lambda get-function-configuration --function-name $name
# get the function's resource-based policy
aws lambda get-policy --function-name $name
# list events that invoked functions
aws lambda list-event-source-mappings --function-name $name
# get configuration for successful and unsusscessful invocations
# good to know if someone will be alerted in case you generate errors i
aws lambda get-function-event-invoke-config --function-name $name
# list urls for function
aws lambda list-function-url-configs --function-name $name
# get the function's URL (if enabled) along with the authentication met
# if URL is enabled and authentication is NONE it meens that can be acc
aws lambda get-function-url-config --function-name $name
# list lambda layers which are like reusable code libraries
aws lambda list-layers
# get the layer version's arn
aws lambda get-layer-version-by-arn --arn $layer_arn
# get a link to download the layer so that you can look through the cod
aws lambda get-layer-version --layer-name $layer_name --version-number
# get policy of a layer and look for misconfiguarions like over permiss
aws lambda get-layer-version-policy --layer-name $layer_name --version-
```

I often encounter functions with source code vulnerabilities as developers think the functions are far from prying eyes and they don't apply the same secure practices as when building internet facing applications.

Comment Reblog Subscribe •••

If you're doing a cloud configuration review, you might not have time to check the code of multiple functions, but if you're in a red team engagement, these function can represent the next step in escalating your privileges so performing source code review might be worth it.

As a last tip, check if the functions are storing credentials in the code/configuration files. Usually the luck is on your side.

5. Closing thoughts

There are other services that deserve a chapter, like API Gateways, Secrets Manager, Parameter Store, KMS and so on. They might be in a future article as this one got quite long.

Until then, make sure to check these commands, leave a comment with suggestions if you feel like I missed something and good luck in getting AdminstratorAccess.

Share this:

Facebook Twitter LinkedIn Reddit

Loading...

Related

AWS Enumeration – Part I (Where to start, Approaches and Tools)
November 1, 2022
In "aws"

IAMActionHunter in action
July 27, 2023
In "aws"

Certified Hybrid Multi-Cloud Red Team Specialist – Review and Tips
August 1, 2022
In "Misc"

Comment Reblog Subscribe

AWS CLOUD ENUMERATION

« AWS Enumeration – Part I (Where to start, Approaches and Tools)

AWS ssm:SendCommand or network agnostic built-in RCE as root »

LEAVE A REPLY

Search

TOP POSTS

- Introduction to AI Prompt Injections (Jailbreak CTFs)
- How to Install .ipa Files on iPhone Without Jailbreak
- Mobile Pentesting 101 - Bypassing Biometric Authentication
- AWS CloudQuarry: Digging for Secrets in Public AMIs
- Mobile Pentesting 101 - How to Set Up Your iOS Environment

BLOG STATS

- 412,570 hits

FOLLOW US VIA EMAIL

Enter your email address to follow this blog and receive notifications of new posts by email.

Comment Reblog Subscribe

Follow

Join 150 other subscribers

CATEGORIES

- Active Directory (4)
- Announcements (2)
- C2 (1)
- Cloud Security (11)
  - aws (9)
  - Azure (2)
  - Kubernetes (1)
- Conferences (4)
- Embedded systems security (3)
  - IoT Pentesting (2)
- Ethical Hacking (18)
- General security (13)
- IT Security Assurance (1)
- IT Security Audit (2)
- Metasploit (1)
- Misc (20)
  - Artificial Intelligence (1)
  - Code Review (1)
  - CVE (1)
- Mobile security (11)
- Network security (7)

Comment Reblog Subscribe

- Operating systems (2)
- Penetration Testing (23)
- Pentest techniques (30)
- Research (1)
- Web security (13)
- Wireless security (1)

Blog at WordPress.com.

Comment Reblog Subscribe