# AWS Pentesting

> Support HackTricks and get benefits!

## Basic Information

**Before start pentesting** an **AWS** environment there are a few **basics things you need to know** about how AWS works to help you understand what you need to do, how to find misconfigurations and how to exploit them.

Concepts such as organization hierarchy, IAM and other basic concepts are explained in:

AWS - Basic Information

## Labs to learn

- https://github.com/RhinoSecurityLabs/cloudgoat
- https://hackingthe.cloud/aws/capture_the_flag/cicdont/
- https://github.com/BishopFox/iam-vulnerable
- https://github.com/nccgroup/sadcloud
- https://github.com/bridgecrewio/terragoat
- https://github.com/ine-labs/AWSGoat
- http://flaws.cloud/
- http://flaws2.cloud/

Tools to simulate attacks:

- https://github.com/Datadog/stratus-red-team/
- https://github.com/sbasu7241/AWS-Threat-Simulation-and-Detection/tree/main

## AWS Pentester/Red Team Methodology

In order to audit an AWS environment it's very important to know: which **services are being used**, what is **being exposed**, who has **access** to what, and how are internal AWS services an **external services** connected.

From a Red Team point of view, the **first step to compromise an AWS environment** is to manage to obtain some **credentials**. Here you have some ideas on how to do that:

- **Leaks** in github (or similar) - OSINT
- **Social** Engineering
- **Password** reuse (password leaks)
- Vulnerabilities in AWS-Hosted Applications
  - **Server Side Request Forgery** with access to metadata endpoint
  - **Local File Read**
    - `/home/USERNAME/.aws/credentials`
    - `C:\Users\USERNAME\.aws\credentials`
- 3rd parties **breached**
- **Internal** Employee
- **Cognito** credentials

Or by **compromising an unauthenticated service** exposed:

AWS - Unauthenticated Enum & Access

Or if you are doing a **review** you could just **ask for credentials** with these roles:

AWS - Permissions for a Pentest

After you have managed to obtain credentials, you need to know **to who do**

> **those creds belong**, and **what they have access to**, so you need to perform some basic enumeration:

## Basic Enumeration

### SSRF

If you found a SSRF in a machine inside AWS check this page for tricks:

Cloud SSRF
HackTricks

### Whoami

One of the first things you need to know is who you are (in where account you are in other info about the AWS env):

```
# Easiest way, but might be monitored?
aws sts get-caller-identity
aws iam get-user # This will get your own user

# If you have a Key ID
aws sts get-access-key-info --access-key-id=ASIA1234567890123456

# Get inside error message
aws sns publish --topic-arn arn:aws:sns:us-east-1:*account id*:aaa --message a

# From metadata
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-met
curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/dynam
```

> Note that companies might use **canary tokens** to identify when **tokens are being stolen and used**. It's recommended to check if a token is a canary token or not before using it.
> For more info **check this page**.

### Org Enumeration

AWS - Organizations Enum

### IAM Enumeration

If you have enough permissions **checking the privileges of each entity inside the AWS account** will help you understand what you and other identities can do and how to **escalate privileges**.

If you don't have enough permissions to enumerate IAM, you can **steal bruteforce them** to figure them out.
Check **how to do the numeration and brute-forcing** in:

AWS - IAM Enum

> Now that you **have some information about your credentials** (and if you are a red team hopefully you **haven't been detected**). It's time to figure out which services are being used in the environment.
> In the following section you can check some ways to **enumerate some common services.**

## Services Enumeration, Post-Exploitation & Persistence

AWS has an astonishing amount of services, in the following page you will find **basic information, enumeration** cheatsheets**,** how to **avoid detection**, obtain **persistence**, and other **post-exploitation** tricks about some of them:

AWS - Services

Note that you **don't** need to perform all the work **manually**, below in this post you can find a **section about automatic tools**.

Moreover, in this stage you might discovered **more services exposed to unauthenticated users,** you might be able to exploit them:

AWS - Unauthenticated Enum & Access

## Privilege Escalation

If you can **check at least your own permissions** over different resources you could **check if you are able to obtain further permissions**. You should focus at least in the permissions indicated in:

AWS - Privilege Escalation

## Publicly Exposed Services

While enumerating AWS services you might have found some of them **exposing elements to the Internet** (VM/Containers ports, databases or queue services, snapshots or buckets...).
As pentester/red teamer you should always check if you can find **sensitive information / vulnerabilities** on them as they might provide you **further access into the AWS account**.

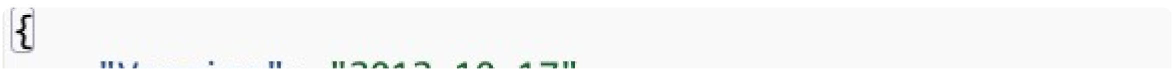
In this book you should find **information** about how to find **exposed AWS services and how to check them**. About how to find **vulnerabilities in exposed network services** I would recommend you to **search** for the specific **service** in:

HackTricks
HackTricks

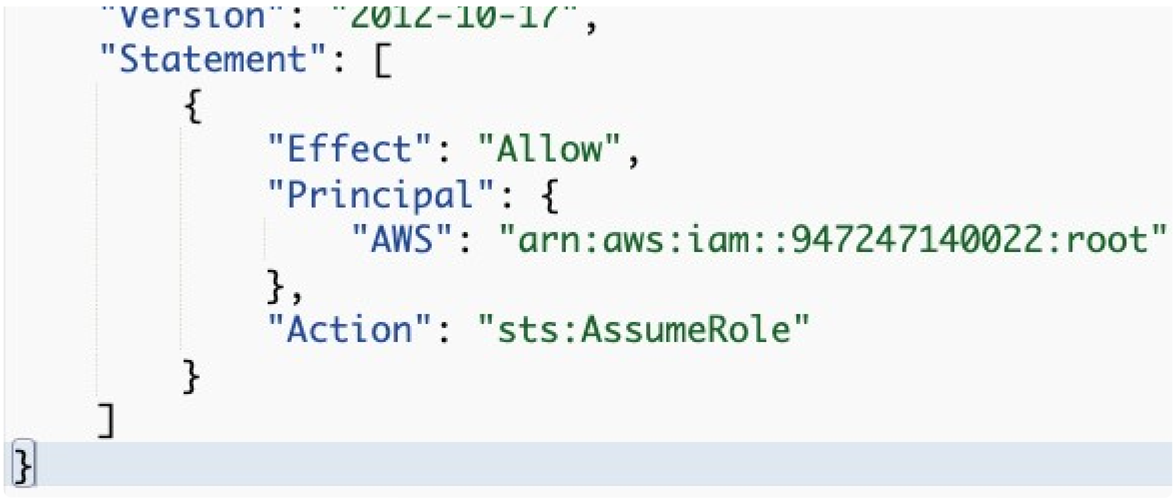
## Compromising the Organization

### From the root/management account

When the management account creates new accounts in the organization, a **new role** is created in the new account, by default named `OrganizationAccountAccessRole` and giving **AdministratorAccess** policy to the **management account** to access the new account.



```
{
```



```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::947247140022:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

So, in order to access as administrator a child account you need:

- **Compromise** the **management** account and find the **ID** of the **children accounts** and the **names** of the **role** (OrganizationAccountAccessRole by default) allowing the management account to access as admin.
  - To find children accounts go to the organizations section in the aws console or run `aws organizations list-accounts`
  - You cannot find the name of the roles directly, so check all the custom IAM policies and search any allowing **`sts:AssumeRole` over the previously discovered children accounts**.
- **Compromise** a **principal** in the management account with **`sts:AssumeRole` permission over the role in the children accounts** (even if the account is allowing anyone from the management account to impersonate, as its an external account, specific `sts:AssumeRole` permissions are necessary).

## Automated Tools

### Recon

- aws-recon: A multi-threaded AWS security-focused **inventory collection tool** written in Ruby.

```
# Install
gem install aws_recon

# Recon and get json
AWS_PROFILE=<profile> aws_recon \
  --services S3,EC2 \
  --regions global,us-east-1,us-east-2 \
  --verbose
```

- **cloudlist**: Cloudlist is a **multi-cloud tool for getting Assets** (Hostnames, IP Addresses) from Cloud Providers.
- **cloudmapper**: CloudMapper helps you analyze your Amazon Web Services (AWS) environments. It now contains much more functionality, including auditing for security issues.

```
# Installation steps in github
# Create a config.json file with the aws info, like:
{
    "accounts": [
        {
            "default": true,
            "id": "<account id>",
            "name": "dev"
        }
    ],
    "cidrs":
    {
        "2.2.2.2/28": {"name": "NY Office"}
    }
}

# Enumerate
python3 cloudmapper.py collect --profile dev
## Number of resources discovered
python3 cloudmapper.py stats --accounts dev

# Create HTML report
## In the report you will find all the info already
python3 cloudmapper.py report --accounts dev

# Identify potential issues
python3 cloudmapper.py audit --accounts dev --json > audit.json
python3 cloudmapper.py audit --accounts dev --markdow > audit.md
python3 cloudmapper.py iam_report --accounts dev

# Identify admins
## The permissions search for are in https://github.com/duo-labs/cloudmapper/b
python3 cloudmapper.py find_admins --accounts dev

# Identify unused elements
python3 cloudmapper.py find_unused --accounts dev

# Identify publivly exposed resources
python3 cloudmapper.py public --accounts dev

python cloudmapper.py prepare #Prepare webserver
python cloudmapper.py webserver #Show webserver
```

- **cartography**: Cartography is a Python tool that consolidates infrastructure assets and the relationships between them in an intuitive graph view powered by a Neo4j database.

```
# Install
pip install cartography
## At the time of this writting you need neo4j version 3.5.*

# Get AWS info
AWS_PROFILE=dev cartography --neo4j-uri bolt://127.0.0.1:7687 --neo4j-passwor
```

- **starbase**: Starbase collects assets and relationships from services and systems including cloud infrastructure, SaaS applications, security controls, and more into an intuitive graph view backed by the Neo4j database.
- **aws-inventory**: (Uses python2) This is a tool that tries to **discover all AWS resources** created in an account.
- **aws_public_ips**: It's a tool to **fetch all public IP addresses** (both IPv4/IPv6) associated with an AWS account.

### Privesc & Exploiting

- **SkyArk:** Discover the most privileged users in the scanned AWS environment, including the AWS Shadow Admins. It uses powershell. You can find the **definition of privileged policies** in the function **`Check-PrivilegedPolicy`** in https://github.com/cyberark/SkyArk/blob/master/AWStealth/AWStealth.ps1.
- **pacu**: Pacu is an open-source **AWS exploitation framework**, designed for offensive security testing against cloud environments. It can **enumerate**, find **miss-configurations** and **exploit** them. You can find the **definition of privileged permissions** in https://github.com/RhinoSecurityLabs/pacu/blob/866376cd711666c775bbfcde0524c817f2c5b181/pacu/modules/iam__privesc_scan/main.py#L134 inside the **`user_escalation_methods`** dict.
  - Note that pacu **only checks your own privescs paths** (not account wide).

```
# Install
## Feel free to use venvs
pip3 install pacu

# Use pacu CLI
pacu
> import_keys <profile_name> # import 1 profile from .aws/credentials
> import_keys --all # import all profiles
> list # list modules
> exec iam__enum_permissions # Get permissions
> exec iam__privesc_scan # List privileged permissions
```

- **PMapper**: Principal Mapper (PMapper) is a script and library for identifying risks in the configuration of AWS Identity and Access Management (IAM) for an AWS account or an AWS organization. It models the different IAM Users and Roles in an account as a directed graph, which enables checks for **privilege escalation** and for alternate paths an attacker could take to gain access to a resource or action in AWS. You can check the **permissions used to find privesc** paths in the filenames ended in `_edges.py` in https://github.com/nccgroup/PMapper/tree/master/principalmapper/graphing

```
# Install
pip install principalmapper

# Get data
pmapper --profile dev graph create
pmapper --profile dev graph display # Show basic info
# Generate graph
pmapper --profile dev visualize # Generate svg graph file (can also be png, do
pmapper --profile dev visualize --only-privesc # Only privesc permissions

# Generate analysis
pmapper --profile dev analysis
## Run queries
pmapper --profile dev query 'who can do iam:CreateUser'
pmapper --profile dev query 'preset privesc *' # Get privescs with admins

# Get organization hierarchy data
pmapper --profile dev orgs create
pmapper --profile dev orgs display
```

- **cloudsplaining**: Cloudsplaining is an AWS IAM Security Assessment tool that identifies violations of least privilege and generates a risk-prioritized HTML report. It will show you potentially **over privileged** customer, inline and aws **policies** and which **principals has access to them**. (It not only checks for privesc but also other kind of interesting permissions, recommended to use).

```
# Install
pip install cloudsplaining

# Download IAM policies to check
## Only the ones attached with the versions used
cloudsplaining download --profile dev

# Analyze the IAM policies
cloudsplaining scan --input-file /private/tmp/cloudsplaining/dev.json --output
```

- **cloudjack**: CloudJack assesses AWS accounts for **subdomain hijacking vulnerabilities** as a result of decoupled Route53 and CloudFront configurations.
- **ccat**: List ECR repos → Pull ECR repo → Backdoor it → Push backdoored image
- **Dufflebag**: Dufflebag is a tool that **searches** through public Elastic Block Storage (**EBS) snapshots for secrets** that may have been accidentally left in.

**Audit**

- **cloudsploit:** CloudSploit by Aqua is an open-source project designed to allow detection of **security risks in cloud infrastructure** accounts, including: Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), Oracle Cloud Infrastructure (OCI), and GitHub (It doesn't look for ShadowAdmins).

```
./index.js --csv=file.csv --console=table --config ./config.js

# Compiance options: --compliance {hipaa,cis,cis1,cis2,pci}
## use "cis" for cis level 1 and 2
```

- **Prowler**: Prowler is an Open Source security tool to perform AWS security best practices assessments, audits, incident response, continuous monitoring, hardening and forensics readiness.

```
# Install python3, jq and git
# Install
pip install prowler
prowler -v

# Run
prowler <provider>
prowler aws --profile custom-profile [-M csv json json-asff html]
```

- **CloudFox**: CloudFox helps you gain situational awareness in unfamiliar cloud environments. It’s an open source command line tool created to help penetration testers and other offensive security professionals find exploitable attack paths in cloud infrastructure.

```
cloudfox aws --profile [profile-name] all-checks
```

- **ScoutSuite**: Scout Suite is an open source multi-cloud security-auditing tool, which enables security posture assessment of cloud environments.

```
# Install
virtualenv -p python3 venv
source venv/bin/activate
pip install scoutsuite
scout --help

# Get info
scout aws -p dev
```

- **cs-suite**: Cloud Security Suite (uses python2.7 and looks unmaintained)
- **Zeus**: Zeus is a powerful tool for AWS EC2 / S3 / CloudTrail / CloudWatch / KMS best hardening practices (looks unmaintained). It checks only default configured creds inside the system.

Pentesting Cloud - Previous
Workspace Pentesting

- **cloud-custodian**: Cloud Custodian is a rules engine for managing public cloud

Next
AWS - Basic Information

the adhoc scripts organizations have into a lightweight and flexible tool, with unified metrics and reporting.

- **pacbot: Policy as Code Bot (PacBot)** is a platform for **continuous compliance monitoring, compliance reporting and security automation for the cloud**. In PacBot, security and compliance policies are implemented as code. All resources discovered by PacBot are evaluated against these policies to gauge policy conformance. The PacBot **auto-fix** framework provides the ability to automatically respond to policy violations by taking predefined actions.
- **streamalert:** StreamAlert is a serverless, **real-time** data analysis framework which empowers you to **ingest, analyze, and alert** on data from any environment, u**sing data sources and alerting logic you define**. Computer security teams use StreamAlert to scan terabytes of log data every day for incident detection and response.

## DEBUG: Capture AWS cli requests

```
# Set proxy
export HTTP_PROXY=http://localhost:8080
export HTTPS_PROXY=http://localhost:8080

# Capture with burp nor verifying ssl
aws --no-verify-ssl ...

# Dowload brup cert and transform it to pem
curl http://127.0.0.1:8080/cert --output Downloads/certificate.cer
openssl x509 -inform der -in Downloads/certificate.cer -out Downloads/certific

# Indicate the ca cert to trust
export AWS_CA_BUNDLE=~/Downloads/certificate.pem

# Run aws cli normally trusting burp cert
aws ...
```

## References

- https://www.youtube.com/watch?v=8ZXRw4Ry3mQ
- https://cloudsecdocs.com/aws/defensive/tooling/audit/