



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 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:

```
1 # USERS
2 # list users
3 aws iam list-users
4 # list groups of an user
5 aws iam list-groups-for-user --user-name $username
6 # list policies attached to a user
7 aws iam list-user-policies --user-name $username
8 aws iam list-attached-user-policies --user-name $username
```

Search

🔍

TOP POSTS

- Introduction to AI Prompt Injections (Jailbreak CTFs)
- How to Install .ipa Files on iPhone Without Jailbreak
- Mobile Pentesting 101 - How to set up your Android Environment
- Passing Your OSCP In 2023 (or 2024)
- Mobile Pentesting 101: How to Install Split APKs

BLOG STATS

- 412,775 hits

FOLLOW US VIA EMAIL

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

Email Address

Follow

```
9 # list signing certificates of a user
10 aws iam list-signing-certificates --user-name $username
11 # list ssh public keys
12 aws iam list-ssh-public-keys --user-name $username
13 # get SSH key details
14 aws iam get-ssh-public-key --user-name $username --encoding P
15 # check mfa devices of users
16 aws iam list-virtual-mfa-devices
17 # check if user can login in web console
18 aws iam get-login-profile --user-name $username
19 # GROUPS
20 # list groups for the AWS account
21 aws iam list-groups
22 # list group policies
23 aws iam list-group-policies --group-name $group_name
24 aws iam list-attached-group-policies --group-name $group_name
25 # POLICIES
26 # list policies for AWS account
27 aws iam list-policies
28 # filter for customer managed policies
29 aws iam list-policies --scope Local | grep -A2 PolicyName
30 # check policy details
31 aws iam get-policy --policy-arn $policy_arn
32 # check policy version which will also details the given perm
33 aws iam get-policy-version --policy-arn $policy_arn --version
34 # check policy for user
35 aws iam get-user-policy --user-name $username --policy-name $
36 # ROLES
37 # list roles for AWS account
38 aws iam list-roles
39 # check details for role
40 aws iam get-role --role-name $role_name
41 # check for policies attached to role
42 aws iam list-attached-role-policies --role-name $role_name
43 aws iam list-role-policies --role-name $role_name
44 # get details for those policies
45 aws iam get-role-policy --role-name $role --policy-name $poli
```

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
- iam:AttachGroupPolicy
- iam:AttachRolePolicy
- iam:AttachUserPolicy
- iam:CreatePolicyVersion
- iam:AddUserToGroup

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

- 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:

```
1 # if the endpoint is private, then it must be used the --endp
2 # aws --endpoint http://$ip:$port s3api list-buckets
3 # list buckets
4 aws s3api list-buckets --query "Buckets[].Name"
5 aws s3 ls
6 # check bucket location
7 aws s3api get-bucket-location --bucket $bucket_name
8 # enumerate bucket objects
9 aws s3api list-objects-v2 --bucket $bucket_name
10 aws s3api list-objects --bucket $bucket_name
11 aws s3 ls $bucket_name
12 # check object versions
13 aws s3api list-object-versions --bucket $bucket_name
14 # check bucket ACLs and object ACLs
15 aws s3api get-bucket-acl --bucket $bucket_name
16 aws s3api get-object-acl --bucket $bucket_name --key $file_na
17 # download objects from the S3 bucket
18 aws s3 cp s3://$bucket_name/$file_name $local_path
19 # check bucket policy status
20 aws s3api get-bucket-policy-status --bucket $bucket_name --ou
21 # check public access for a bucket
22 aws s3api get-public-access-block --bucket $bucket_name
23 # check if object listing is allowed for anonymous users
24 # should get something like directory listing if allowed
25 curl http://$domain/$bucket_name | xmllint --format -f
26 # check if ListBucket is explicitly allowed
27 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.

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.

```
1 # EC2
2 # list instances
3 aws ec2 describe-instances
4 # check if they use Metadata API version 1 (easier to exfil a
5 aws ec2 describe-instances --filters Name=metadata-options.ht
6 # get user data of instances and look for secrets
7 aws ec2 describe-instance-attribute --instance-id $id --attril
8 # list volumes
9 aws ec2 describe-volumes
10 # list snapshots (check if anything is public)
11 aws ec2 describe-snapshots
12 # list security groups
13 aws ec2 describe-security-groups
14 # list security groups that allow SSH from the internet (from
15 aws ec2 describe-security-groups --filters Name=ip-permission
16 # better yet, just check for ingress rules from the internet
17 aws ec2 describe-security-groups --filters Name=ip-permission
18 # get EC2 instances that are part of a fleet
19 aws ec2 describe-fleet-instances
20 # get details about existing fleets
21 aws ec2 describe-fleets
22 # list dedicated hosts that can contains multiple EC2 instance
23 aws ec2 describe-hosts
24 # list profile association for each instances
25 aws ec2 describe-iam-instance-profile-associations
26 # find what role is allocated to a instance profile
27 aws iam get-instance-profile --instance-profile-name $name
28 # display private AMIs
29 aws ec2 describe-images --filters "Name=is-public,Values=false"
30 # list names of SSH keys
31 aws ec2 describe-key-pairs
32 # retrieve latest console output from an instance
33 # output differs based on OS
34 aws ec2 get-console-output --instance-id $id --output text
35 # takes a screenshot of the terminal and returns it as base64
36 aws ec2 get-console-screenshot --instance-id $id
37 # get the admin password for an EC2 instance
38 # the returned password is encrypted with the key pair specif
39 # if you want the clear text data, include the next parameter
40 aws ec2 get-password-data --instance-id $id
41 # VPN
42 # list client VPN endpoints
43 aws ec2 describe-client-vpn-endpoints
44 # list active connections or connections terminated in the la
45 # can return domain users if VPN integrated with AD
```



```
46 aws ec2 describe-client-vpn-connections --client-vpn-endpoint
47 # check if anyone can connect, from which AD group and more
48 aws ec2 describe-client-vpn-authorization-rules --client-vpn-
49 # list customer VPN gateways
50 aws ec2 describe-customer-gateways
51 # list site to site VPN connections
52 aws ec2 describe-vpn-connections
53 # Network
54 # list elastic IPs
55 aws ec2 describe-addresses
56 # list gateways
57 aws ec2 describe-internet-gateways
58 aws ec2 describe-local-gateways
59 aws ec2 describe-nat-gateways
60 aws ec2 describe-transit-gateways
61 aws ec2 describe-vpn-gateways
62 # list network interfaces (a lot of useful information)
63 aws ec2 describe-network-interfaces
64 # list VPCs
65 aws ec2 describe-vpcs
66 # list subnets
67 aws ec2 describe-subnets
68 # list network ACLs
69 # useful when filtering on specific VPCs/subnets (--filters Name=tag:Name=tag:Name)
70 aws ec2 describe-network-acls
71 # list VPC endpoints
72 aws ec2 describe-vpc-endpoints
73 # list allowed connections between VPC pairs
74 aws ec2 describe-vpc-peering-connections
```

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 AML. 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 function's URL if is not already and that's it. No need to have the permission of invoking the function.

The Lambda Function cheat sheet:

```
1  # List functions
2  aws lambda list-functions
3  # get a single function
4  # the --function-name supports both arn and function name on
5  # you can get a specific version with "--function-name $name:
6  aws lambda get-function --function-name $name
7  # list versions of function
8  aws lambda list-versions-by-function --function-name $name
9  # get function's code download link
10 aws lambda get-function --function-name $name --query 'Code.L
11 # get information like the role attached to the function, ver
12 aws lambda get-function-configuration --function-name $name
13 # get the function's resource-based policy
14 aws lambda get-policy --function-name $name
15 # list events that invoked functions
16 aws lambda list-event-source-mappings --function-name $name
17 # get configuration for successful and unsusscessful invocati
18 # good to know if someone will be alerted in case you generat
19 aws lambda get-function-event-invoke-config --function-name $
20 # list urls for function
21 aws lambda list-function-url-configs --function-name $name
22 # get the function's URL (if enabled) along with the authentic
23 # if URL is enabled and authentication is NONE it means that
24 aws lambda get-function-url-config --function-name $name
25 # list lambda layers which are like reusable code libraries
26 aws lambda list-layers
27 # get the layer version's arn
28 aws lambda get-layer-version-by-arn --arn $layer_arn
29 # get a link to download the layer so that you can look throug
30 aws lambda get-layer-version --layer-name $layer_name --versi
31 # get policy of a layer and look for misconfiguarions like ov
32 aws lambda get-layer-version-policy --layer-name $layer_name
```

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.

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 AdministratorAccess.

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"

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