Top 5 security tips: AWS Cloud Infrastructure

Faraday

Join us at faradaysec.com

Top 5 security tips: AWS Cloud Infrastructure

Are you trying to improve your security posture on AWS?

Read the following recommendations and tips below to avoid these common cloud security issues. This top 5 has been created by our Security Consulting team, based on real-world experience from diverse assessments performed during the last years.

1. Logging

One of the most common issues in Cloud architectures is the lack of logging mechanisms for services used in AWS..

Logs are registers of usage and performance of applications. After attacks, logs may be consulted to get information on how the attack happened and on the damages suffered in what is called forensic analysis.

For organizations of higher security maturity levels, logs can even be useful in real-time, to detect and mitigate attacks as soon as they happen.

During our security assessment projects in Cloud architectures -whether AWS, GCP, or Azure- one of the key points to audit is how logging policies are configured for each utilized service.

A basic validation should answer “yes” to all the following questions:

- Is logging activated for the service?
- Is only the useful information written in logs? Writing everything with no filter is inefficient and could cause other security issues.
- Are logs being stored in an independent AWS account? Isolation from productive

infrastructure is important, as logs should be preserved when the prod account is compromised.

- Are accesses to logging services like CloudTrail and CloudWatch configured in IAM, using the principle of least privilege? Following this principle reduces the attack surface by minimizing the chances of an attacker compromising related logs and processes.

2. Principle of least privilege

The principle of least privilege consists of providing users in the organization only those privileges and accesses required to carry out their tasks. Not more, not less.

Why is this principle useful for a Cloud infrastructure?

Suppose users have the least possible privileges, and an attack compromises the AWS account by exploiting a particular vulnerability. In that case, applying this principle will create an adverse context for the attackers. It will prevent them from accessing all services and information and/or escalating their privileges to higher ones. Of course, this depends on the exploited vulnerability and other particular factors in each scenario.

A very common mistake is to grant “Administrator” privileges to developers, for example. Granting unlimited access to users that don’t require them exposes the Cloud infrastructure to higher attack vectors, increasing the risk of a total compromise, including the information stored in the Cloud.

How can we verify that we follow this principle in our Cloud?

First, we need to obtain a map of all users and groups and their privileges to verify that these accesses are the least necessary for each.

For example, a simple way to verify if we are following this principle is to check that all users have used their assigned privileges during the last 90 days.

We can verify which users haven't been using their privileges or have never used them using the AWS Access Advisor tool. We can assume that these privileges are no longer needed and can be removed in this case.

A second recommendation is to use tools that allow the creation of a map of users, groups, roles and privileges, looking for paths to execute privilege escalations. Purple Panda, Pacu iam__privesc_scan, and AWS Access Advisor are the best options to start.

3. AWS Instance Metadata API

AWS Instance Metadata API (IMDS) is an API that contains information about the EC2 instance being used. This information is used to configure or manage the current instance and even access other services of the AWS account being used (depending on the role set in the instance).

This last detail is important: IMDS API exposes temporary credentials that an attacker could use to access certain services, according to the role set in the instance.

Since this API is an API Rest Json that is available at a specific local address (169.254.169.254), some attacks, such as a Server Side Request Forgery (the most common and mentioned one) or others -like poor Firewall settings and Reverse proxies-, may take advantage of the existence of this API to exploit a vulnerability in the application and get access to the AWS account.

How do we prevent these kinds of attacks?

IMDSv1 doesn't contain any mitigation to avoid these kinds of attacks.

In IMDSv2, the second API version, AWS added a token verification to validate and allow the request. This invalidates SSRF attacks since attackers don't have access to the required token.

The recommendation is to activate IMDSv2 in all ECS instances that contain web applications running or are exposed to a Firewall and/or reverse proxy.

4. AWS S3 buckets

AWS S3 is a storing object service that offers scalability, data availability, security, and performance.

These buckets can be set as public for any user -with no authentication or authorization needed- to access the information stored in them. Since this data might be private and/or sensitive to the organization, AWS S3 has become the most known leak vector in the Cloud.

The recommendation is to verify that the “Block all public access” policy is activated in the AWS account. This way, no bucket will be public under any circumstance

If public buckets are needed, it’s necessary to evaluate the related accesses for each one and use the principle of least privilege to reduce the attack surface

An excellent tool to quickly detect this type of issue is Prowler and its check: check_extra73

For more information about monitoring and alerting public buckets, go to: https://aws.amazon.com/blogs/security/how-to-use-aws-config-to-monitor-for-and-respond-to-amazo n-s3-buckets-allowing-public-access/

5. AWS ECR Container Images

Amazon ECR is a complete registry of containers that offers high-performance hosting, allowing the deployment of images and artifacts anywhere.

This registry allows images of containers and/or artifacts to be public. This way, information that might be sensitive to the organization includes source code, internal applications, and API/third-party apps. And more- it could be compromised by an external attacker.

The recommendation is to verify that all images and artifacts are private and compliant with an authorization process. This way, we prevent a potential leak of sensitive information and intellectual property.

Again, Prowler and its check (check_extra77) is an excellent tool to detect these issues quickly.

Truvy, Anchor, and Clair are also useful for searching vulnerabilities in images of AWS ECR containers.

Last but not least, here are some general recommendations that are always good to remember:

- Activate multi-factor authentication for all accounts through software.
- Regularly rotate access passwords every 90 days.
- Apply a strong password policy: 14 characters minimum with not-repeated symbols and numbers.
- Use AWS Configuration and AWS Organizations to separate the environments from applications, resources, and services. Apply security policies AWS account level.
- For more good practices, check the AWS Cloud Adoption Framework.