ManageEngine
Log360

5 common AWS configuration mistakes that lead to cyberattacks

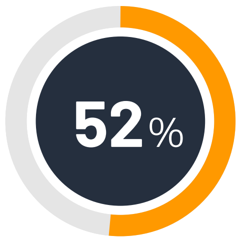
www.manageengine.com/log360

Table of contents

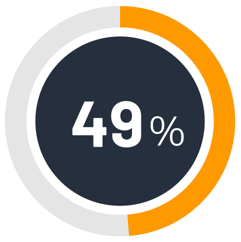
Introduction

Cloud technology has been in the market for almost a decade. Now, with remote work adoption on the rise, 59 percent of enterprises' cloud usage exceed their prior plans. While convenient for business operations, this boom in cloud adoption and remote work is also a magnet for cybercriminals to launch attacks, and many companies are ill-prepared for cloud security threats. Threat actors are watching everything; they monitor businesses constantly and look for vulnerabilities and opportunities to hack.

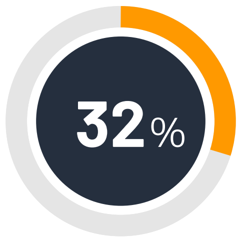
When it comes to the cloud, hackers often exploit human error and behavior rather than technical security flaws. According to a recent survey, cloud **misconfiguration is the top reason for data breaches in the cloud.** Further, breaking down this statistic reveals that



of respondents cited **lack of awareness of** cloud security and policies resulted in a data breach.



of respondents stated that **lack of adequate controls** and oversight is responsible for data leak.



think that negligent **insider behavior** gave rise to security incidents.

These numbers reveal a clear need to manage cloud misconfigurations more effectively and efficiently to avoid untoward incidents. They indicate the need to deploy a tool that can provide better visibility into cloud infrastructure, automatically detect and remediate misconfigurations, and send timely notifications in case of inappropriate and dangerous configuration changes.

In this e-book, we'll talk about the common misconfigurations in Amazon Web Services (AWS) that you should avoid in order to thwart security incidents and breaches.

Configuring AWS Cloud Trail:
The prerequisite for auditing

To gain complete visibility into your AWS cloud environment and to tighten security configurations, you first need to **enable CloudTrail log collection**. CloudTrail is an AWS service that provides event history for all AWS account activity. It generates log data for all the API calls made within AWS, including AWS management console, software development kits (SDKs), command-line tools, and other AWS services. Enabling CloudTrail logging for all regions helps prevent potential monitoring gaps, and also enables you to comply with regulatory mandates and conduct post-incident forensic investigations.

The log data generated here is stored in an Amazon S3 bucket. However, if cyberattackers were to intrude into your cloud network, the first thing they would do is disable your CloudTrail and try to compromise the log files. For this reason, it's vital to take the below precautions to make your CloudTrail log files attack-proof:

- **Ensure CloudTrail log file integrity:** Enable CloudTrail log file integrity validation so you can track any changes made to the log file data after it is stored in the S3 bucket.
- **Prevent unauthorized access attempts:** Enable access logging for the CloudTrail S3 bucket so you can track all access requests and spot unauthorized access attempts.
- **Double-up security to access log files:** Turn on the multi-factor authentication (MFA) option for deleting CloudTrail S3 buckets so even if an account is compromised, you can ensure the safety of log files.

Common AWS configuration mistakes and how to fix them

Mistake 1

Providing unrestricted access to EC2 security groups

The security groups (SGs) associated with AWS Elastic Cloud Compute (EC2) instances are similar to a firewall, and they control access at the protocol and port levels. These SGs are a set of rules that filter the incoming (ingress) and outgoing (egress) traffic of an EC2 instance. An EC2 instance can have multiple SGs, and the rules of an SG can be modified at any time. These SG rules enable a specific source, like an IP or range of IP addresses, to access the instance using destination ports or a certain protocol, like the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP), the Simple Mail Transfer Protocol (SMTP), or HTTP/HTTPS. While configuring the SG rules, ensure that you:

- Give permissions to only specific IP ranges to access the EC2 instances.
- Allow traffic only from specified ports. Opening all ports is not recommended as it opens the instance for unwanted traffic.

Not taking these into account while configuring SGs could not only hinder daily operations, but also attract a wide range of malicious attacks such as brute-force, denial-of-service (DoS), or man-in-the middle (MITM).

How to detect and fix this issue
When you enable logging using CloudTrail and monitor log data using a third-party log management tool or native solution, you should be able to detect unrestricted access. Configure your monitoring tool to detect when the SGs attached to your EC2 instances allow unrestricted access for ports other than 25 (SMTP), 80, and 443 (HTTP and HTTPS).

Mistake 2

Making Amazon Machine Images (AMIs) public, or providing unrestricted access to the AMIs

An AMI provides the required information to launch an EC2 instance. When you create an EC2 instance, you must specify the AMI it belongs to. An AMI includes:

- The permissions that control the AWS accounts for launching EC2 instances.
- A block device mapping that specifies the volumesthat are to be attached to the instance when it's being launched.
- A template that contains the software configuration (operating system, application server, and applications) that will be used to launch the EC2 instance.

AWS users can create their own AMIs, utilize publicly available AMIs, or purchase custom AMIs. When a user creates an AMI, they have the option to make the AMI public, share it with only specific accounts, or make it private.

Unrestricted access to AMIs or making them public will result in these AMIs being available in the Amazon Community; anyone who has an AWS account can use these publicly available AMIs to launch EC2 instances.

How to fix this issue

As AMIs contain proprietary or sensitive data, like the snapshots of enterprise-specific applications, you should ensure that they are always set to Private. Any AMI that is publicly available should be monitored carefully.

Mistake 3

Not terminating unused user access keys

The AWS Identity and Access Management (IAM) user access keys are long-term credentials for IAM users or AWS account route users to access AWS CLIs or APIs. It is essential that you judiciously provide user access keys to your employees and also monitor the user access keys on a regular basis toremove unused or inappropriate user access keys.

Anyone who has your access key gets the same level of access to your AWS resources. AWS takes significant steps to protect your access keys; keeping in mind the shared-responsibility model, you should also ensure the security of your access keys. Not removing unused access keys can result in compromised user accounts or even takeover.Further, removing these unused keys will help you comply with many regulations such as ISO 27001, HIPAA, PCI DSS, and more.

Bonus tips:

Apart from removing the unused user access keys, ensure that you take the below security steps as well:

- Do not provide access keys to your third-party vendors, as they could gain permanent access to your account.
- Use temporary security credentials (IAM roles) wherever possible instead of access keys. This will help prevent unauthorized access by unintended users.
- If possible, do not have account access keys for your AWS account root user. If you have access keys for your root user, a single compromise might expose your entire cloud resource.
- Use different access keys for different applications. This will help secure the other applications if one of the app's access keys is compromised.
- Rotate your access keys periodically to reduce the risk of being compromised.
- Configure MFA for your most sensitive operations so your resources stay safe even if the access key is compromised.

Bonus tips:
Which compliance requirements mandate you to manage unused user keys?

- ISO 27001 - A.9.2.4 -
Management of secret authentication information of users.

- HIPAA 164.308(a)(5)(ii)(D) -
Procedures for creating, changing, and safeguarding passwords.

How to detect and fix this issue

Auditing AWS accounts is one of the best ways to detect unused access keys. Follow the steps below in your AWS management console to detect and remove the unused access keys.

- Log in to the AWS console. Select **Users** under the IAM service option from the left-hand side pane.
- Click the username you want to audit.
- In the consequent panel, click the **Security Credentials** tab.
- Investigate the **Last Used** column, and ensure that it doesn't read N/A. If it says N/A, it means that the access key has never been used by the selected user and is hence an unused key.

You'll have to repeat this procedure for every suspected user account, and check if the access key is unused or used. Alternatively, you can also find the unused access keys by downloading the credentials report and executing specific commands in the command-line interface (CLI). After getting the list of unused access keys, you can go ahead and delete it from the same console.

Mistake 4

Providing unrestricted access to Redshift cluster

A Redshift data warehouse is a collection of computing resources called nodes, which are organized into a group called a cluster. Each cluster runs an Amazon Redshift engine and contains one or more databases.

When you first provision an Amazon Redshift cluster, nobody has access to it by default. To grant other users access to this cluster, you need to associate it with a security group and define rules for allowing access.

Note:

If you are on the EC2-VPC platform, you can either associate an existing Virtual Private Cloud (VPC) security group or define a new one. If you're on the EC2-Classic platform, you need to define the security group and associate it with a cluster.

If you do not configure the security groups associated with the cluster properly, these Redshift clusters will be made publicly accessible. Anyone on the internet can then establish a connection to your database, increasing the security risk of brute-force attacks, SQL injections, or DoS attacks.

Bonus tip:

To secure the Redshift cluster traffic further, enable encryption for all the traffic on the wire. To do this, you must have the ***require_ssl*** parameter enabled.

How to fix it

Do not configure the security group open to a broad range of IP addresses. Provide inbound access to specific IP ranges, and also expose only the ports that are needed.

Mistake 5

Overly permissive access to cloud resources

We would never allow any direct connections to our network devices through the internet without a firewall. However, when it comes to the cloud, sometimes administrators unintentionally miss out on configuring inbound access rules; this provides direct access of sensitive resources through the internet. Below are some examples of overly permissive access provided to cloud resources and components that could lead to a cloud security disaster.

- **Kubernetes cluster being exposed to the internet:**

  Over the last couple of years, the usage of Kubernetes has increased rapidly, as many cloud companies have adopted it as the default way to orchestrate and scale their container-based workloads. A recent analysis shows that etcd services, an open source distributed key-value store used to hold and manage the critical information that distributed systems need to keep running, have shown up on the internet without proper authentication.

According to this research, a simple search on Shodan revealed nearly 2,284 etcd servers on the internet, highlighting that this is a problem to look out for while setting up your clusters. Ensure that the etcd port 2379 for Kubernetes cluster is not exposed to the internet.

How to fix it

Constantly audit and review the VPC security groups and network access control lists (NACLs) that guard the inbound and outbound traffic to resources. Track changes to these groups and get alerted when the above protocols and ports are configured for unrestricted access.

Bonus tips:

Apart from removing the unused user access keys, ensure that you take the below security steps as well:

- **It's not just the etcd:** Depending on your cluster's configuration, other services might also pose vulnerabilities that lead to a security incident. Take for instance the Kubelet API, used by Kubernetes to manage containers on individual nodes or in some clusters; if this API is available unauthenticated, it could lead to a Kubelet exploit. Hackers could leverage this opportunity to execute code in your containers, and even take over your entire cluster, so you must always have an authentication gateway for Kubelet API.

- **A compromised container is equal to a compromised cluster:** As you know, a Kubernetes cluster hosts a set of application containers, so it's important to ensure the security of each and every application running on the cluster to avoid a full-blown cluster compromise.
  If role-based access control (RBAC) isn't configured for these clusters, attackers who gain access to a single container in a cluster can easily escalate their privilege and gain full control of it. Therefore, it's always mandatory to configure RBAC for these clusters.
  If you really want to follow the default behavior wherein a token that provides access to the Kubernetes API is mounted onto each container, ensure that you've restricted the rights the user has on the other cluster resources.

- **Unrestricted access to well-known ports:** Ensure that legacy ports and protocols are enabled on the cloud host without any restrictions. This will lead to unrestricted and easy access of these hosts by the malicious actor over the internet. Ensure access to required entities for:

  - **Common Internet File System (CIFS)** to avoid unauthorized data access.

  - **File Transfer Protocol (FTP),** which accesses through port 20/21 to avoid unauthorized data access and accidental data breach.

  - **Internet Control Message Protocol (ICMP)** to avoid unauthorized data access, rogue scanning of network vulnerabilities, or DoS attacks against the cloud infrastructure.

- **Port 27017 (MongoDB), port 1433 (MSSQL), port 3306 (MySQL), port 1521 (Oracle DB), and port 5432 (PostgreSQL)** to avoid accidental data breach, and unauthorized data access and leaks.
- **Remote Desktop Protocol (RDP)** accessed through **port 3389** to avoid unauthorized resource access and security breaches.
- **Remote procedure call (RPC)** accessed through **port 135** and **SMTP through port 25** to avoid data leaks.
- **Secure Shell (SSH)** through **port 22** and **Telnet** through **port 23** to avoid data leaks.

About Log360

The need of the hour is a solution that monitors, audits, and helps you manage your AWS cloud infrastructure. Introducing Log360, a comprehensive security and information event management (SIEM) solution that provides you:

- A central console that collects, analyzes, and monitors the log data from CloudTrail and S3 bucket to give complete visibility into activities happening in your AWS environment.
- Exhaustive information on user activities in your AWS.
- Real-time notifications and detailed reports on the changes happening to VPC security groups, subnet changes, and more.
- In-depth reports on the configuration changes to critical resources such as ELB, RDS, EC2, and more to spot unauthorized changes instantly and stop data leaks.
- Details on the changes to files stored on the S3 bucket to ensure integrity of the files stored on these resources.
- Exhaustive S3 and ELB traffic details so you can gain visibility into who is accessing what resources in your AWS environment.

$ Get Quote   Download   Schedule a demo