Pentesting Cloud Methodology

Basic Methodology

Each cloud has its own peculiarities but in general there are a few **common things a pentester should check** when testing a cloud environment:

- **Benchmark checks**
  - This will help you **understand the size** of the environment and **services used**
  - It will allow you also to find some **quick misconfigurations** as you can perform most of this tests with **automated tools**
- **Services Enumeration**
  - You probably won't find much more misconfigurations here if you performed correctly the benchmark tests, but you might find some that weren't being looked for in the benchmark test.
  - This will allow you to know **what is exactly being used** in the cloud env
  - This will help a lot in the next steps
- **Check exposed assets**
  - This can be done during the previous section, you need to **find out everything that is potentially exposed** to the Internet somehow and how can it be accessed.
    - Here I'm taking **manually exposed infrastructure** like instances with web pages or other ports being exposed, and also about other **cloud managed services that can be configured** to be exposed (such as DBs or buckets)
  - Then you should check **if that resource can be exposed or not** (confidential information? vulnerabilities? misconfigurations in the exposed service?)
- **Check permissions**
  - Here you should **find out all the permissions of each role/user** inside the cloud and how are they used
    - Too **many highly privileged** (control everything) accounts? Generated keys not used?... Most of these check should have been done in the benchmark tests already
    - If the client is using OpenID or SAML or other **federation** you might need to ask them for further **information** about **how is being each role assigned** (it's not the same that the admin role is assigned to 1 user or to 100)
  - It's **not enough to find** which users has **admin** permissions "*:*". There are a lot of **other permissions** that depending on the services used can be very **sensitive**.
    - Moreover, there are **potential privesc** ways to follow abusing permissions. All this things should be taken into account and **as much privesc paths as possible** should be reported.
- **Check Integrations**
  - It's highly probably that **integrations with other clouds or SaaS** are being used inside the cloud env.

- For **integrations of the cloud you are auditing** with other platform you should notify **who has access to (ab)use that integration** and you should ask **how sensitive** is the action being performed. For example, who can write in an AWS bucket where GCP is getting data from (ask how sensitive is the action in GCP treating that data).
- For **integrations inside the cloud you are auditing** from external platforms, you should ask **who has access externally to (ab)use that integration** and check how is that data being used. For example, if a service is using a Docker image hosted in GCR, you should ask who has access to modify that and which sensitive info and access will get that image when executed inside an AWS cloud.

Multi-Cloud tools

There are several tools that can be used to test different cloud environments. The installation steps and links are going to be indicated in this section.

PurplePanda

A tool to **identify bad configurations and privesc path in clouds and across clouds/SaaS.**

Prowler

It supports **AWS, GCP & Azure**. Check how to configure each provider in https://docs.prowler.cloud/en/latest/#aws

CloudSploit

AWS, Azure, Github, Google, Oracle, Alibaba

ScoutSuite

AWS, Azure, GCP, Alibaba Cloud, Oracle Cloud Infrastructure

Steampipe

AWS & GCP

~~cs-suite~~

AWS, GCP, Azure, DigitalOcean. It requires python2.7 and looks unmaintained.

Nessus

Nessus has an ***Audit Cloud Infrastructure*** scan supporting: AWS, Azure, Office 365, Rackspace, Salesforce. Some extra configurations in **Azure** are needed to obtain a **Client Id**.

cloudlist

Cloudlist is a **multi-cloud tool for getting Assets** (Hostnames, IP Addresses) from Cloud Providers.

cartography

GCP

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

starbase

GCP

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.

SkyArk

Discover the most privileged users in the scanned AWS or Azure environment, including the AWS Shadow Admins. It uses powershell.

Cloud Brute

A tool to find a company (target) infrastructure, files, and apps on the top cloud providers (Amazon, Google, Microsoft, DigitalOcean, Alibaba, Vultr, Linode).

CloudFox

- CloudFox is a tool to find exploitable attack paths in cloud infrastructure (currently only AWS & Azure supported with GCP upcoming).
- It is an enumeration tool which is intended to compliment manual pentesting.
- It doesn't create or modify any data within the cloud environment.

More lists of cloud security tools

- https://github.com/RyanJarv/awesome-cloud-sec

Google

GCP

GCP Pentesting

Workspace

GWS - Workspace Pentesting

AWS

AWS Pentesting

Azure

Azure Pentesting

Attack Graph

Stormspotter

creates an “attack graph” of the resources in an Azure subscription. It enables red teams and pentesters to visualize the attack surface and pivot opportunities within a tenant, and supercharges your defenders to quickly orient and prioritize incident response work.

Office365

You need **Global Admin** or at least **Global Admin Reader** (but note that Global Admin Reader is a little bit limited). However, those limitations appear in some PS modules and can be bypassed accessing the features **via the web application**.