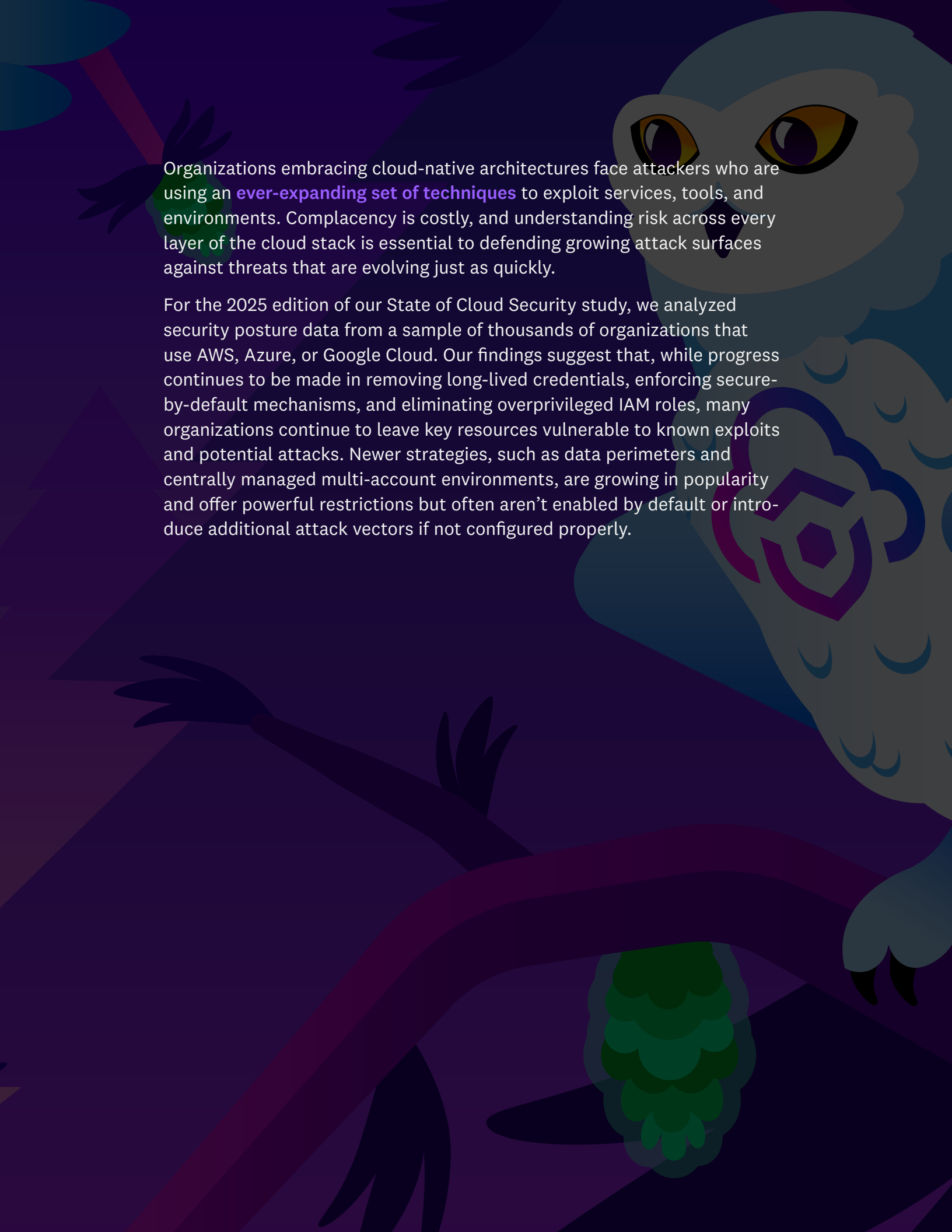
# STATE OF CLOUD SECURITY

DATADOG



Organizations embracing cloud-native architectures face attackers who are using an **ever-expanding set of techniques** to exploit services, tools, and environments. Complacency is costly, and understanding risk across every layer of the cloud stack is essential to defending growing attack surfaces against threats that are evolving just as quickly.

For the 2025 edition of our State of Cloud Security study, we analyzed security posture data from a sample of thousands of organizations that use AWS, Azure, or Google Cloud. Our findings suggest that, while progress continues to be made in removing long-lived credentials, enforcing secure-by-default mechanisms, and eliminating overprivileged IAM roles, many organizations continue to leave key resources vulnerable to known exploits and potential attacks. Newer strategies, such as data perimeters and centrally managed multi-account environments, are growing in popularity and offer powerful restrictions but often aren't enabled by default or introduce additional attack vectors if not configured properly.

FACT 1

# Multi-account environments allow organizational guardrails in AWS

Enforcing minimal privileges in a single AWS account is challenging. Similarly, running workloads from the different stages (dev, prod) in the same AWS account can be difficult due to quota, IAM, network, and performance issues. This is why a common best practice, both from a security and operational perspective, is to have multiple AWS accounts centrally managed through AWS Organizations, a governance service offered by AWS. Note that an "AWS organization" refers to one grouping of AWS accounts within this service and is not the same as an "organization."

We found that at least **84% of organizations** use more than one AWS account, while **86% of organizations** have AWS accounts using AWS Organizations. AWS Organizations has become a popular way to centrally manage multi-account environments, with over **two in three organizations (70%)** having all their AWS accounts part of an AWS organization and **three in four AWS accounts (74%)** part of an AWS organization. A meaningful minority of organizations (14%) do not use AWS Organizations at all for any of their AWS accounts.

When using multi-account environments managed through AWS Organizations, it's possible to enforce security invariants across all AWS accounts with organization-level guardrails. We found that among organizations using AWS Organizations, **two in five (40%)** take advantage of service control policies (SCPs), and 6% use resource control policies (RCPs)—a recently introduced type of organizational policy that applies at the resource level.

By analyzing the IAM condition keys in use in the explicit deny statements of these policies, we saw that SCPs are most commonly used to protect shared infrastructure and "landing zones" deployed in member accounts, as well as security mechanisms like CloudTrail and GuardDuty. When RCPs are defined, they are most commonly used to enforce encryption across all S3 buckets in an organization and ensure that all buckets in an AWS organization can only be accessed by principals from the same AWS organization.

However, using AWS Organizations comes with a few things to watch out for. By design, authorized principals in the organization management account can access all child accounts, creating an extremely powerful lateral movement vector. To protect against this, AWS recommends that the management account should not run any workloads, as doing so reduces the risk of it getting compromised. Still, we identified that 9% of organizations run EC2 instances in their AWS organization management account, up from 6% in 2024. In these configurations, an attacker who's able to compromise a single EC2 instance may be able to escalate their privileges and access all child AWS accounts.

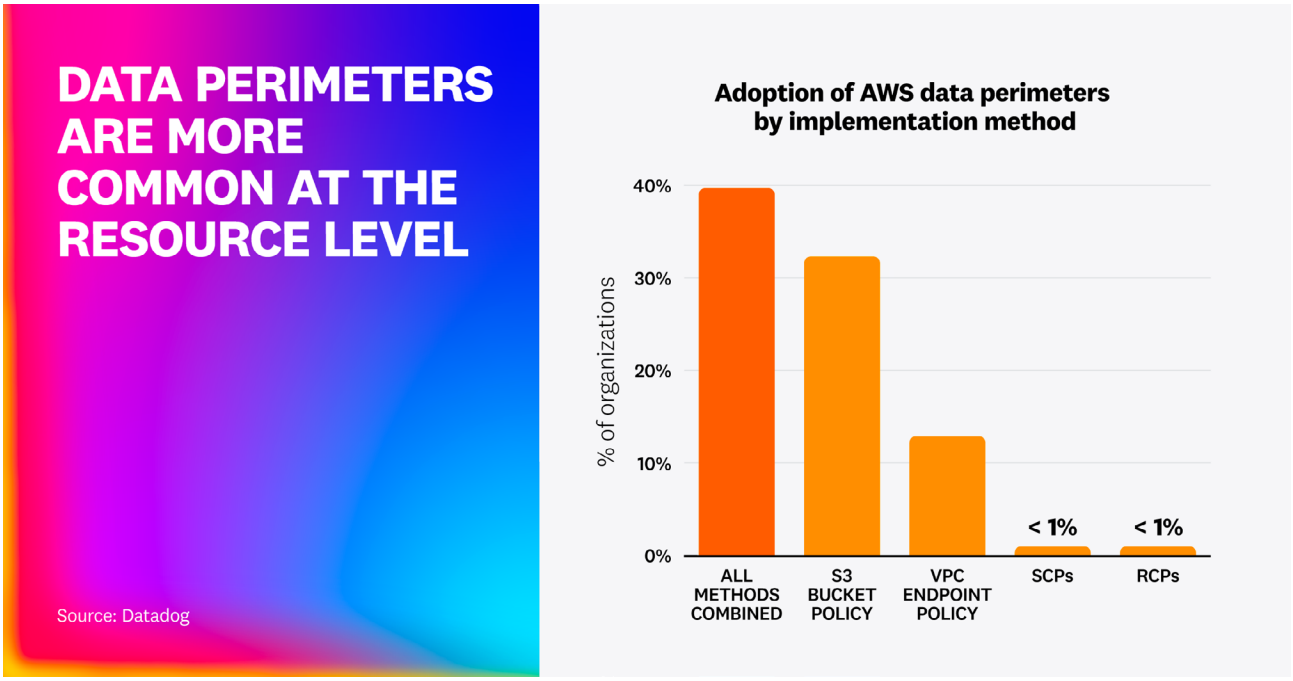
FACT 2

# As organizations adopt data perimeters, most lack centralized control

In the cloud, identity is the new perimeter. There is little concept of an internal network, since cloud APIs are exposed to the internet by design. This means that an attacker stealing cloud credentials can call cloud APIs from anywhere in the world without the need to be in an internal network. With credential theft being a major attack vector, the concept of data perimeters has emerged in recent years. Data perimeters enable teams to restrict certain cloud API calls so they only succeed if coming from approved networks or trusted cloud accounts.

However, data perimeters are not implemented by default. They require enacting a set of various policies such as SCPs, RCPs, or resource-based policies—for example, at the S3 bucket or VPC endpoint level. Using a data perimeter requires using advanced IAM policy condition keys such as `aws:SourceAccount` or `aws:PrincipalOrgID`.

We determined that almost **two in five (40%)** organizations use data perimeters through either SCPs, RCPs, VPC endpoints, or S3 bucket policies.



Overall, the most popular ways to implement data perimeters are through:

- S3 bucket policies (32% of organizations), mostly through the `aws:SourceAccount` condition key. This ensures that nobody outside of a specific AWS account can access the bucket, even if its ACLs or individual files are mistakenly made public.
- VPC endpoint policies (13% of organizations), mostly through the `aws:PrincipalAccount` condition key. This ensures that an attacker cannot exfiltrate data to an attacker-owned AWS account.
- SCPs (0.6% of organizations), mostly through the `aws:PrincipalAccount` condition key.
- RCPs (0.1% of organizations), mostly through the `aws:PrincipalOrgID`, `aws:SourceOrgID`, and `aws:SourceAccount` condition keys. These allow defenders to make sure that a specific type of resource can't be accessed by anyone outside the organization, even if it's misconfigured.

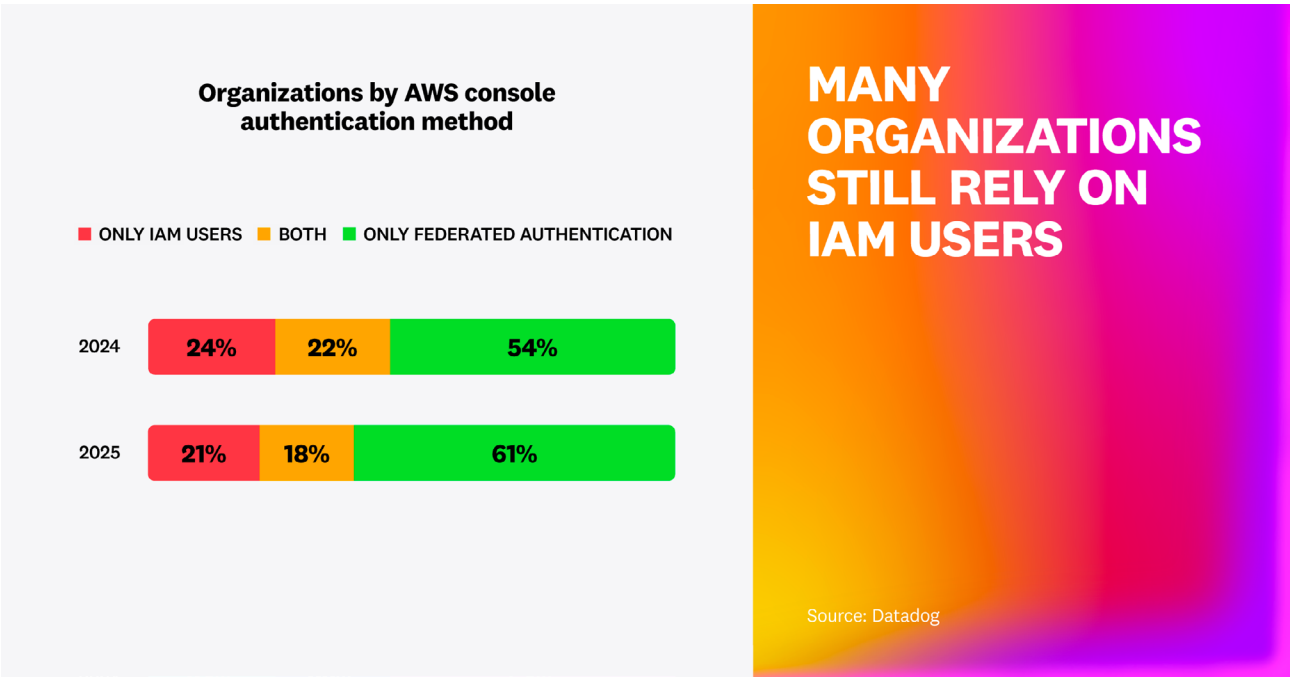
Although data perimeters are considered an advanced practice, over a third of organizations already use them. This reflects growing concern over credential theft and a willingness to adopt provider-supported safeguards for cloud data. But our analysis also reveals that most data perimeters are still applied per resource. Many organizations beginning to use data perimeters should shift toward modern controls like RCPs, which enforce guardrails at the organizational level, eliminating gaps in coverage.public. Although there are legitimate use cases for public storage buckets, common uses such as static web assets should typically use a content delivery network (CDN) such as Amazon CloudFront, as this is usually a more efficient and cheaper solution.

FACT 3

# Long-lived cloud credentials remain pervasive and risky

Long-lived cloud credentials continue to pose a major security risk, with previous installments of this report having shown that they are the most common cause of publicly documented cloud security breaches. For this year's findings, we followed up on how organizations are using legacy versus modern authentication methods to authenticate humans and applications across AWS, Azure, and Google Cloud.

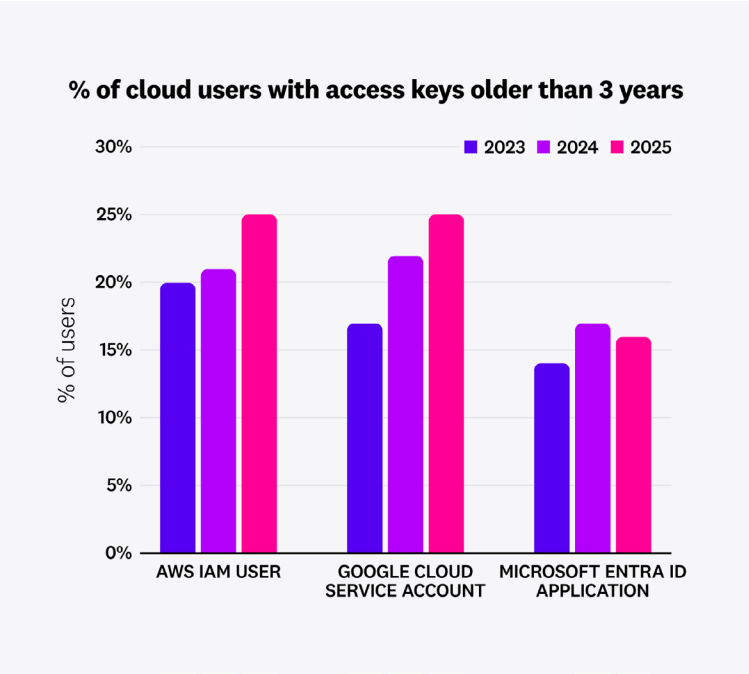
We found that most organizations **(79%)** continue to use federated authentication for human users to access the AWS console, such as through AWS IAM Identity Center or Okta. This represents a small increase in comparison to 2024, **up from 76%**, indicating that some organizations have transitioned fully to only federated authentication. However, almost **two in five (39%)** organizations still use IAM users in some capacity, and **one in five** organizations relies exclusively on IAM users. This shows that although organizations are increasingly adopting central human identities to access their cloud environments, they are struggling to get rid of their legacy IAM practices, such as humans using IAM users. It's easier to add a new secure authentication practice than to get rid of a risky legacy one, but to ensure proactive security and mitigate risks, it's critical to not only implement new solutions but also manage tech debt.



Similar to last year, we found that long-lived cloud credentials are often old and even unused, which increases the risks associated with these types of credentials.

- In AWS, **59%** of IAM users have an active access key older than one year. **Over half** of these users have credentials that have been unused for over 90 days, hinting at risky credentials that could easily be removed. Crucially, this number hasn't decreased in comparison to 2024, suggesting that organizations have plateaued in improving their security posture in this area.
- Google Cloud service accounts follow a similar pattern but have greatly improved year over year: **55%** of Google Cloud service accounts have active service account keys older than a year, down from 62% a year ago. We attribute this not to a decrease in the number of service accounts (which can be used to generate short-lived credentials, such as with service account impersonation or workload identity federation), but to a decrease in the number of service account keys for these service accounts.
- Comparably, **40%** of Microsoft Entra ID applications have credentials older than one year, down from 46% a year ago.

Analyzing access key age over time also shows that organizations using long-lived credentials are struggling to rotate them. The percentage of access keys older than three years has **mostly increased** across all clouds (AWS 21% → 25%, Google Cloud 22% → 25%, Azure stable)—same for five years (AWS 8% → 10%, Google Cloud 6% → 8%, Azure 6% → 10%). This increase is concerning, because as long as they exist and continue to grow stale, these access keys become increasingly risky. The only scalable response is to stop using them and switch to secure alternatives.




## RISKY LONG-LIVED CLOUD CREDENTIALS ARE STILL COMMON

Source: Datadog

Instead of long-lived credentials, organizations should use mechanisms that provide time-bound, temporary credentials. For workloads, this can be achieved with IAM roles for EC2 instances or EKS Pod Identity in AWS, managed identities in Azure, and service accounts attached to workloads for Google Cloud. For humans, the most effective solution is to centralize identity management by using a solution like AWS IAM Identity Center, Okta, or Microsoft Entra ID and to avoid the use of an individual cloud user for each employee, which is highly inefficient and risky.

*"From an incident response perspective, exposed access keys belonging to IAM users remain the primary initial access vector in AWS. Previously, we observed access key exposure more in scripts that were accidentally exposed through various code sharing platforms. In more recent incidents, the keys were exposed through improperly protected CI/CD pipelines and shared developer workstations."*

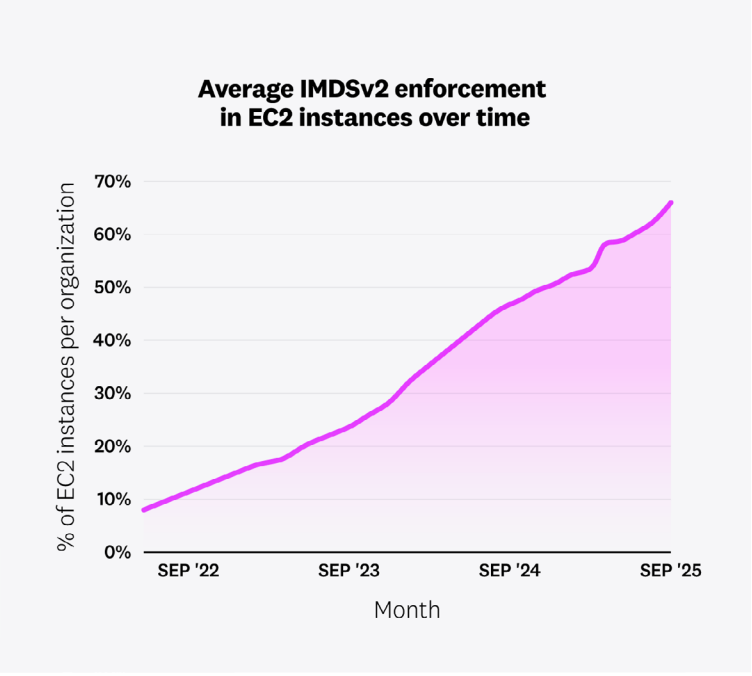
**Korstiaan Stam**
**Founder and CEO, Invictus Incident Response**

## FACT 4

# One in two EC2 instances enforces IMDSv2, but older instances lag behind

Adoption of IMDSv2—an AWS security mechanism that blocks credential theft in EC2 instances—continues to grow, largely driven by AWS's move toward secure-by-default mechanisms that enable IMDSv2 automatically. Historically, IMDSv2 had to be manually enforced on individual EC2 instances or autoscaling groups.

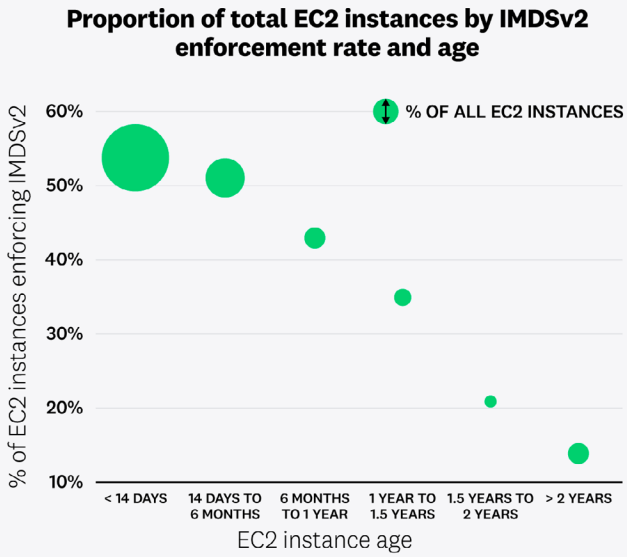
On average, organizations now enforce IMDSv2 on **66%** of their instances, up from 47% a year ago and 25% two years ago. Overall, **49% of EC2 instances** have IMDSv2 enforced, up from 32% a year ago, 21% two years ago, and 7% in 2022.

While this reinforces an ongoing trend, we also found that not all EC2 instances are created equal. Although **over half of instances** created in the two weeks preceding the collection of the data for this report enforce IMDSv2 (representing two-thirds of overall EC2 instances), only **14%** of instances created over two years ago enforce it.

## OLDER EC2 INSTANCES LAG IN ENFORCING IMDSv2

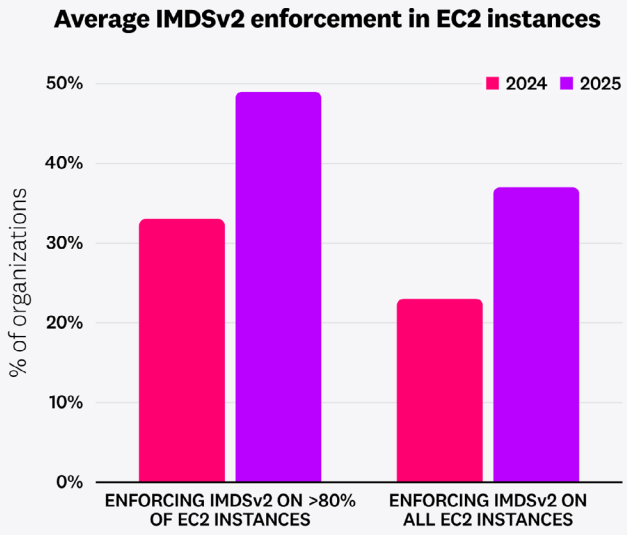
Source: Datadog




We also identified that more organizations are taking IMDSv2 seriously. Over **one in three organizations (37%)** enforces IMDSv2 on 100% of their EC2 instances, up from less than one in four in 2024. One in two enforces it on at least 80% of their instances (up from one in three), and only 24% enforce it on fewer than 20% of their EC2 instances—down from **39%**.

## MORE ORGANIZATIONS ENFORCE IMDSv2 ON MOST EC2 INSTANCES

Source: Datadog



Similar to last year's data, while IMDSv2 enforcement is increasing, most instances still remain vulnerable. However, even when IMDSv2 is not enforced on an EC2 instance, individual applications and processes on that instance can use it. By looking at CloudTrail logs, we identified that although only 49% of EC2 instances have IMDSv2 enforced, **82%** had exclusively used IMDSv2 in the past two weeks, meaning they could have enforced it with no functional impact. This data shows that there continues to be a disconnect between enforcement of IMDSv2 and actual usage, particularly for older EC2 instances that didn't enable IMDSv2 by default. When mechanisms like IMDSv2 aren't enabled by default, security settings fall behind even if they have no functional impact or don't block developers.

Organizations looking to enforce IMDSv2 should implement secure-by-default mechanisms such as using the IMDSv2-by-default on AMIs and enforcing IMDSv2 by default on new instances on a per-region basis, as introduced in March 2024. We found that while fewer than **3%** of organizations use this setting, when it is enabled, IMDSv2 enforcement reaches levels of **95%+**. In most cases, security by default is a simple configuration change that can have an enormous impact on closing vulnerability gaps.

*"While IMDSv2 adoption has been increasing year over year, enforcement still lags despite AWS introducing configurable defaults. Organizations should enable these security defaults and enforce IMDSv2 through SCPs across their environments. IMDSv2 is a simple, effective control that significantly reduces the impact of SSRF vulnerabilities. As long as IMDSv1 remains available, attackers will continue to exploit it, and the only way to close this gap is for AWS to deprecate IMDSv1."*

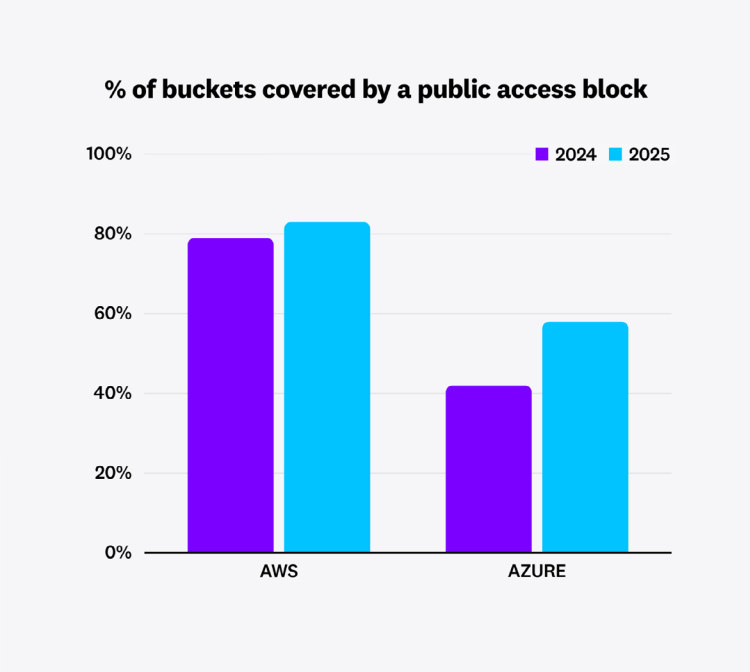
**Adam Pietrzycki**
**Senior Infrastructure Security Engineer, Zapier**

FACT 5

# Adoption of public access blocks in cloud storage services is plateauing

Paved roads and guardrails are two security practices that make it easy to standardize processes and prevent human mistakes from turning into data breaches, respectively. Because public storage buckets continue to be the source of a large number of high-profile data breaches, many modern controls—which fall into the category of guardrails—exist to ensure that a bucket cannot be made public by mistake.

In this year's report, we found that adoption of public access blocks continues to increase but at a slower pace than previous years. **One percent of S3 buckets** are effectively public, down from **1.5%** in 2024 and 2023. Over four in five (83%) S3 buckets are covered by an account-wide or bucket-specific S3 Block Public Access, a small increase from 79% a year ago and 73% in 2023. This increase is likely caused by continuous wider awareness of the issue, and the fact that AWS proactively blocks public access for newly created S3 buckets as of April 2023.




AWS LEADS IN BLOCKING PUBLIC ACCESS BY DEFAULT, AZURE SHOWS FASTER GROWTH

Source: Datadog

On the Azure side, **1.3%** of Azure Blob Storage containers are effectively public, down from 2.6% a year ago and 5% in 2023. Relatedly, almost **three in five (58%)** Azure Blob Storage containers are in a storage account that proactively blocks public access, up from 42% a year ago. This is likely due to Microsoft proactively blocking public access on storage accounts that were created after November 2023, making them secure by default.

To avoid mistakenly exposing S3 buckets, organizations should turn on S3 Block Public Access at the account level and protect the configuration with an SCP. At the organization level, it's possible to deploy an RCP to apply a data perimeter to all S3 buckets in an AWS organization and ensure that they cannot be accessed externally from unauthenticated actors, even if the bucket policy or ACL of that bucket is misconfigured. In Azure, you can block public access in a storage account configuration, which prevents Blob Storage containers in that storage account from inadvertently becoming public.

For most common, legitimate use cases for public storage buckets, such as static web assets, it's typically more secure, efficient, and cost-effective to use a content delivery network (CDN) such as Amazon CloudFront.

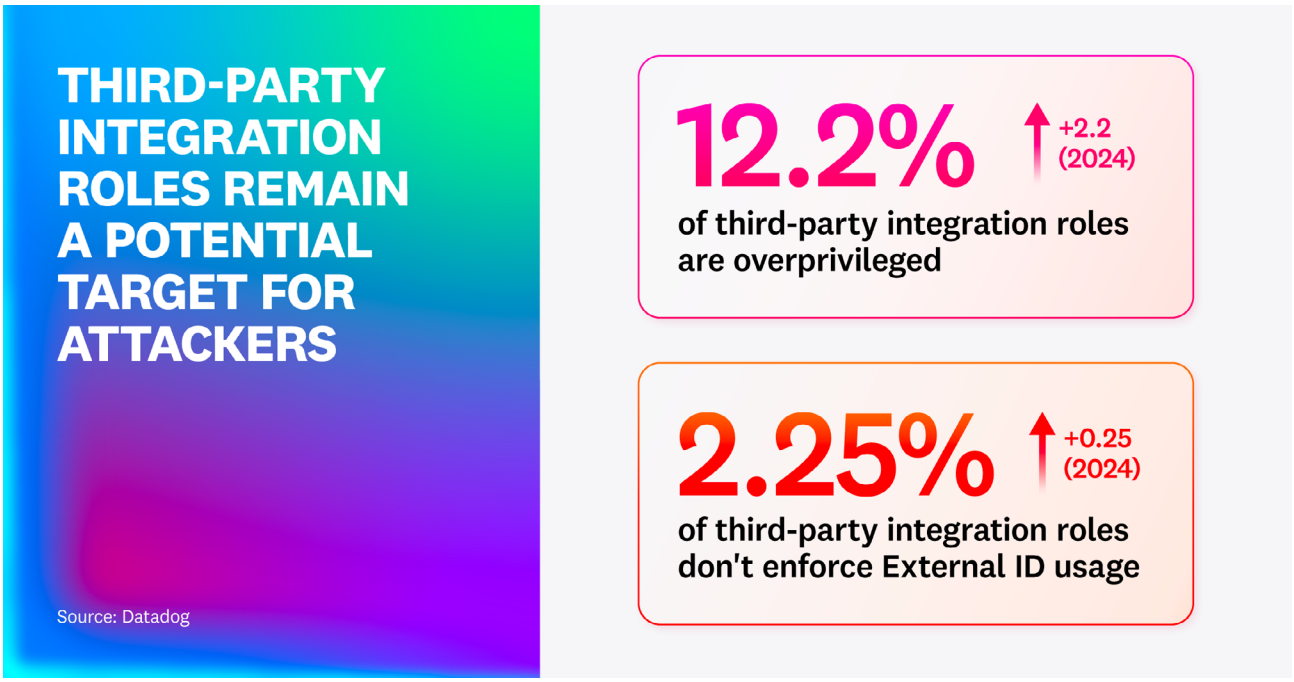
## FACT 6

# Overprivileged and misconfigured IAM roles for third-party integrations remain risks for AWS accounts

With more vendors integrating with their customers' AWS accounts through IAM roles, a compromised provider AWS account increases cloud supply chain risks, as it's likely that an attacker can access the same data as the provider can.

For this fact, we reviewed IAM roles trusting known AWS accounts that correspond to SaaS providers. We found that on average, an organization deploys 13 third-party integration roles (up from 10.2 in 2024), linked to an average of 2.5 distinct vendors (same as 2024).

We then looked at two types of common security risks in these integration roles. We found that **12.2% of third-party integrations** are dangerously overprivileged, allowing the vendor to access all data in the account or to take over the whole AWS account. This is slightly up from 10% in 2024. We also identified that 2.25% of third-party integration roles don't enforce the use of an external ID. This indicates that third-party risks are as prevalent as ever in cloud environments, and organizations should continue to remove unused roles and grant minimum permissions.

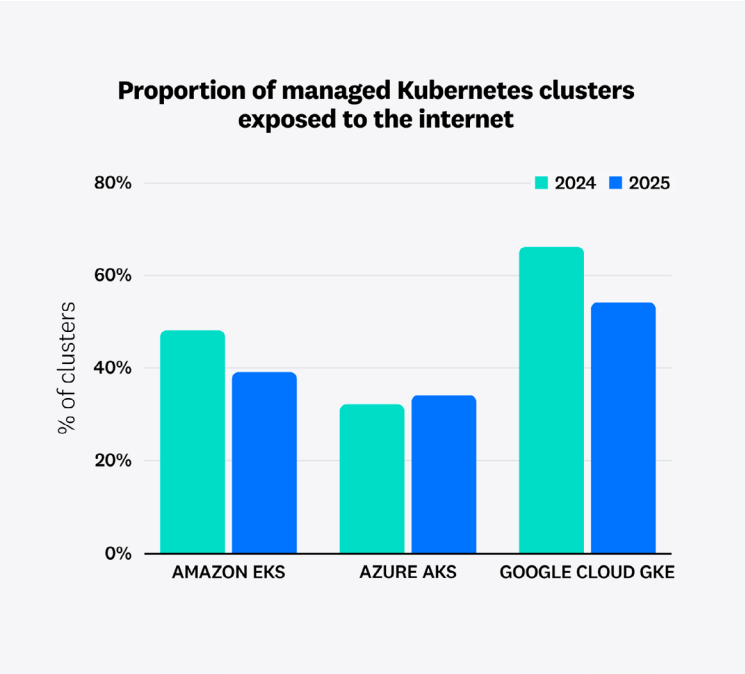


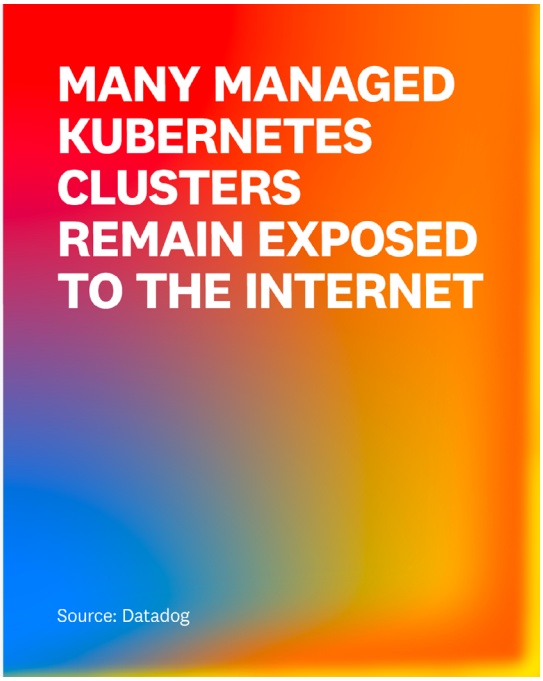
## FACT 7

# Insecure default configurations leave managed Kubernetes clusters at risk

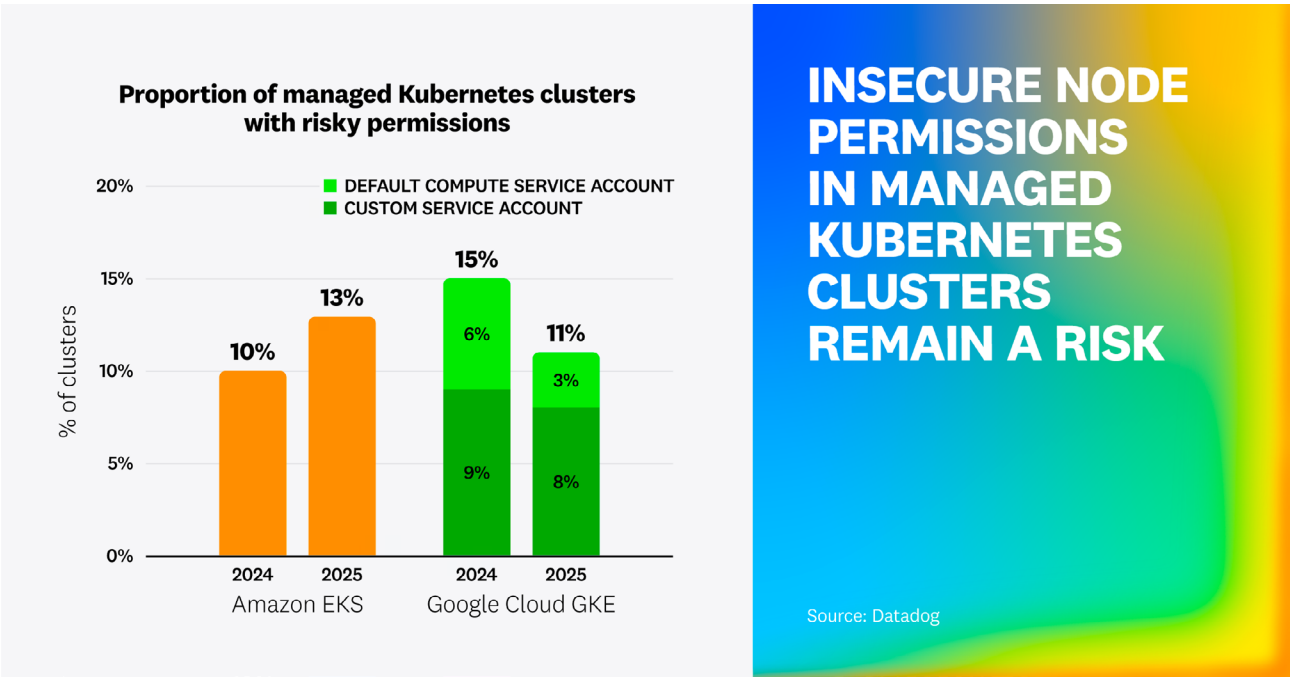
Popular managed Kubernetes services, such as Amazon EKS, Azure AKS, and Google Cloud GKE, allow teams to focus on application workloads instead of complex Kubernetes control plane components such as etcd. But their default configurations still often lack adequate security. This can be problematic, as these clusters are intrinsically running in cloud environments; compromising a managed cluster opens up a number of possibilities for an attacker to pivot to the underlying cloud infrastructure. Anything exposed to the internet is scanned and creates noise, which can divert defenders from genuine threats. These internet-facing clusters also provide additional entry points for attackers to further abuse identities and escalate control.

First, we found that the proportion of managed Kubernetes clusters exposing their managed API server to the internet has decreased in comparison to last year, but remains relatively high. **Two in five (39%)** EKS clusters and **one in three (34%)** AKS clusters are exposed to the internet. This number is much higher for GKE clusters, reaching over **one in two (54%)**. Interestingly, we confirmed for EKS and GKE that the chance of internet exposure was unrelated to the age of the cluster, meaning that newer clusters aren't getting less exposed than older ones.



We also analyzed the IAM roles attached to these managed Kubernetes clusters to understand the likelihood that an attacker compromising them would be able to pivot to the cloud environment. For EKS, we determined that **13%** of clusters have a dangerous node role that has full administrator access, allows for privilege escalation, has overly permissive data access (e.g., all S3 buckets), or allows for lateral movement across all workloads in the account. In Google Cloud, **10%** of GKE clusters have a privileged service account: 3% through the use of default compute service accounts with the unrestricted `cloud-platform` scope, and 8% through customer-managed Google Cloud service accounts.




Although public exposure of managed Kubernetes clusters is decreasing, a large portion of these clusters is still exposed to the internet. This creates noise (automated attacks and scans), initial access risks (identity theft and exploitation), and pivot risks (if a cluster is compromised, cloud environments are at risk, too).

Organizations are also becoming more aware of the risks of attaching privileged cloud roles to cluster nodes, which lowers the risks that a compromised pod may be able to compromise other resources outside of the cluster. Risks related to default permissions had a noticeable change (down to 3% of GKE clusters), suggesting that organizations are becoming more aware of insecure-by-default configurations and taking action to fix them.

## Fact 8

# Organizations continue to run risky workloads

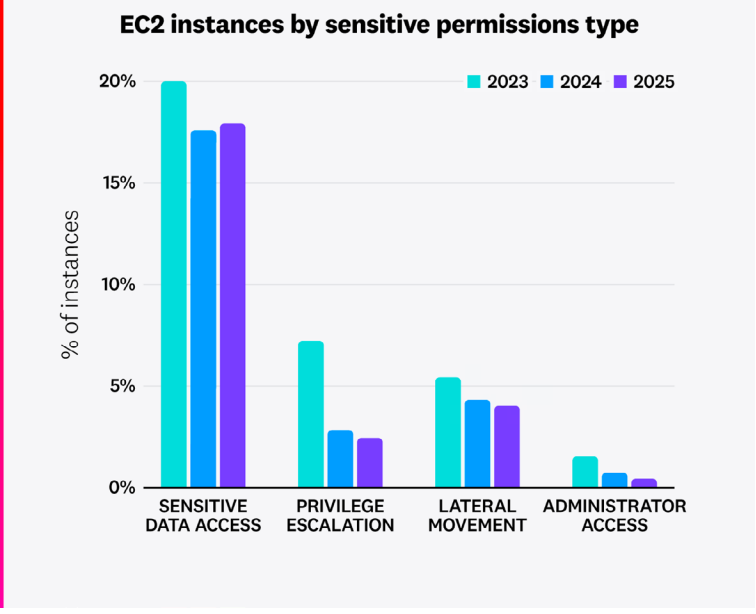
Assigning overprivileged permissions to EC2 instances or Google Compute Engine (GCE) virtual machines (VMs) continues to create substantial risk. Any attacker who compromises a workload—for instance, by exploiting an application-level vulnerability—would be able to steal the associated credentials and access the cloud environment.

In AWS, while fewer than 0.4% of EC2 instances have administrator privileges, almost **one in five (19.4%)** is overprivileged—stable from 19% in 2024. Among these:

- 3.8% have risky permissions that allow lateral movement in the account, such as connecting to other instances using SSM Sessions Manager (slighty down from 4.3%). Lateral movement can allow an attacker to expand their scope beyond a single piece of infrastructure.
- 2.4% allow an attacker to gain full administrative access to the account by privilege escalation, such as permissions to create a new IAM user with administrator privileges (slightly down from 2.8%).
- 17.9% have excessive data access, such as listing and accessing data from all S3 buckets in the account (slightly up from 17.6%).

**NEARLY 1 IN 5 EC2 INSTANCES HAVE EXCESSIVE PERMISSIONS**

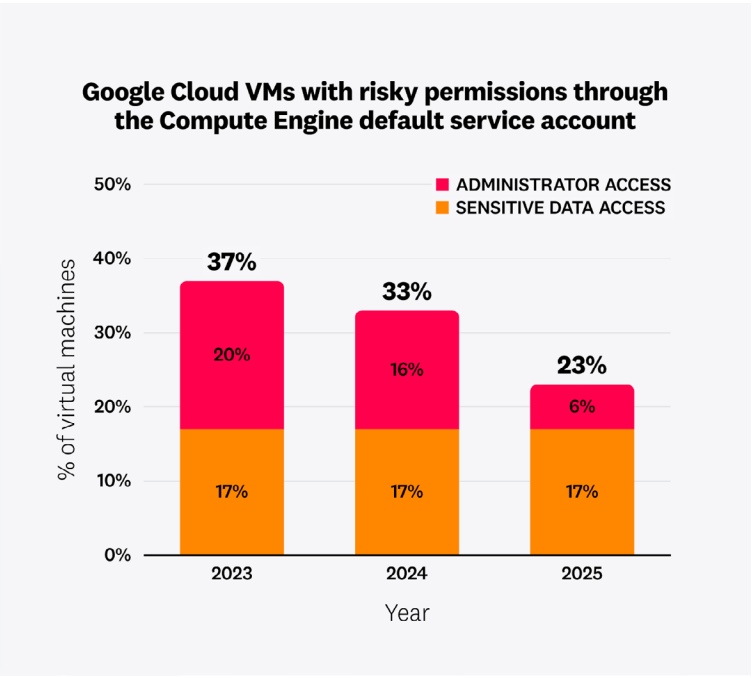
Source: Datadog



In Google Cloud, we found that **17%** of VMs have full administrator "editor" permissions on the project they run in through the use of the Compute Engine default service account with the unrestricted `cloud-platform` scope. In addition, another 6% have full read access to Google Cloud Storage (GCS) through the same mechanism. In total, nearly one in four Google Cloud VMs (23%) has overly permissive access to a project.

**MANY GOOGLE CLOUD VMs STILL RUN WITH OVERLY PERMISSIVE PRIVILEGES**

Source: Datadog




Compared to 2024, the overall number of privileged Google Cloud VMs has decreased. Notably, the proportion of VMs using an unrestricted scope was divided by three, while the percentage of privileged VMs attributable to the Compute Engine default service account remains unchanged.

# Methodology

Findings are based on data collected in September 2025.

## Population

For this report, we analyzed the cloud security posture of a sample of thousands of organizations. Data in this report has come from customers of Datadog Infrastructure Monitoring, Datadog Logs, and Datadog Cloud Security.

## Fact 1

For this fact, we analyzed how many AWS accounts each organization has. We take into account both when AWS accounts are integrated individually, and integrated at the AWS Organization level.

When measuring adoption of non-default service control policies (SCPs) and resource control policies (RCPs), we can only analyze organizations that integrate their AWS Organizations management account, and we reflect this in the denominator of the ratios used.

## Fact 2

We analyzed SCPs, RCPs, VPC endpoint policies, and S3 bucket policies and considered that they were using an AWS data perimeter when they contained any of the following AWS IAM condition keys in any of the statements:

## Fact 3

In the graph that depicts average enforcement of IMDSv2, we computed for each organization the percentage of its EC2 instances where IMDSv2 is enforced and averaged this number across all organizations. We used this method so as not to overrepresent organizations that have a large number of EC2 instances and instead to measure adoption trends.

In the graph that depicts average enforcement of IMDSv2, we computed for each organization the percentage of its EC2 instances where IMDSv2 is enforced and averaged this number across all organizations. We used this method so as not to overrepresent organizations that have a large number of EC2 instances and instead to measure adoption trends.

## Fact 4

**Internet exposure**

We determined that an Amazon EKS cluster was exposed to the internet if both of the following were true:

`aws:PrincipalAccount`
`aws:PrincipalOrgID`
`aws:PrincipalOrgPaths`
`aws:ResourceAccount`
`aws:ResourceOrgID`

`aws:ResourceOrgPaths`
`aws:SourceAccount`
`aws:SourceIp`
`aws:SourceOrgID`
`aws:SourceOrgPath`
`aws:SourceVpc`
`aws:SourceVpce`

For this edition, we did not take into account the condition keys `aws:VpceAccount`, `aws:VpceOrgPaths`, or `aws:VpceOrgID`, which were released a few days before the snapshot of the data in use.

The ratio of organizations using SCPs, RCPs, VPC endpoints, or S3 bucket policies to implement a data perimeter was computed over the number of total organizations having at least one such SCP, RCP, VPC endpoint, or S3 bucket.

## Fact 3

**AWS console authentication method**

To identify the method used to authenticate the AWS console, we used CloudTrail events with the event type `ConsoleLogin`. Specifically, we considered that the authentication was performed through an IAM user for all events matching:

```
source:cloudtrail
  AND @evt.name:ConsoleLogin
  AND @userIdentity.arn:*user/*
  AND NOT @userIdentity.arn:*:root*
```

We considered that the authentication was performed through a federated identity provider for all events matching:

```
source:cloudtrail AND (
(@evt.name:ConsoleLogin @userIdentity.arn:*assumed-role/*)
 OR  (source:cloudtrail @evt.name:ConsoleLogin
@userIdentity.arn:*federated-user/*)
)
```

**Long-lived credentials**

For AWS, we considered IAM users that have at least one active access key. When an IAM user had several active access keys, we considered only the oldest one.

For Google Cloud, we considered service accounts that are not disabled and have at least one active access key. We excluded from our analysis Google-managed service accounts, for which it's not possible to create user-managed access keys.

For Microsoft Entra ID, we considered Microsoft Entra ID app registrations that had active "password" credentials (corresponding to static access keys that can be exchanged for OAuth 2.0 access tokens).

## Fact 4

In the graph that depicts average enforcement of IMDSv2, we computed for each organization the percentage of its EC2 instances where IMDSv2 is enforced, and averaged this number across all organizations. We used this method so as not to overrepresent organizations that have a large number of EC2 instances and instead to measure adoption trends.

In the bubble chart that depicts average enforcement of IMDSv2 per instance age, we computed for each organization the percentage of its EC2 instances where IMDSv2 is enforced and averaged this number across all EC2 instances within a specific age bucket. The area of each bubble is proportional to how many EC2 instances fall into this age bucket, independently of whether they enforce IMDSv2.

For historical data, we queried 14 days of CloudTrail logs and used the field `userIdentity.sessionContext.ec2RoleDelivery` to determine if IMDSv2 had been used to request initial session credentials. We only analyzed CloudTrail generated by EC2 instances, meaning where `userIdentity.session_name` starts with `i-`.

For data related to the IMDSv2-by-default regional feature, we analyzed the percentage of EC2 instances with IMDSv2 enforced by AWS account and region, and compared the rate depending on whether the "IMDSv2-by-default" feature was enabled in this region and AWS account.

## Fact 5

For AWS, we considered an S3 bucket public if both of the following were true:

- The bucket policy allows `s3:GetObject` on all objects in the bucket to a wildcard principal.
- The public access block configuration of neither the bucket nor the AWS account has `restrict_public_buckets` set to true.

We considered that an S3 bucket is "covered by a public access block" if the bucket public access block configuration or the AWS account public access block configuration has the four settings `block_public_acls`, `block_public_policy`, `ignore_public_acls`, and `restrict_public_buckets` set to true.

We excluded from the analysis S3 buckets that are made public through the static website feature, as this is a valid use case for public S3 buckets that is not necessarily covered by CloudFront, and also because it gives a strong signal that the bucket was explicitly meant to be public by design.

For Azure, we considered that an Azure blob storage container is publicly accessible if both of the following were true:

- Its `PublicAccess` field is set to `blob` or `container`.
- The storage account it resides in does not block public access—i.e., its `allowBlobPublicAccess` attribute is not set to false.

We considered that an Azure blob storage container is "covered by a public access block" if it's in a storage account that blocks public access—i.e., whose `allowBlobPublicAccess` attribute is not set to false.

## Fact 6

We considered that an IAM role is an integration role for a known SaaS vendor if its trust policy has a trust relationship with an AWS account from the list of known AWS accounts, with some additional tuning to add known vendors that aren't part of this list and consulting companies that are not applicable to the SaaS integration use case.

We considered that an IAM role does not enforce an external ID if its trust policy does not have any condition or if it has a condition that does not use the operators `StringEquals`, `StringLike`, or `ForAnyValue:StringEquals` on the condition key `sts:ExternalId`.

## Fact 7

**Internet exposure**

We determined that an Amazon EKS cluster was exposed to the internet if both of the following were true:

- Its `resources_vpc_config.endpoint_public_access` attribute is set to `true`.
- Its `resources_vpc_config.public_access_cidrs` attribute contains `0.0.0.0/0`.

We determined that a Google Cloud GKE cluster was exposed to the internet if all of the following were true:

- Its private endpoint is disabled (i.e., `private_cluster_config.enable_private_endpoint` was set to `false`).
- Its public endpoint is enabled (i.e., p`rivate_cluster_config.public_endpoint contained` a public IP).

Its network configuration allows internet traffic—i.e., it meets one of the following conditions:

- Its "authorized networks" are disabled (i.e., `master_authorized_networks_config.enabled` is false).
- Its "authorized networks" contain 0.0.0.0/0 (`master_authorized_networks_config.cidr_blocks`).
- Its "authorized networks" configuration explicitly allows all Google Cloud IPs (`gcp_public_cidrs_access_enabled`).

We determined that an Azure AKS was exposed to the internet if:

- It is not configured as a private cluster (i.e., `api_server_access_profile.enable_private_cluster` was set to `false`).

Its network configuration allows internet traffic—i.e., if one of the following was true:

- Its authorized IP ranges (`api_server_access_profile.authorized_ip_ranges`) contained `0.0.0.0/0.`
- It contained no specific IP range, meaning it did not limit ingress traffic.

**Node privileges**

We determined that an Amazon EKS cluster had nodes with a risky role using the same methodology as described in Fact 7. We considered that an EKS cluster was risky if at least one of its managed node groups had a risky node role.

For Google Cloud, we determined that a GKE cluster had nodes with a risky service account if one of these conditions were met:

- The service account associated with the cluster nodes is the Compute Engine default one, with an unrestricted `cloud-platform` scope, and this default service account is mapped to the `editor` or `owner` role at the project level.
- The service account associated with the cluster nodes is any service account mapped to the `editor` or `owner` role at the project level.

For Azure AKS, analyzing node permissions requires visibility on both Azure AKS and Entra ID service principals. As our dataset did not have a satisfying number of organizations with both data sources to be representative enough, we did not include Azure AKS in this analysis.

Following a change in methodology to make this number more accurate, we have edited the data for AKS we published in our 2024 edition down from 41% to 32%.

## Fact 8

For this fact, AWS data was sourced in August 2025 and October 2025.

For AWS, we considered that an EC2 instance has an administrative role if it has an instance role that's attached to either the `AdministratorAccess` AWS-managed policy or to a custom policy that has at least one statement allowing all actions on all resources, with no conditions.

We considered that an EC2 instance has "excessive data access" if one of the following conditions is true:

- Its instance role has any of the following combinations of permissions on the resource `*`, through an inline or attached policy:

`s3:listallmybuckets, s3:listbucket, s3:getobject`
`dynamodb:listtables, dynamodb:scan`
`dynamodb:listtables, dynamodb:exporttabletopointintime`
`ec2:describesnapshots, ec2:modifysnapshotattribute`
`ec2:describesnapshots, ebs:listsnapshotblocks, ebs:listchangedblocks, ebs:getsnapshotblock`
`rds:describedbclustersnapshots, rds:modifydbclustersnapshotattribute`
`rds:describedbsnapshots, rds:modifydbsnapshotattribute`
`secretsmanager:listsecrets, secretsmanager:getsecretvalue`
`ssm:describeparameters, ssm:getparameter`
`ssm:describeparameters, ssm:getparameters`
`secretsmanager:getparametersbypath`

- Its instance role is attached to one of the following AWS managed policies (which all meet the criteria above):

`AdministratorAccess`
`AdministratorAccess-Amplify`
`AmazonApplicationWizardFullaccess`
`AmazonDataZoneProjectRolePermissionsBoundary`
`AmazonDocDBConsoleFullAccess`
`AmazonDocDBElasticFullAccess`
`AmazonDocDBFullAccess`
`AmazonDynamoDBFullAccess`
`AmazonDynamoDBFullAccesswithDataPipeline`
`AmazonDynamoDBReadOnlyAccess`
`AmazonEC2FullAccess`
`AmazonEC2RoleforDataPipelineRole`
`AmazonElasticMapReduceforEC2Role`
`AmazonElasticMapReduceFullAccess`
`AmazonElasticMapReduceReadOnlyAccess`
`AmazonElasticMapReduceRole`
`AmazonGrafanaRedshiftAccess`
`AmazonLaunchWizard_Fullaccess`
`AmazonLaunchWizardFullaccess`

`AmazonLaunchWizardFullAccessV2`
`AmazonMacieServiceRole`
`AmazonMacieServiceRolePolicy`
`AmazonRDSFullAccess`
`AmazonRedshiftFullAccess`
`AmazonS3FullAccess`
`AmazonS3ReadOnlyAccess`
`AmazonSageMakerFullAccess`
`AmazonSSMAutomationRole`
`AmazonSSMFullAccess`
`AmazonSSMReadOnlyAccess`
`AWSBackupServiceRolePolicyForBackup`
`AWSCloudTrailFullAccess`
`AWSCodePipelineReadOnlyAccess`
`AWSCodeStarServiceRole`
`AWSConfigRole`
`AWSDataExchangeFullAccess`
`AWSDataExchangeProviderFullAccess`
`AWSDataLifecycleManagerServiceRole`
`AWSDataPipelineRole`
`AWSElasticBeanstalkCustomPlatformforEC2Role`
`AWSElasticBeanstalkFullAccess`
`AWSElasticBeanstalkReadOnlyAccess`
`AWSIoTDeviceTesterForFreeRTOSFullAccess`
`AWSLambdaFullAccess`
`AWSLambdaReadOnlyAccess`
`AWSMarketplaceSellerFullAccess`
`AWSMarketplaceSellerProductsFullAccess`
`AWSOpsWorksRole`
`DatabaseAdministrator`
`DataScientist`
`NeptuneConsoleFullAccess`
`NeptuneFullAccess`
`ReadOnlyAccess`
`SecretsManagerReadWrite`
`ServerMigrationServiceRole`
`SystemAdministrator`
`VMImportExportRoleForAWSConnector`

We considered that an EC2 instance has “permissions allowing lateral movement” if one of the following conditions was true:

– Its instance role has any of the following combinations of permissions on the resource `*`, through an inline or attached policy:

`ec2-instance-connect:sendsshpublickey`
`ssm:startsession`
`ssm:sendcommand`
`ec2:getserialconsoleaccessstatus, ec2:describeinstances, ec2:describeinstancetypes, ec2-instance-connect:sendserialconsolesshpublickey`

– Its instance role is attached to one of the following AWS managed policies (which all meet the criteria above):

`AdministratorAccess`
`AmazonApplicationWizardFullaccess`

`AmazonLaunchWizardFullaccess`
`AmazonSSMAutomationRole`
`AmazonSSMFullAccess`
`AmazonSSMMaintenanceWindowRole`
`AmazonSSMServiceRolePolicy`
`AWSOpsWorksCMServiceRole`
`EC2InstanceConnect`

To identify if an EC2 instance was running in an organization's AWS Organization management account, we selected all Datadog customers who integrate with their AWS Organization management account and identified these where at least one EC2 instance was running in it.

For Google Cloud, we considered that:

- A VM has administrator privileges on the project when the default compute service account is used with the `cloud-platform` scope.
- A VM has read access to the project's cloud storage when the default compute service account is used with the `devstorage.read_only` scope.

We excluded VMs in a project where the default compute service account had not been granted editor or owner access. In other words, we did take into account the case where the recommended `iam.automaticIamGrantsForDefaultServiceAccounts` organization policy is in effect, "neutralizing" the default permissions of the default compute service account.

For Azure, analyzing virtual machine permissions requires visibility on both Azure compute and Entra ID service principals. As our dataset did not have a satisfying number of organizations with both data sources to be representative enough, we did not include Azure virtual machines in our analysis.

## Licensing


**datadog.com**