Step by Step AWS Cloud Hacking

SecTor 2019

@AndresRiancho
The attacker's view
The most common attack

Using **publicly available tools, widely known techniques** and **no zero days**.

This is how one of your AWS accounts will eventually get compromised.
Storing AWS credentials

Most applications running in EC2 instances need credentials to consume the AWS API.

Developers can provide these in multiple ways:

- Hard-coded into the application
- Environment variables
- Provided via an instance profile
EC2 instance metadata

AWS and other cloud service providers attach a virtual HTTP server to each compute instance. This server stores information about the instance such as host name, IP address, and instance profile credentials.
Metadata URLs

The instance metadata server lives at:

http://169.254.169.254/

The following paths are used to gain access to the instance profile credentials:

/latest/meta-data/iam/security-credentials/
/latest/meta-data/iam/security-credentials/{role-name}

The first one returns the role name, which is required to perform the second HTTP request. The second HTTP response yields a JSON document containing the credentials.
Server-Side Request Forgery
Web application vulnerable to SSRF

```python
from urllib.request import urlopen
from flask import request

@app.route('/ssrf')
def handler():
    url = request.args.get('url')
    return urlopen(url).read()
```
SSRF

Instance metadata
Compromised S3 buckets
The attacker's view
Enumerate IAM permissions

In the AWS cloud there are two ways to enumerate permissions for a given credential set:

- **Use the IAM service** to get the role permissions. In most cases this will fail because the role itself has no permission for the IAM API.

- **Call each AWS API** and analyze the response: **Brute-force**
Read only calls

Enumerate **Get***/ **List***/ **Describe**. Try anything else and you might change (break) the target AWS account and generate an undesired denial of service attack.

There are **thousands of API calls**
The `iam_bruteforce_permissions` module implements enumeration for only two services:

```python
SUPPORTED_SERVICES = ['ec2', 's3']
```
Permission enumeration

enumerate-iam.py
The attacker's view
After permission enumeration the attacker knows that he's able to run **all read-only API actions for S3 and Lambda.**

The most common scenario is that all API calls for S3 and Lambda are allowed.

Even after permission enumeration there are many things the attacker doesn't know.
Escalating IAM privileges

Most attackers will try to **elevate privileges to a principal with full access** to the AWS account.

There are **28 well known privilege escalation techniques**. Most of them are implemented in Pacu.
Lambda function and privileged role

A commonly exploited privilege escalation method:

- Identify an existing IAM role with **high privileges**
- **Create a new Lambda** and associate the role
- **Run** the lambda function

The Lambda function will have access to the IAM role, just like EC2 instances have access to EC2 instance profile credentials.
Privilege escalation

Getting * on *
Persistence via trust policies

**Trust policies** limit which principals are allowed to `iam:AssumeRole`.

In the compromised AWS account there is an existing trust policy in the **AdminRole** which allows all principals in the SSO AWS account to assume role.
Persistence via trust policies
AdminRole trust policy modification

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::925877178748:root"
    },
    "Action": "sts:AssumeRole"
}
```

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": ["arn:aws:iam::925877178748:root",
                 "arn:aws:iam::320222540496:root"]
    },
    "Action": "sts:AssumeRole"
}
```
Backdooring IAM Role

Trust policies
The Private VPC

At this point the attacker was able to gain access to most resources in the AWS account.

The **accounts payable** EC2 instance remains out of reach. The VPC is completely isolated from the Internet.
**Pivoting into VPC networks**

`vpc-vpn-pivot` automates the process of **creating a VPN** between the attacker's workstation and a VPC.
Pivoting into VPC networks

The permissions required to create a VPN connection using `vpc-vpn-pivot` depend on the technique being exploited.

When no permissions are limiting the attack the default technique with AWS Client VPN is used.
Connecting to private VPCs

vpc-vpn-pivot
YOU WIN!
Attack summary

From SSRF to complete pwn
Closing thoughts
Key takeaways

These are the **three most important things** to remember:

- It is possible to enumerate AWS credential permissions in a fast, safe and in-depth manner using `enumerate-iam`
- Cloud exploitation can be automated using `pacu`
- Private VPC networks can be breached using `vpc-vpn-pivot`

Follow [@AndresRiancho](https://twitter.com/AndresRiancho) on twitter for more interesting cloud security content
Thanks!
For hire

Does your company or startup need these services?

- Cloud Security Assessment
- Intro to AWS Hacking training
- Application Penetration Test
- Source Code Review

Let me know, I can help you deliver secure web applications.

anesriancho.com  @AndresRiancho  self@andresriancho.com