WHITE PAPER

Comparing CSP-Managed Machine Identities

VECTRA®

This paper compares the threat models of machine identities managed by the Cloud Service Providers (CSP-managed) across the three major cloud service providers (AWS, Google Cloud, and Microsoft), with a focus on the risks associated with impersonation and usage.

While all major clouds provide managed non-human identities, their differing architectures create unique security risks and non-portable controls, placing the burden of prevention on different parties in each ecosystem.

A Comparative Threat Model of CSP-Managed Machine Identities

Each Cloud Service Provider is tasked with solving the same fundamental challenge of identity systems: How can non-human, automated processes be empowered to access cloud resources? To solve this problem, each cloud made design choices that resulted in nuances in their respective threat models. Understanding the differences in risk between the machine identities helps organizations know where to implement controls and where classes of threats are mitigated by design.

The challenge of implementing non-human identities is interwoven with the 'On behalf of' problem. That is when identity systems are tasked with creating a secure mechanism for one identity to use the permissions of another, either directly or indirectly. In this article, we'll examine the risks associated with impersonation and direct usage of non-human identities.

Furthermore, security researchers should use this paper as a guide to identify weak points in machine identity usage, noting that these weak points differ between AWS, Google Cloud, and Microsoft.

Table of Contents

Machine Identities Overview

Machine Identities, sometimes referred to as Non-Human Identities (NHIs), can be cleaved into two distinct baskets with slightly varying architectures, use cases, and threat models.

	AWS	GCP	MSFT
Customer Managed Machine Identities	AWS Roles	Service Accounts	3rd Party App Registrations and Service Principals
CSP-Managed Machine Identities	AWS Service-Linked Roles	Service Agents	1st Party App Registrations and Service Principals

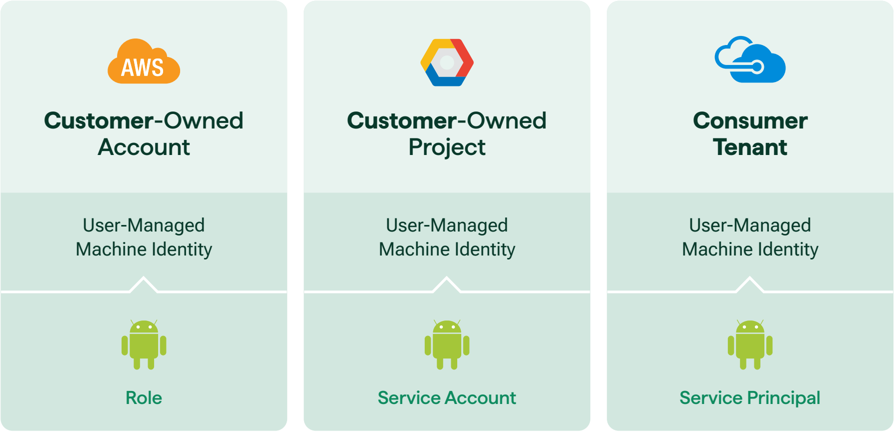
Research tends to focus on the risks associated with only one variety, the customer-managed machine identities, with little to no consideration given to their counterpart, the CSP-managed identities. This oversight likely occurs because the risks of customer-managed identities are more direct and easier to understand. The vulnerabilities that arise from CSP-managed identities, however, are more nuanced and often obscured by architectural complexities, leading to a misplaced assumption that they are secure by default.

To help close this gap, the following threat model compares only the CSP-managed machine identities between AWS, Google Cloud, and Microsoft.

Customer-managed Machine Identities

This article will not cover any of the characteristics or risks associated with the following Customer-managed Machine Identities.

- AWS: AWS Roles and IAM Users
- Google Cloud: User-Managed Service Accounts
- Microsoft: 3rd Party Application Registrations and Service Principals
  - Azure: Managed Identities



Customer-Owned Account
User-Managed Machine Identity
Role

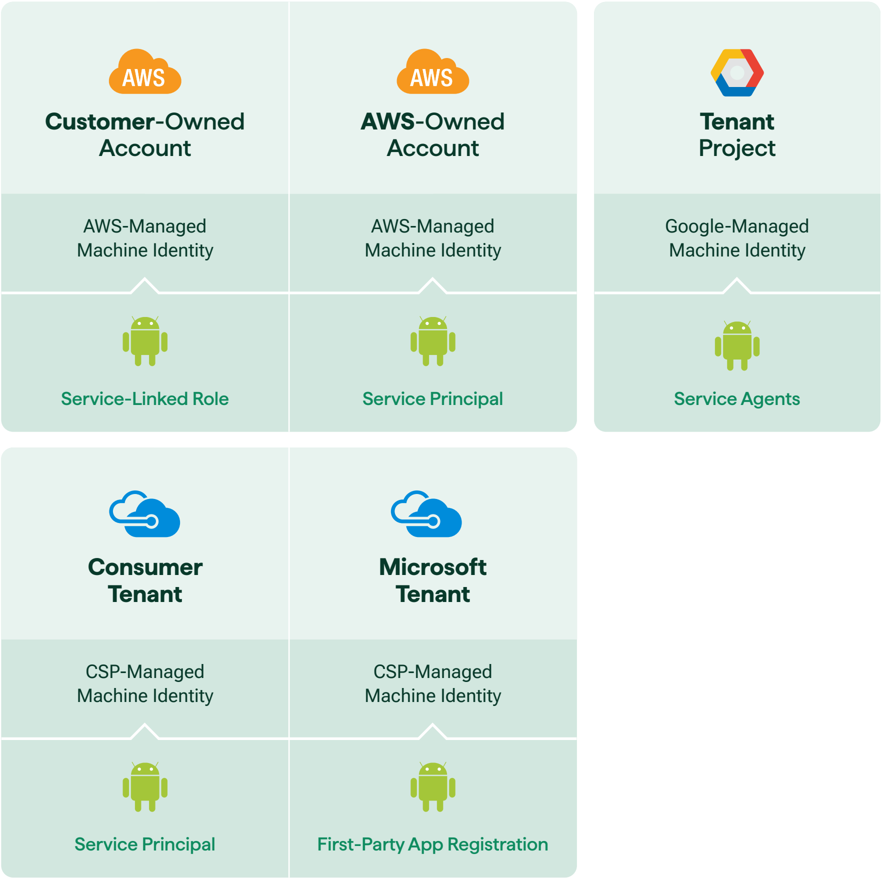
Customer-Owned Project
User-Managed Machine Identity
Service Account

Consumer Tenant
User-Managed Machine Identity
Service Principal

CSP-Managed Machine Identities

This article focuses entirely on the characteristics and risks associated with the following CSP-managed Machine Identities. Each has slight variations as to where the resources reside and, therefore, what manipulations are possible.

- AWS: AWS Service-Linked Roles and Service Principals
- Google Cloud: Google Service Agents (aka P4SAs)
- Microsoft: First-Party Application Registrations and Service Principals



Customer-Owned Account
AWS-Managed Machine Identity
Service-Linked Role

AWS-Owned Account
AWS-Managed Machine Identity
Service Principal

Tenant Project
Google-Managed Machine Identity
Service Agents

Consumer Tenant
CSP-Managed Machine Identity
Service Principal

Microsoft Tenant
CSP-Managed Machine Identity
First-Party App Registration

What Are We Working On?

Threat modeling, as described by Adam Shostack, involves four key framework questions: "What are we working on?", "What can go wrong?", "What are we going to do about it?" and "Did we do a good job?".

To answer the question, "What Are We Working On?", let's look at where each CSP-managed machine identity resides (where they are housed) and their relationship to customer tenants.

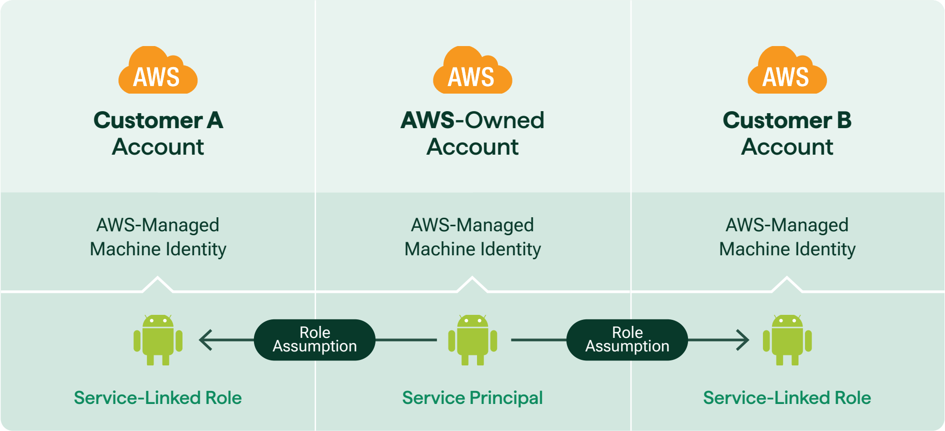
AWS Machine Identities

In AWS, CSP-managed machine identities are implemented through a combination of Service Principals and Service-Linked Roles. A Service Principal is a unique identifier for a global AWS service (e.g., ecs.amazonaws.com), while the Service-Linked Role resides within the customer's AWS account.

To perform actions on a customer's resources, an AWS Service Principal "assumes" the corresponding Service-Linked Role within that customer's account. These are specialized roles with unmodifiable trust policies, strictly defining which specific Service Principal is permitted to assume them.

Key characteristics of this model include:

1. **Multi-Tenancy:** The same global Service Principal for a particular AWS service is used to assume Service-Linked Roles across many different customer accounts.
2. **Hybrid CSP and Customer-Managed (with Safeguards):** Although the Service-Linked Role exists within the customer's account, its trust policy cannot be altered. This ensures that only the designated AWS service can utilize the policy attached to that role and no other actor is authorized.



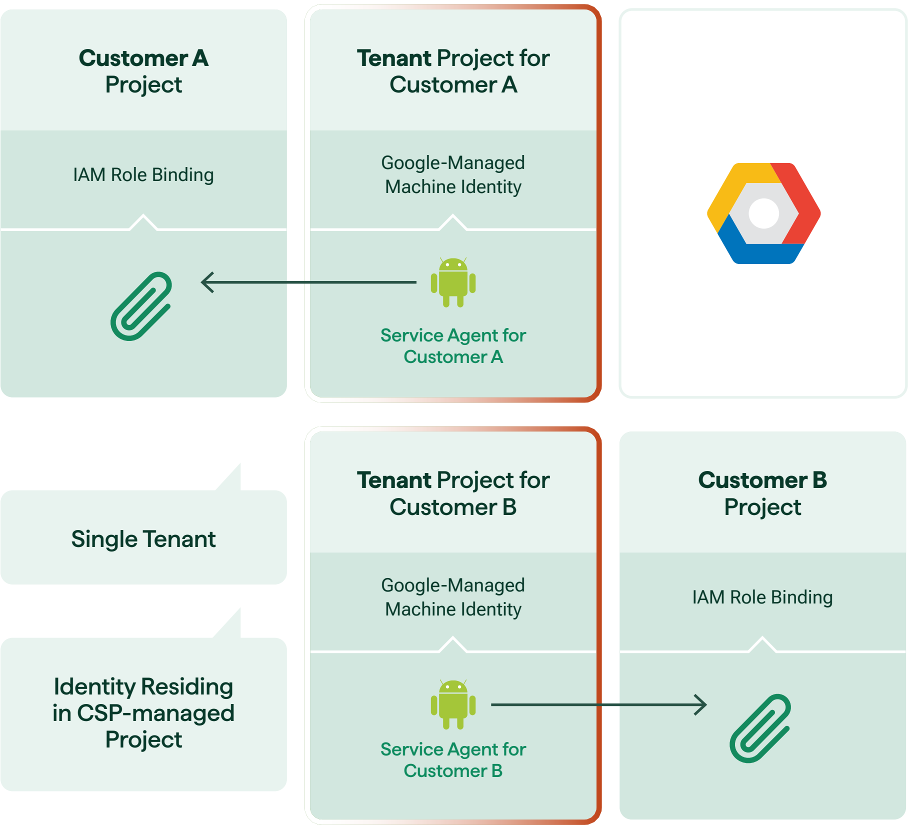
Google Cloud Machine Identities

In Google Cloud, CSP-managed machine identities are known as Service Agents. These are unique service accounts automatically created for each Google Cloud service within a specific project. Their naming convention follows the format: `service-<PROJECT_NUMBER>@<SERVICE_NAME>.iam.gserviceaccount.com`.

When a Google Cloud service needs to act on resources within a customer's project, it utilizes its designated Service Agent. The necessary permissions are granted through IAM (Identity and Access Management) roles that are automatically assigned to the Service Agent upon its creation.

Key characteristics of Google Cloud Service Agents include:

1. **Single-Tenancy:** A new Service Agent resource is created for every customer project and service, meaning it will only ever have IAM permissions to a single project.
2. **Fully CSP-Managed:** Service Agents reside as resources in Google-managed tenant projects, in a Google-controlled hierarchy. This prevents direct manipulation or modification by users from their own consumer projects.


Customer A Project
IAM Role Binding
Tenant Project for Customer A
Google-Managed Machine Identity
Service Agent for Customer A
Single Tenant
Identity Residing in CSP-managed Project
Tenant Project for Customer B
Google-Managed Machine Identity
Service Agent for Customer B
Customer B Project
IAM Role Binding

Microsoft Machine Identities

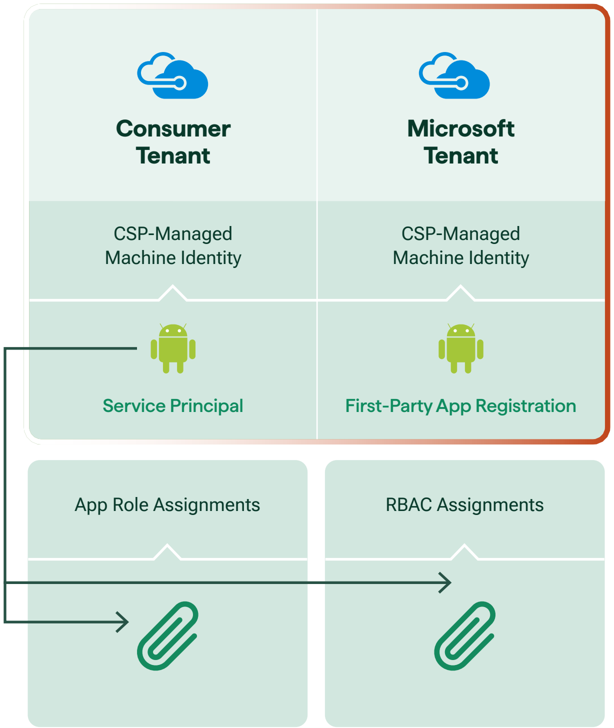
In the Microsoft ecosystem, **CSP-managed machine identities** are represented by two distinct but related objects: First-Party Application Registrations and their corresponding Service Principals. These identities represent internal services across both Azure and Microsoft 365.

The **Application Registration** acts as a global blueprint for a Microsoft service. It outlines the application's fundamentals and the set of permissions it requires to function. When a customer consumes or uses a service that requires access to their environment, a local **Service Principal** is created within their Microsoft Entra ID tenant. This Service Principal is linked to the global First-party application registration.

To perform actions, the Microsoft First-Party Application uses this local Service Principal to access resources. Permissions are granted via **App Role Assignments**, which are automatically assigned to the Service Principal, giving it the specific entitlements needed to operate within that tenant.

Key characteristics of Microsoft Machine Identities include:

1. **Multi-Tenancy:** Similar to AWS, the model is multi-tenant. A single global first-party Application Registration, identified by its unique application ID, is instantiated as a local Service Principal in multiple customer tenant.
2. **Complex Interconnectivity:** These principals create an intricate fabric of services that orchestrates critical background tasks for platforms such as Azure Machine Learning and SharePoint Online.


Consumer Tenant
Microsoft Tenant
CSP-Managed Machine Identity
CSP-Managed Machine Identity
Service Principal
First-Party App Registration
App Role Assignments
RBAC Assignments

What Can Go Wrong?

In this section, we will analyze what can go wrong by highlighting a distinct vulnerability class for each cloud. We will examine how the specific implementation of GCP Service Agents, AWS Service-Linked Roles, and Microsoft First-Party Application Registrations makes each platform susceptible to a unique type of security risk.

AWS | Confused Deputy - Multi-Tenant

One type of confused deputy problem in AWS occurs when a malicious actor coerces an AWS service to perform an action in another account that the attacker does not have permission to perform directly such as reading or writing to an S3 bucket. AWS describes this vulnerability as a cross-service confused deputy issue.

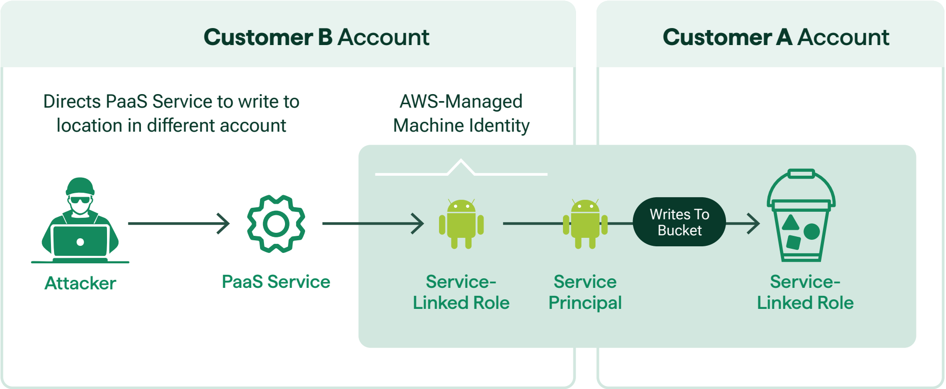
Since the AWS service principal (e.g., cloudtrail.amazonaws.com) is a single, global identity that acts on behalf of millions of different customers, it will have cross-account permissions allowing this vulnerability to be exploited across different AWS accounts.

The Multi-Tenant Confused Deputy

The risk emerges when a customer grants an AWS service permission to access their resources without any conditions through an IAM resource policy (i.e., an S3 bucket policy).

1. **Granting Trust**: A user in Account A configures a Service-Linked Role and grants the AWS service (e.g., ssm.amazonaws.com) permissions to perform tasks in the Account.
2. **The Flaw**: Without proper constraints, this IAM policy allows the global AWS service to be invoked from any Account, not only when acting on behalf of Account A.
3. **Cross-Account Exploitation**: Using the same service, an attacker in Account B can then trick the service into acting in Account A.
4. **Unauthorized Access**: The AWS service, now acting as a "confused deputy," uses the permissions allowed in Account A to perform actions on behalf of the attacker in Account B, commonly leading to unauthorized read or write access to data.

This vulnerability can allow an attacker in an entirely separate AWS account to access or modify resources in a victim's account.



Google Cloud | Confused Deputy - Single Tenant (Transitive Access)

Transitive access abuse is a privilege escalation technique where an attacker indirectly leverages the permissions of a trusted machine identity. The service with the identity becomes a "confused deputy," tricked into misusing its privileges to perform actions on the attacker's behalf.

The Single Tenant Attack Pattern

1

Exploit a Trusted Service

An attacker identifies a cloud service that operates with powerful permissions (e.g., a data processing service).

2

Provide Malicious Input

The attacker makes a valid request to the service but directs the service to interact with a resource they control, such as an external storage location.

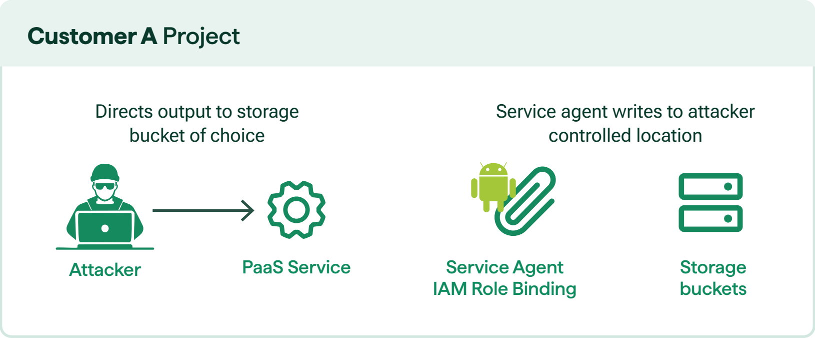
3

Bypass Permissions

The service utilizes its own permissions to execute the request, allowing the attacker to read or write data that they would not be able to access directly.

Case Study: Google Cloud Document AI

This pattern was observed in an exploit discovered by Kat Traxler (Vectra AI) in September 2024. The Document AI Service Agent could be manipulated to exfiltrate data from a Cloud Storage bucket to an attacker-controlled location, bypassing the caller's lack of IAM permissions. This issue has since been resolved, and callers are now required to have permissions on the objects the service accesses on their behalf.


Customer A Project
Directs output to storage bucket of choice
Service agent writes to attacker controlled location
Attacker
PaaS Service
Service Agent IAM Role Binding
Storage buckets

The `iam:PassRole` Permission in AWS

AWS is inherently more resistant to this type of single-tenant confused-deputy due to a specific permission check: iam:PassRole. This permission is required whenever a user 'passes' an Identity to an AWS service to use for an operation, whether a traditional IAM Role or a Service-Linked Role.

The `iam:PassRole` permission is a critical, but imperfect, preventative control because it applies only during the initial configuration of a service, not during its ongoing use from a service. Its purpose is to ensure a user is explicitly authorized to pass a role to a service, which serves as a key checkpoint for potential privilege escalation. Once the role is passed and bound to the service, an authorized user can leverage the potentially more powerful role.

Architecting Yourself into a Corner - Google Cloud

Google Cloud has a similar permission, `iam.serviceAccounts.actAs`. However, unlike in AWS, this permission is only required when associating customer-managed Service Accounts with resources; it is not required for CSP-managed Service Agents leveraged by PaaS services.

This gap is a direct result of the platform's architectural choices. Since Service Agents reside in Google-managed projects, customers cannot be granted the `iam.serviceAccounts.actAs` permission on the Service Agents themselves. Consequently, Google's platform cannot use this type of secondary permission to gatekeep its PaaS services, leaving a potential risk that the AWS `iam:PassRole` model effectively addresses.

Microsoft | Credential Abuse - Service Principal Hijacking

Microsoft Azure: Vulnerable by Design

The architecture of Microsoft's application registrations and their corresponding Service Principals creates a unique privilege escalation opportunity. The Service Principal is a local object residing directly within the consumer's tenant. This allows administrators with sufficient privileges within their own tenant to add credentials to Microsoft-owned, highly privileged applications.

This technique, known as **Service Principal Hijacking**, was exploited by the APT29 group in the December 2020 "SolarWinds" attack. An attacker with privileged access to an Entra ID tenant (such as holding either the *Application Administrator* or *Cloud Application Administrator* role) can add a new credential (e.g., a certificate or client secret) to a pre-existing Service Principal. By doing so, the attacker can authenticate as that trusted Microsoft service, inheriting all of its "birthright" permissions without altering roles or triggering common alerts.

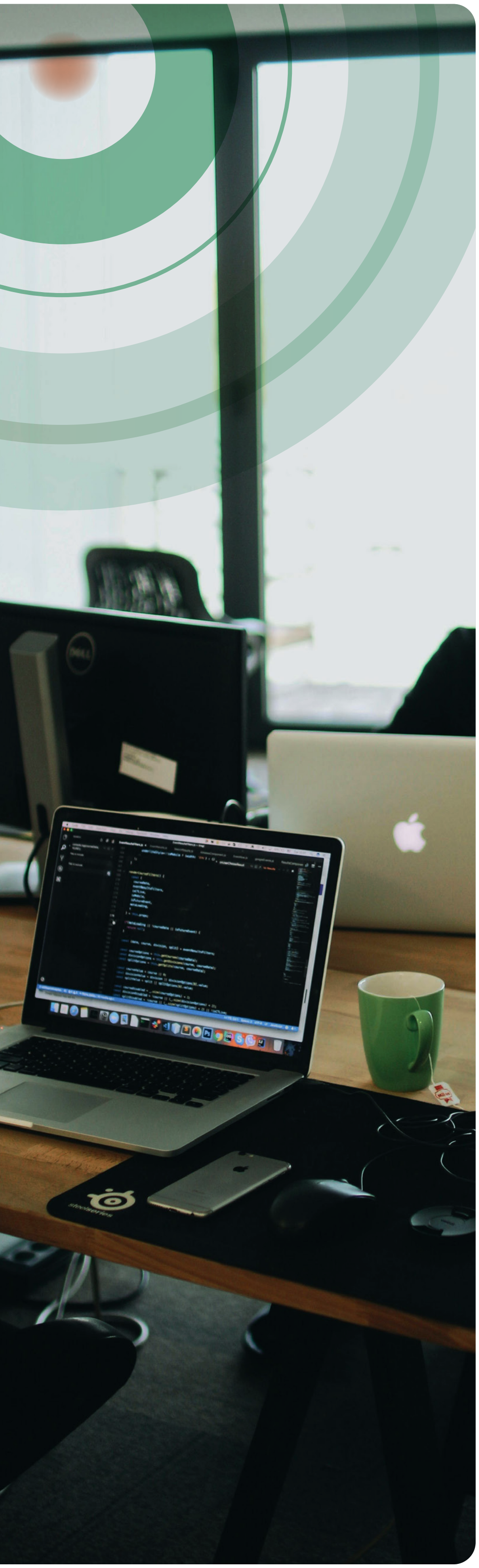
The public history of this vulnerability shows a gradual escalation in understanding and awareness before mitigations were introduced:

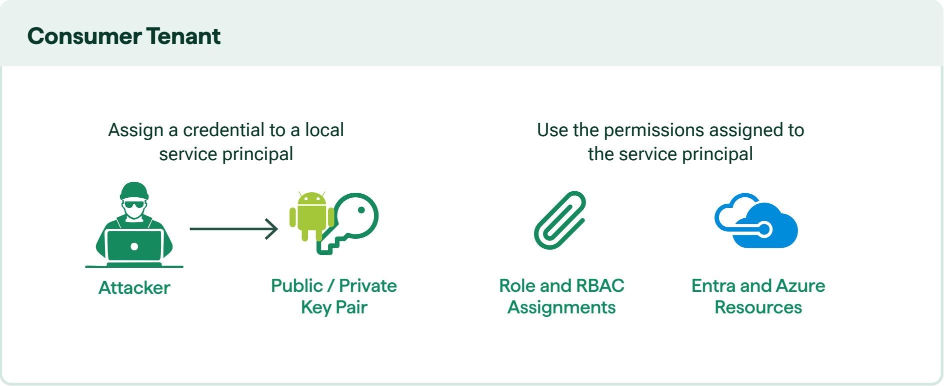
2019 The technique was publicly detailed initially by researcher Dirk-jan Mollema, who demonstrated how the *Application Admin* role could be used to take over application permissions.

2020 The method was observed in the high-profile SolarWinds attack. In response, Microsoft documented Service Principal persistence techniques, and the Azure team released the "Stormspotter" tool to help defenders map these relationships.

2021 Further community research from Emilian Cebuc and Christian Philipov ("Has Anyone Seen the Principal") continued to highlight the risks.

2022–2024 Public awareness continued to grow, for example, with further research from security professionals, such as Eric Woodruff ("Un0uthorized"). In response to this sustained community focus, Microsoft introduced a new security mechanism in Entra ID called *App Instance Property Lock* and began to implement it in its own first-party applications.




Consumer Tenant
Assign a credential to a local service principal
Use the permissions assigned to the service principal
Attacker
Public / Private Key Pair
Role and RBAC Assignments
Entra and Azure Resources

AWS and Google Cloud: A More Resilient Architecture

Both AWS and Google Cloud have architectures that are inherently resistant to this form of credential abuse for their managed identities.

In **AWS**, long-term credentials like access keys cannot be generated for IAM Roles, a category that includes Service-Linked Roles. Roles are intended to be utilized only with short-term credentials; however, with Service-linked Roles, only AWS-owned and operated Service Principals are authorized to generate these.

In Google Cloud, while cryptographic key pairs can be created for Service Accounts (including Service Agents), doing so requires the specific `iam.serviceAccountKeys.create` permission. Since Service Agents reside in Google-managed projects, consumers cannot grant this permission, effectively preventing them from creating new credentials for these powerful identities.

What are we going to do about it?

In the "What are we going to do about it?" section, I categorize the recommendations into two groups: actions that customers can take and mitigations and controls that only cloud service providers can implement.

Changes Cloud Providers Should Make

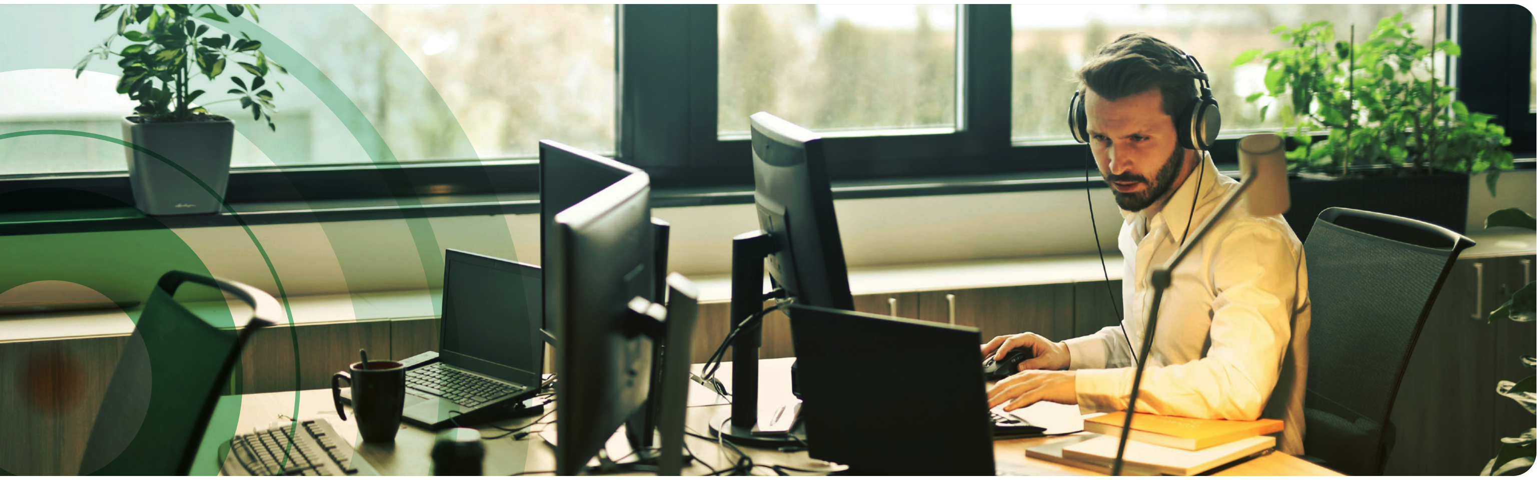
Single Tenant Identities

To reduce the blast radius from a potential confused deputy attack, service providers should design their CSP-managed machine identities to be single-tenant. Google Cloud implements this best practice by default, as its Service Agents are single-tenant identities created uniquely for each enabled service within every project. In contrast, the Principals for AWS Service-Linked Roles and Microsoft's first-party applications are global and multi-tenant. This design increases the blast radius of a successful confused deputy attack, which can occur when a deputy is directed to target another customer's account or tenant.

The explicit recommendation to CSPs is to architect managed identities that are single-tenant by design. This approach eliminates the cross-tenant impact of confused deputy attacks and, crucially, relieves the customer of the burden of implementing and maintaining manual mitigations against a platform-level vulnerability.

Zero Birthright Permissions

When a CSP-managed machine identity is created, its permissions are assigned using one of two models. The first is "birthright permissions," where the identity automatically receives privileges regardless of the creator's authority. The second, more secure model requires that the creator possess the specific authority to grant any assigned roles or policies.



The major cloud providers implement these models differently:

AWS (Explicit-Authority Model): AWS requires the caller to have the authority to grant permissions. Attaching a policy to a Service-Linked Role depends on the caller's explicit permissions, such as `iam:AttachRolePolicy` and `iam:PutRolePolicy`, and will fail without them.

Google Cloud (Birthright Model): When a user enables a service in a project, its associated Service Agent is automatically created and granted the necessary IAM permissions. This action does not check the user's own authority to grant those permissions.

Microsoft Entra ID (Birthright Model): Similar to Google, Entra ID automatically provisions first-party application service principals with predefined, often highly privileged, application roles (Application Role Assignments) when a service like Microsoft 365 is enabled. This process bypasses any check on the administrator's specific authority to assign those permissions.

The explicit recommendation to the CSPs is to phase out the birthright permissions model. This model represents a privilege escalation vector, as it allows a caller without permission-granting authority to instantiate a new, often powerful, identity simply by enabling a service.

Instead, CSPs should adopt an explicit-authority model when assigning permissions to non-human identities. The creation and permissioning of any managed identity (including those managed by the CSP) should be an explicit, auditable event that fails unless the calling principal possesses the specific authority to grant the requested roles. This change would ensure that the creation of privilege is always an intentional and authorized act.

Identity Location and Security Responsibility

The responsibility for securing a CSP-managed non-human identity—whether it falls to the customer or the provider—is determined by where that identity resides. The key distinction is whether the identity exists as a resource within the customer's environment or is managed exclusively by the provider.

Here is how the major cloud providers approach this:

Google Cloud (Provider-Controlled): Service Agents are managed entirely within a Google-controlled environment. There is no corresponding identity resource provisioned inside the customer's project, which centralizes security responsibility with Google.

AWS & Microsoft (Hybrid Model): These providers use a split-responsibility model where the identity is both global and local. The global identity (the AWS Service Principal or Microsoft's Application Registration) resides in the provider's environment. A local representation (an AWS Service-Linked Role or a Microsoft Service Principal) is also created as a resource within the customer's account or tenant, creating a shared security responsibility.

Recommendations for Cloud Providers

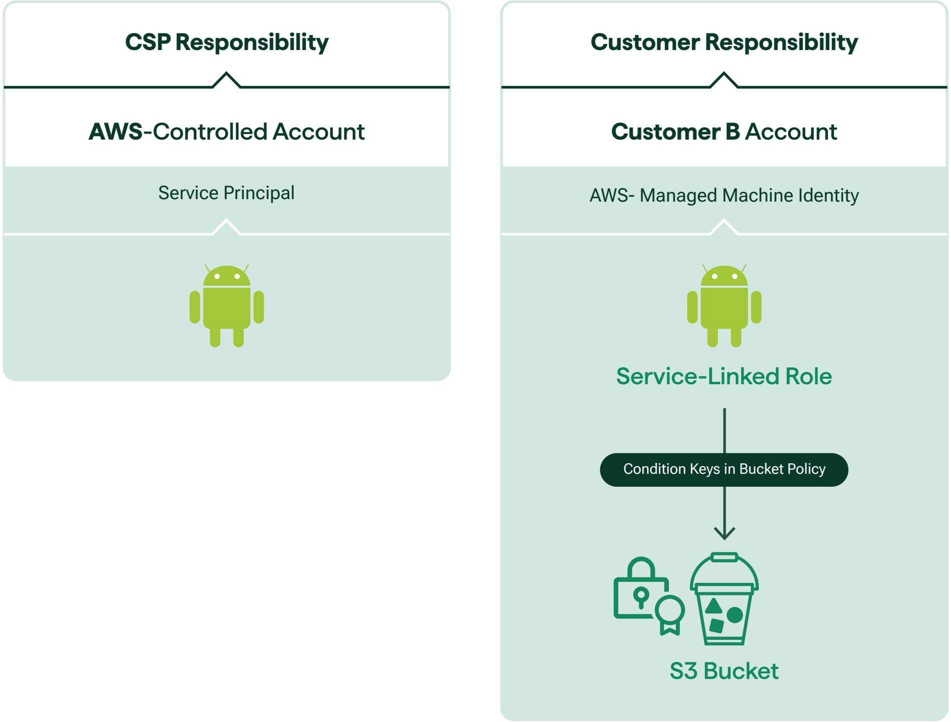
Cloud providers should architect managed identities to reside exclusively within their own controlled environments whenever possible. This approach minimizes the customer's attack surface and prevents the identity from being repurposed or misconfigured. In hybrid models where an identity resource must exist within the customer's tenant, the provider must implement strict guardrails to ensure security. These controls should lock the identity to its intended service, prohibiting customers from attaching it to their own compute resources or otherwise using its privileges for unintended actions. Ultimately, the burden of preventing the abuse of a provider's own identity should fall on the provider, not the customer.

Actions on the Customer-side of the Shared Responsibility Model

While AWS and Microsoft use a hybrid model for managed non-human identities, the preventative controls available to the customer are not equal across the major cloud providers. The customer's role ranges from being a critical part of the solution to having no direct controls at all.

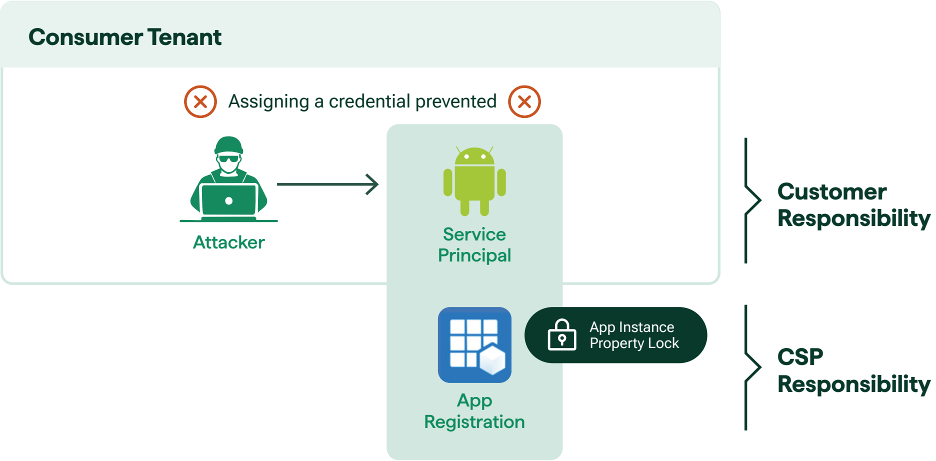
AWS Customer Action Required

Customers have both the ability and the responsibility to secure their resources from erroneous Service Principal access. They must manually configure resource-based policies with condition keys (e.g., `aws:SourceArn`) to prevent confused deputy attacks. This gives them powerful control but also places the configuration burden on them.



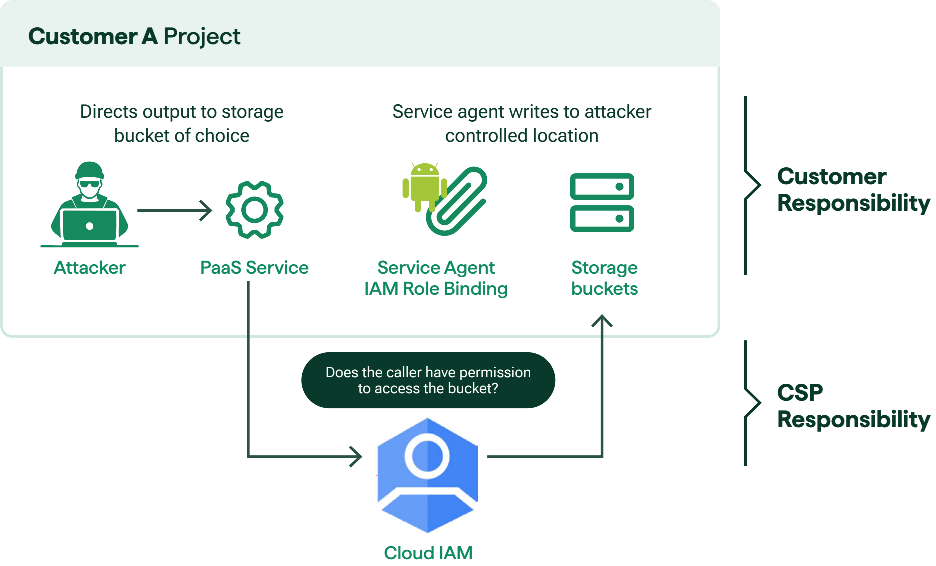
Microsoft Entra ID: No Customer-Side Controls

Customers have no direct controls to configure. The primary mitigation, the App Instance Property Lock, is a property that can only be set by Microsoft as the publisher of its first-party applications. Responsibility rests entirely with the provider.



Google Cloud: No Customer-Side Controls

Similar to Microsoft, customers cannot implement their own preventative measures against Service Agent abuse. It is solely Google's responsibility to design its services to validate the caller's explicit authority. Any failure to do so is a vulnerability that only Google can fix.


Customer A Project
Directs output to storage bucket of choice
Service agent writes to attacker controlled location
Attacker
PaaS Service
Service Agent IAM Role Binding
Storage buckets
Does the caller have permission to access the bucket?
Cloud IAM
Customer Responsibility
CSP Responsibility

Ultimately, the debate over these controls boils down to a core security philosophy.

Is the absence of a customer-side control a "missing tool" in their toolbox, or is it the provider properly shouldering its security burden? AWS empowers—and burdens—the customer with this responsibility, while Google and Microsoft choose for them, placing the onus for securing these identities entirely on their own platforms.

Did We Do a Good Job?

Whether the preventative measures are the responsibility of the customer or the CSP, in this section, let's review how well each cloud (and its customers) mitigates the inherent vulnerabilities.

Adoption of Condition Keys in AWS

The consistent theme from cloud security research is that overall IAM hygiene in AWS is poor. While direct studies on the precise adoption rate of condition keys, such as `aws:SourceAccount` and `aws:SourceArn`, don't exist, the consensus from the security community indicates that their usage is dangerously low. Given the widespread challenges with general IAM policy management, it's reasonable to infer that these specific, crucial controls are often overlooked, likely leaving many organizations vulnerable to multi-tenant confused deputy attacks in AWS.

Caller Authority Checks in Google Cloud

The primary prevention against the abuse of Google Cloud Service Agents is for the provider to ensure its services validate the caller's authority before taking action. To date, there has been one publicly documented failure of this control: a privilege escalation via Transitive Access Abuse in the Document AI Service Agent, which Google has since remediated. However, the limited number of public disclosures may suggest that further independent research is necessary to determine if other pathways for this attack technique exist.

The appInstancePropertyLock Control in Microsoft Entra ID

The App Instance Property Lock is a security control developers use to protect their multi-tenant applications by preventing credentials from being added to local service principals in customer tenants. While this property has been enabled by default on all new applications created after March 2024, the setting was not applied retroactively to existing applications. As a result, many of the 300+ first-party applications installed by default in Entra ID tenants remain unlocked until Microsoft individually updates them, creating a significant window of risk.

This gap was highlighted at the fwd:cloudsec 2025 conference by researcher Katie Knowles. She demonstrated that the Office 365 Exchange Online application, which is installed by default in all Entra ID tenants, lacks the App Instance Property Lock. This oversight allows a user with the *Application Administrator* or *Cloud Application Administrator* role to add their own credentials to the application's service principal. Because this specific service principal holds high-level directory privileges, the action allows the user to escalate their permissions to that of a *Global Administrator*.

In Conclusion

The design choices made by AWS, Google Cloud, and Microsoft have resulted in fundamentally different security postures for their managed non-human identities. While each provider aims to solve the same challenge, their architectural nuances create distinct threat models, shifting the security burden between the provider and the customer.

There is no single "best" approach; instead, security professionals must understand where they have preventative controls available and where they must rely solely on detection and response to thwart non-human identity abuse.

Here are the key takeaways from this comparative threat model:

Google Cloud's architecture is inherently resilient against the specific attacks discussed, utilizing single-tenant, provider-controlled identities. However, this model places the entire burden on Google to ensure its services are not vulnerable to transitive access abuse, leaving customers with no direct controls.

AWS provides customers with powerful preventative tools, but this control comes with significant responsibility. The multi-tenant nature of its services makes it susceptible to a type of Confused Deputy attack, and the security of the ecosystem hinges on customers correctly implementing condition keys in IAM Policies. This practice is currently likely underutilized and can pose a significant cross-tenant risk.

Microsoft's hybrid model has historically been vulnerable to credential abuse via Service Principal hijacking. While the App Instance Property Lock is a new and effective preventative control, its slow and non-retroactive rollout on hundreds of default first-party applications creates the most immediate and high-impact risk among the three providers today.

Threats are not uniform. The most critical threat in one cloud may be a non-issue in another. Defenders and researchers must tailor their strategies, recognizing that there is no "1-to-1" approach to security controls in a multi-cloud environment.

About Vectra AI

Vectra AI, Inc. is the cybersecurity AI company that protects modern networks from modern attacks. When modern cyber attackers bypass existing controls, evade detection and gain access to customers' data center, campus, remote work, identity, cloud, and IoT/OT environments, the Vectra AI Platform sees their every move, connects the dots in real-time, and stops them from becoming breaches. With 35 patents in AI security and the most vendor references in MITRE D3FEND, organizations worldwide rely on Vectra AI to see and stop attacks their other tools can't. For more information, visit www.vectra.ai.

For more information please contact us:
Email: info@vectra.ai | vectra.ai