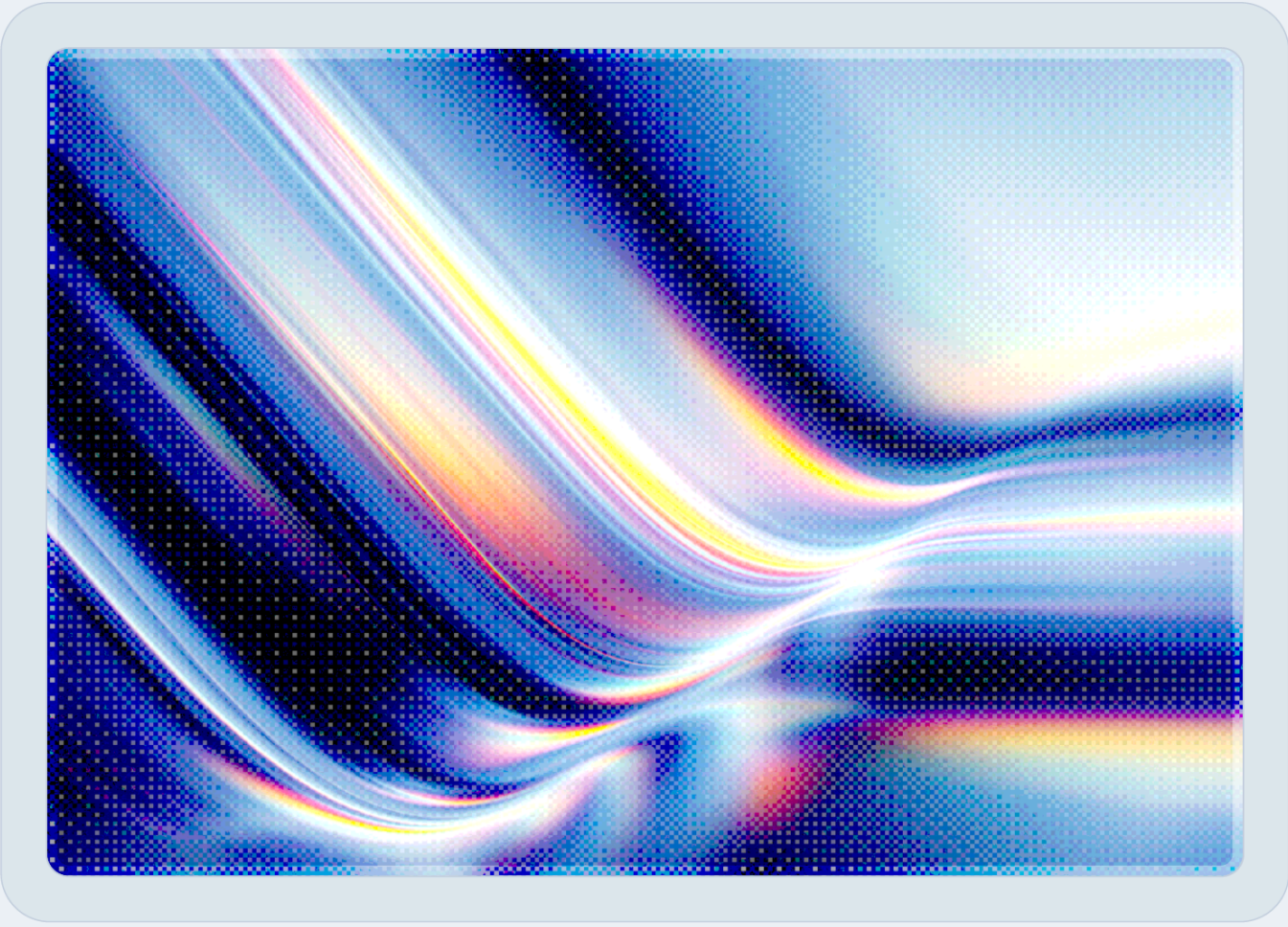
CSAI Foundation | Cloud Security Alliance

Cloud Namespace Hijacking: When Deleted Buckets Become Backdoors

How Abandoned Storage Bucket Names Enable Persistent Data Exfiltration Across AWS, GCP, and Azure

2026-06-29

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Table of Contents

Executive Summary

Cloud storage bucket names occupy a globally unique namespace. No two accounts can simultaneously hold a bucket with the same name in the same region—or, in the case of AWS S3 legacy buckets, the same name anywhere on earth. This design choice, made in the earliest days of commodity cloud storage to simplify access URL construction, has accumulated a security debt of extraordinary proportions: when an organization deletes a bucket, its name re-enters the global pool and can be registered by any other account, including an attacker's.

The consequences are not theoretical. In a landmark demonstration published in February 2025, researchers at watchTowr re-registered approximately 150 abandoned Amazon S3 buckets—buckets that had previously hosted software distribution assets for government agencies, Fortune 500 companies, military organizations, and major open-source projects—for a total cost of $420.85 [1]. Over a two-month monitoring period, those buckets received more than eight million HTTP requests for software updates, unsigned executables, virtual machine images, CloudFormation templates, and SSL VPN configurations. The requestors included NASA, military networks from the United States, United Kingdom, Australia, and South Korea, Fortune 100 and Fortune 500 companies, a major payment card network, banks, universities, and cybersecurity firms [1][2]. Not a single requestor knew they were delivering their environment's trust to an attacker.

Separate research from Palo Alto Networks Unit 42 expanded the threat model beyond supply chain manipulation. Researchers demonstrated that an attacker with bucket-deletion permissions on a target cloud account can hijack active data streams—cloud audit logs, telemetry pipelines, Pub/Sub routing, and Amazon Data Firehose deliveries—by deleting the destination bucket and immediately re-creating it under their own account, exploiting the same global namespace flaw to redirect live organizational data into attacker-controlled storage, silently and without disturbing the upstream configuration [3]. The technique works across AWS, GCP, and Microsoft Azure.

The Aqua Security research team identified a parallel variant in the AWS Cloud Development Kit (CDK) in June 2024: because CDK's bootstrap process creates staging buckets with predictable, publicly derivable names, an attacker who causes the deletion of a CDK staging bucket can substitute their own bucket and intercept the next deployment, injecting malicious infrastructure-as-code into a victim's CloudFormation pipeline with administrative-level consequences [4][5]. AWS confirmed that approximately one percent of all CDK users were exposed; analysis of 38,560 well-known AWS account IDs found 81 demonstrably vulnerable environments [4].

These three research streams—supply chain poisoning, active stream hijacking, and infrastructure-as-code compromise—converge on a single structural vulnerability: the cloud storage global namespace makes bucket ownership permanently ambiguous after deletion. The name survives; the owner does not. Every reference that continues to resolve that name is an unsecured trust relationship.

AWS partially addressed the problem in March 2026 by launching account regional namespaces for S3 general purpose buckets—an opt-in feature that scopes new buckets to an account-specific suffix, making re-registration by another party impossible [6]. The fix is prospective and opt-in only; existing global-namespace buckets, the vast majority of cloud storage infrastructure in use today, remain exposed, and references to deleted buckets remain unmitigable from the cloud provider's side.

This paper examines the architectural origin of global namespace risk, documents how the attack surface has evolved across providers and attack vectors, surveys the known impact of real-world exploitation research, and provides structured guidance for security teams seeking to identify and remediate their organization's exposure.

Introduction: The Architecture of a Permanent Vulnerability

How Cloud Storage Buckets Got Their Names

Object storage emerged as a public cloud primitive in the mid-2000s, designed for simplicity and global accessibility. Amazon's Simple Storage Service, launched in 2006, gave each bucket a DNS-accessible endpoint of the form `bucketname.s3.amazonaws.com`. Because the endpoint was globally resolvable without authentication context, the bucket name had to be globally unique—any two accounts could not hold the same name simultaneously, because that would make the DNS record ambiguous [7].

Google Cloud Storage and Microsoft Azure followed similar conventions, each constructing storage endpoint URLs from bucket or container names in ways that tied the name to the globally reachable endpoint. The result was a naming model where bucket identity was encoded in the URL itself, decoupled from account identity. This was architecturally convenient—applications could reference storage resources by name without needing to embed account credentials in URLs—but it introduced a structural property with lasting security consequences: the name was the asset. Whoever controlled the name controlled the endpoint.

The Deletion Gap

When an organization deletes a cloud storage bucket, the provider frees the name. Unlike domain registrations, which expire after a contractually defined period and enter a recovery grace window before becoming publicly available again, cloud bucket names re-enter the available pool immediately or within seconds. Any account—including one controlled by an attacker—can register the same name in the same namespace.

The structural problem is that code, configurations, and systems that referenced the deleted bucket name have no way to know the bucket is gone, and no way to know that a new entity now controls it. A software package that was compiled six years ago and still ships in enterprise Linux distributions may contain a hardcoded URL for a binary it downloads at install time. A CloudFormation template saved in an internal artifact repository may reference a bucket deleted when the owning team was reorganized. A DNS CNAME record pointing to a static S3 website may outlive the hosting bucket by years if the operator forgets to delete it. An audit log sink routing CloudTrail records to an S3 destination continues routing after the destination bucket is gone—and after an attacker creates a new bucket under the same name.

None of these references generate errors visible to the system operator until the bucket is absent; once it is present again—under attacker control—they generate no errors at all. The configuration appears valid. The traffic flows. The data arrives at the wrong destination.

The Permanence Problem

What distinguishes cloud namespace hijacking from most cloud misconfigurations is its permanence. A misconfigured S3 ACL can be corrected. A leaked access key can be rotated. An over-permissive IAM policy can be tightened. But a bucket name that has been referenced in distributed software artifacts, open-source package dependencies, third-party deployment templates, and archived configuration repositories cannot be centrally recalled.

The watchTowr research made this concrete: researchers did not find 150 buckets that had been deleted recently and might be reclaimed by their original owners. They found buckets whose names were still being actively requested—eight million times in two months—by systems whose operators had no knowledge of the references [1][2]. Many of those requests came from automated deployment pipelines, software update mechanisms, and infrastructure provisioning tools that had no human in the loop to notice that something was wrong. The organizations making those requests were, in a meaningful sense, already compromised. They were delivering request-borne trust—DNS resolution, TLS negotiation, object fetching—to infrastructure they did not own.

This is the core of the namespace hijacking risk: the attack surface is not a configuration that can be corrected, but a historical reference distribution problem that cannot be fully inventoried, let alone remediated, from any single organization's vantage point.

Attack Mechanics: Three Paths to Compromise

Supply Chain Poisoning via Abandoned Buckets

The most operationally accessible form of namespace hijacking involves identifying abandoned buckets still referenced by software ecosystems and re-registering them to serve malicious payloads. The prerequisites are low: the attacker needs no credentials, no insider access, and no prior relationship with the victim organization. They need only find a bucket name that is available for registration and still referenced by software.

Finding such names is not technically demanding. Attackers can scan public code repositories for hardcoded S3 bucket URLs in install scripts, build configurations, and package manifests. They can analyze the dependency trees of popular open-source packages for references to external storage. They can monitor DNS resolution patterns and look for domains whose CNAME records point to non-existent S3 buckets—the classic "dangling CNAME" pattern that underpins subdomain takeover attacks. They can search archived package registries for deprecated package versions that shipped with bucket references now pointing at unregistered names.

The Checkmarx research team documented this attack surface in detail through the compromise of the `bignum` NPM package. Versions 0.12.2 through 0.13.0 of the package downloaded a native binary from an S3 bucket during installation. The bucket was abandoned and deleted approximately six months before the attack. An attacker registered the abandoned bucket name and replaced the legitimate binary with a malicious payload. The malicious binary harvested user credentials, environment variables, and system identifiers using standard POSIX functions, then exfiltrated them by encoding the stolen data in the user-agent string of a GET request back to the hijacked bucket—disguising the exfiltration as ordinary web traffic [8]. The attack required no modification to the package's source code, no access to the NPM registry's administrative interface, and no interaction with the victim beyond the ordinary installation of a widely used dependency. Checkmarx researchers noted that dozens of other packages in the NPM ecosystem exhibited the same pattern of hardcoded references to deleted or unclaimed S3 buckets [8].

The watchTowr demonstration in late 2024 extended this analysis to a broader institutional scope. Researchers identified approximately 150 abandoned S3 buckets previously used by government agencies, major open-source projects, commercial software vendors, and infrastructure tooling companies, and registered all of them for a total of $420.85 [1]. Over two months of monitoring, those buckets received

more than eight million requests for executables, virtual machine images, CloudFormation templates, SSL VPN configurations, and JavaScript files. The requestors were not consumer users casually browsing the web; they were automated deployment pipelines, software update services, and infrastructure provisioning tools running inside enterprise and government networks [1][2]. watchTowr CEO Benjamin Harris described the attack path as "terrifyingly simple," requiring an attacker to do nothing more than register the bucket and place a malicious payload in the expected object path [2].

In responsible disclosure, watchTowr notified AWS, which agreed to sinkhole the identified buckets—preventing malicious registration for the specific names identified in the research [1]. That action addressed the known 150 names. The broader population of abandoned bucket names with live references across the global software ecosystem remains unquantified.

Active Data Stream Hijacking

A second attack vector, documented by Palo Alto Networks Unit 42, targets not software distribution but live operational data flows. The technique exploits the same global namespace flaw to hijack cloud-native logging, telemetry, and replication streams directed at storage buckets [3].

The attack's precondition is that an adversary has compromised an identity with bucket-deletion permissions inside a target cloud account—a common scenario following credential theft, IAM misconfiguration exploitation, or insider access. Once they hold that permission, the attacker can delete a bucket that serves as a destination for operational data flows: a Cloud Logging sink routing audit events to a GCS bucket in Google Cloud, a CloudTrail logging configuration delivering to an S3 bucket in AWS, an Amazon Data Firehose stream writing telemetry to S3, or a storage replication job transferring objects across accounts.

Deletion of the destination bucket does not halt the data flow. Cloud-native routing and replication services continue to attempt delivery. When the attacker immediately re-creates a bucket with the identical name in their own account, the data stream begins delivering to attacker-controlled storage. From the victim account's perspective, the logging configuration, replication rule, or Firehose stream appears intact—it references a valid, existing bucket. There is no error condition, no alert, and no visible indication that the destination has changed ownership [3].

Unit 42 researchers confirmed the technique across Google Cloud Platform (Cloud Logging sinks, Pub/Sub topic routing, Storage Transfer Service), Amazon Web Services (S3 replication, Amazon Data Firehose), and Microsoft Azure (Monitor diagnostic settings, though Azure's soft-delete protections provide a narrower window for exploitation) [3]. The data exposed through this vector includes cloud audit logs—the very records that a security operations team would rely on to detect compromise—along with application telemetry, infrastructure metrics, and any business data flowing through managed replication or transfer services.

The researchers identified no evidence of active threat actor exploitation of this specific technique as of their publication, but noted that real-world exploitation would be "difficult to detect" precisely because the attack leaves no anomalous signals in the systems that direct the data flows [3]. An attacker who hijacks an audit log stream has also degraded their victim's capacity to detect that hijacking through the audit record.

Infrastructure-as-Code Compromise via Predictable Names

A third attack path combines namespace hijacking with the predictable bucket names generated by infrastructure-as-code frameworks. Aqua Security researchers disclosed in June 2024 that the AWS Cloud Development Kit's bootstrap process created S3 staging buckets with names following a deterministic pattern: `cdk-{qualifier}-assets-{account-ID}-{region}` [4][5].

The qualifier value defaults to `hnb659fds` and most CDK users do not override it, meaning that the only unknowns in the staging bucket name are the AWS account ID and the target region. AWS account IDs, while not publicly advertised, are not secrets in practice—they appear in IAM role ARNs, which appear in CloudFormation template outputs, public documentation, error messages, and resource policies that are sometimes publicly visible. A determined attacker who knows a target organization's AWS account ID and knows they use CDK can construct the expected staging bucket name and check whether it is available for registration.

If the CDK staging bucket has been deleted—which occurs when organizations tear down environments, run CDK destroy operations, or clean up legacy infrastructure—the predictable name becomes claimable. An attacker who registers it has positioned themselves in the critical path of the victim's next CDK deployment. When the victim organization runs `cdk deploy`, CDK uploads CloudFormation templates and deployment assets to the staging bucket. Under normal circumstances, the victim's CloudFormation service then retrieves those templates from the staging bucket. With the attacker's bucket substituted, the attacker can modify the CloudFormation templates in transit—injecting malicious IAM roles, backdoors, or administrative policies—before the victim's CloudFormation service processes them with full administrative privileges [4][5].

Aqua Security's analysis of 38,560 well-known AWS account IDs found 782 accounts with CDK installed, of which 81 were vulnerable to this specific vector. AWS subsequently confirmed that approximately one percent of all CDK users faced this risk, and pushed a fix in CDK version 2.149.0 in July 2024 that adds an IAM condition restricting the deployment role to buckets within the same account [4][5]. The fix applies only to new bootstrap operations; organizations that bootstrapped prior to July 12, 2024 remain vulnerable until they re-run `cdk bootstrap` with the updated version. AWS notified affected customers in October 2024, but the fix requires active operator action that many organizations may not have taken.

Provider-Specific Analysis

Amazon Web Services

AWS S3 has operated on a globally flat namespace since its 2006 launch. Any bucket name chosen by a customer must be unique not just within their account but across all AWS accounts globally. When a bucket is deleted, its name becomes immediately available for registration by any AWS customer worldwide.

The full scope of the S3 namespace risk became apparent only as the ecosystem built around S3 URLs over nearly two decades. Software distribution infrastructure, CDN origin configurations, static website hosting, log delivery destinations, and infrastructure-as-code deployment pipelines all developed reference patterns that treat a bucket URL as a durable address. None of those references carry any mechanism to verify that the bucket at the referenced address is still owned by the expected account.

AWS partially resolved the forward-looking risk on March 12, 2026, by launching account regional namespaces for S3 general purpose buckets [6][7]. Under the new system, customers can create buckets with a name structured as `{prefix}-{account-id}-{region}-an`, where the account-regional suffix is globally reserved to that account. No other account can create a bucket with that account's suffix, eliminating the re-registration attack for buckets created in the account regional namespace. The feature is available in 37 AWS regions with no additional cost and is supported by the AWS CLI, Boto3, CloudFormation, and IAM condition keys [6].

The critical limitation is that the account regional namespace is opt-in for new bucket creation and cannot be applied retroactively to existing buckets. The vast installed base of legacy global-namespace buckets and all references to previously deleted global-namespace bucket names are entirely unaffected. Organizations that adopt the new naming convention for all new bucket creation will protect their future infrastructure but cannot reclaim the attack surface created by historical deletions or by buckets they continue to operate under legacy global names [7].

Google Cloud Platform

Google Cloud Storage uses a global namespace model substantially similar to S3's. Bucket names must be globally unique within the GCS namespace, and a deleted bucket's name is available for immediate re-registration [9]. Unit 42's research confirmed that Cloud Logging sinks routing to GCS buckets, Pub/Sub topic routing configurations, and Storage Transfer Service jobs are all vulnerable to the active stream hijacking technique described above [3].

Google launched a soft-delete feature for GCS in March 2024, enabling a configurable retention period—defaulting to seven days, extensible up to 90 days—during which deleted objects remain recoverable [9][10]. Soft delete at the object level provides protection against accidental or malicious deletion of data within buckets, but it does not hold bucket names in a reserved state during the retention window. A deleted bucket's name can be re-registered by another account even while its contents remain in a soft-deleted state, and Google's documentation explicitly notes that a soft-deleted bucket cannot be restored if its name has been reused [9].

The GCS bucket-level soft delete does provide a meaningful protection window against the specific attack path where an adversary deletes a bucket and immediately re-creates it in their own account: if soft delete is enabled with an adequate retention period and deletion events are monitored with short alert latency, the security team may be able to detect and respond to the bucket deletion before the attacker's re-creation window expires. This protection is conditional on proactive monitoring and rapid response, neither of which is guaranteed in most operational environments.

Microsoft Azure

Azure Blob Storage uses a slightly different namespace structure than AWS or GCP. Storage accounts have globally unique names within the `blob.core.windows.net` domain, but containers within a storage account are scoped to the account. The globally unique requirement is at the storage account level rather than at the container level, which provides some structural isolation [13].

Azure also offers soft-delete protections at the storage account and container level that are more robust than GCS's object-level protection, with configurable retention periods during which deleted storage accounts and containers remain in a soft-deleted state and their names are held in a reserved pool inaccessible to new registrations. Unit 42 noted that Azure's protections limit the exploitation window for the active stream hijacking variant to scenarios where Azure's soft-delete retention window has expired—a meaningful mitigation compared to AWS and GCP, though not an absolute one, particularly for storage accounts soft-deleted long enough ago that the retention period has elapsed [3].

Despite these structural advantages, Azure is not immune to the broader class of namespace hijacking attacks. The cross-subscription attack variant—where an attacker registers a storage account name in their own Azure subscription after the victim's subscription deletes it—remains viable once the soft-delete window expires. Organizations operating legacy Azure infrastructure, particularly those that have undergone mergers, acquisitions, or organizational restructuring that left stale references in codebases, face residual exposure.

Scale and Impact: What the Evidence Shows

The Known Exposed Surface

The research conducted across 2024 and 2025 provides concrete, if incomplete, evidence of the scale of the problem. The watchTowr study identified approximately 150 abandoned S3 buckets with active references for registration at negligible cost; the actual population of such buckets across the global S3 namespace is orders of magnitude larger and remains uncharacterized [1][2]. The researchers were not conducting an exhaustive scan—they were demonstrating a representative sample of the risk. The eight million requests received across those 150 buckets over two months represents only the traffic attributable to automated systems that happened to be requesting the specific bucket names the researchers had identified. It does not capture the traffic associated with the remaining universe of abandoned, un-registered buckets that other researchers or attackers may have found.

The Aqua Security CDK research offers a different calibration of scale. AWS confirmed that approximately one percent of CDK users were exposed to the staging bucket hijacking vulnerability [4][5]. AWS does not publish CDK usage figures, but CDK has been among the most widely adopted infrastructure-as-code tools since its general availability in 2019. A one percent exposure rate across a user base of this scale represents thousands of vulnerable environments. The affected accounts span every industry vertical and organization size category, because CDK's adoption is broad.

The Checkmarx analysis of the open-source NPM ecosystem found "dozens of packages" with references to abandoned or unclaimed S3 buckets following the bignum compromise disclosure [8]. NPM hosts hundreds of thousands of packages, each with their own dependency trees that may include transitive references to external storage. The actual number of packages vulnerable to supply chain attacks through abandoned bucket references is unknown and actively changing as packages are updated, deprecated, or abandoned.

Institutional Visibility Gap

A distinctive feature of the watchTowr research was the institutional profile of the organizations making requests to the re-registered buckets. These were not naive consumer users or hobbyist developers. They were production systems operating within the networks of government agencies—NASA, Department of Defense contractors, military organizations from multiple allied nations—financial institutions, payment networks, and companies from nearly every sector of the Fortune 500 [1][2]. The requests came from automated deployment pipelines and software update mechanisms running at organizational scale.

This profile reveals something important about the nature of the exposure. The organizations making those requests to attacker-controlled infrastructure almost certainly had active security programs, vulnerability management processes, and cloud security posture management tooling. None of that tooling was

positioned to detect that a bucket URL embedded in a legacy deployment script or inherited software package had changed hands. The attack surface existed in the gap between the security perimeter that organizations can monitor—their own cloud resources, their own network traffic, their own code repositories—and the distributed reference landscape that extends far beyond any single organization's visibility.

The Economics of Exploitation

One of the most concerning aspects of the namespace hijacking risk is its asymmetric economics. The watchTowr demonstration cost $420.85 for 150 buckets—under three dollars per bucket on average [1]. The unit economics of cloud storage registration are deliberately low: cloud providers have always wanted storage adoption to be frictionless. An attacker can systematically register abandoned bucket names with active references for costs that are trivially offset by any monetizable outcome—credential theft, ransomware staging, software supply chain poisoning, or the sale of collected operational data.

Contrast this with the cost of effective detection and response. An organization that wants to audit its codebase, historical configurations, infrastructure templates, and third-party dependencies for references to deleted or potentially compromised bucket names must conduct a time-intensive discovery exercise across a distributed, often poorly documented artifact landscape. Many of the references that matter most are in compiled artifacts, archived templates, and legacy system configurations that may not be indexed by standard SAST or SCA tooling. The offensive cost is trivial; the defensive cost is substantial.

Detection and Monitoring Challenges

The Silent Stream Problem

The Unit 42 research on active data stream hijacking identified a detection property that distinguishes this attack class from most cloud misconfigurability risks: the attack is specifically damaging to the logging and audit infrastructure that defenders rely on for detection [3]. When an attacker hijacks a Cloud Logging sink or a CloudTrail delivery bucket, they are diverting the records that would document the attack. The upstream routing configuration reports valid status. No alert fires. The audit logs that would show the bucket deletion, the bucket re-creation under a different account, and the subsequent data flow simply stop appearing in the environments where they should be.

This creates a compound risk. The attacker gains access to sensitive operational data—often including security-relevant audit events. The defender simultaneously loses visibility into those events. The attack is self-concealing in exactly the domain where detection most matters.

Dangling Reference Discovery

Organizations seeking to understand their exposure to the supply chain variant of namespace hijacking face a non-trivial discovery problem. Dangling references to deleted buckets appear in many artifact types that are difficult to enumerate comprehensively: compiled binaries, archived deployment scripts, CloudFormation and Terraform templates stored in artifact repositories, CI/CD pipeline configuration files, container images, virtual machine snapshots, configuration management databases, and third-party software packages incorporated as dependencies. Many of these artifacts are not actively monitored, indexed by security tooling, or visible through standard CSPM platforms.

DNS-based discovery provides partial coverage for one variant: dangling CNAME records pointing to S3 static website endpoints can be identified by enumerating DNS records for an organization's domains and checking whether the referenced bucket names are currently registered. Tools exist to automate this check, and subdomain takeover scanning has become a routine element of external attack surface management [11][12]. However, DNS-based discovery addresses only the subset of references that manifest as external DNS resolutions; internal references embedded in code and infrastructure templates require a different discovery approach.

The Inherited Reference Problem

A structurally difficult category of exposure involves references inherited through acquisitions, mergers, vendor relationships, or the adoption of open-source software. An organization that acquires a company inherits that company's codebase and infrastructure—including any hardcoded bucket references that were abandoned at some point in the acquired entity's history. An organization that deploys open-source software inherits that software's external dependencies, including any bucket references embedded in install scripts or build processes. These inherited references are particularly difficult to inventory because they are not visible in the acquiring organization's own codebase history and may not be surfaced by any tooling the organization routinely operates.

Organizational Recommendations

Immediate Actions

Organizations should begin with a time-bounded audit of their most likely exposure categories. The highest-priority investigation targets are DNS records with CNAME entries pointing to cloud storage hostnames. These can be systematically enumerated and verified against currently registered bucket names using any of

several publicly available subdomain takeover scanning tools, or through external attack surface management platforms. A CNAME record pointing to a non-existent bucket presents an immediate subdomain takeover risk and should be removed or repointed [12].

Simultaneously, organizations should audit active cloud logging and telemetry routing configurations across all cloud accounts and providers. Every Cloud Logging sink, CloudTrail S3 delivery configuration, Amazon Data Firehose stream with S3 destination, Pub/Sub topic routing, and Azure Monitor diagnostic setting should be verified: confirm that the destination bucket or storage resource exists and is owned by an account within the organization's cloud footprint. Configuration management tooling and cloud security posture management platforms can automate this inventory across account hierarchies. Any destination that cannot be verified as organization-owned should be treated as compromised until demonstrated otherwise [3].

Organizations using AWS CDK should verify their CDK version and bootstrap status. Any environment bootstrapped prior to July 12, 2024 is potentially vulnerable to the staging bucket hijacking technique described by Aqua Security [4][5]. The remediation requires upgrading to CDK version 2.149.0 or later and re-running the `cdk bootstrap` command in every affected region. This is a non-trivial operational action in environments with many accounts and regions, but it closes a specific, well-documented attack path with a known fix.

Near-Term Controls

For new cloud storage provisioning, organizations should adopt account-scoped naming conventions wherever the cloud provider supports them. AWS's account regional namespace feature, launched in March 2026, provides structural protection for newly created S3 buckets by making re-registration by another account structurally impossible [6][7]. Adoption should be enforced through AWS Organizations Service Control Policies that require the `s3:x-amz-bucket-namespace` condition key for new bucket creation, ensuring that bucket sprawl does not re-create the legacy global namespace exposure in newly provisioned infrastructure. Teams using AWS CloudFormation and CDK should migrate templates to use the new `BucketNamePrefix` and account-regional namespace parameters as a standard practice going forward.

On GCP, organizations should verify that soft delete is enabled with an adequate retention period on all production and logging-critical GCS buckets, and establish monitoring alerts for GCS bucket deletion events with latency low enough to enable response before the seven-day default soft-delete window expires. The Cloud Audit Log for bucket deletion should be routed to a monitoring destination that is not itself a GCS bucket—preventing the circularity where the audit log for a bucket deletion would also be lost if the alert bucket were the target.

Across all cloud providers, data perimeter controls should be configured to restrict telemetry and logging destinations to storage resources within the organization's verified account inventory. On AWS, this takes the form of S3 bucket policies with `aws:ResourceAccount` conditions and Service Control Policies enforcing that assumption, ensuring that CloudTrail and Firehose configurations cannot deliver data to buckets outside the organization's account hierarchy even if a sink configuration is manually modified to point elsewhere. On GCP, VPC Service Controls perimeters constrain Cloud Logging sink destinations to resources within the perimeter's scope, providing a similar structural enforcement mechanism [3].

Pre-deletion checklists are a control mechanism that is straightforward in concept but requires operational discipline to maintain. Before any cloud storage bucket is deleted, organizations should conduct a search for active references to the bucket name in code repositories, CI/CD configurations, infrastructure templates, DNS records, monitoring configurations, and telemetry routing rules. This search will not always be comprehensive—inherited references in compiled artifacts and third-party dependencies may escape it—but it provides a systematic opportunity to identify and remediate the most accessible categories of dangling reference before the name enters the global pool. Automation through CSPM integrations or pre-deletion approval workflows in infrastructure management tooling can make the checklist a mandatory rather than advisory control.

Strategic Posture

At the portfolio level, organizations should treat cloud storage bucket names as persistent identifiers that outlive the infrastructure they describe. A bucket name embedded in distributed code, templates, or configurations should be considered a durable organizational commitment to maintaining that bucket as an organization-owned resource—or actively retiring every known reference before deletion. The conventional treatment of cloud resources as ephemeral and disposable is operationally convenient but strategically problematic when applied to globally unique names with distributed reference footprints.

Organizations with significant software distribution responsibilities—commercial software vendors, open-source project maintainers, infrastructure tooling providers—carry a heightened obligation to audit and retire external storage references in their distributed artifacts. The bignum and watchTowr incidents both demonstrated that the harm from an abandoned bucket reference is not limited to the organization that created the reference; it propagates to every downstream user who installs the software, runs the template, or configures the VPN client. Software supply chain security programs should include bucket reference auditing as a standard practice alongside dependency pinning and SBOM generation.

Security teams should incorporate namespace hijacking scenarios into cloud threat modeling exercises. The STRIDE framework applied to cloud storage services typically focuses on information disclosure through misconfigured access controls and spoofing through credential theft—both of which are important—but underweights the structural risk that arises when an organization's storage resources are deleted and the

names are claimed by adversaries. Threat models that include the deletion-and-re-registration scenario will produce more complete control recommendations and may surface reference auditing requirements that would otherwise be missed.

CSA Resource Alignment

Cloud Controls Matrix

The Cloud Controls Matrix (CCM) provides a direct mapping framework for the controls most relevant to namespace hijacking risk. DSP-07 (Data Retention and Deletion) requires that organizations establish and enforce data classification, retention, and deletion policies. Applied to cloud storage infrastructure, this control should explicitly include procedures for validating that all references to a storage resource have been identified and retired prior to deletion—extending the scope of data lifecycle governance from the data itself to the infrastructure identifiers that locate it.

IAM-03 (Privilege Management) and IAM-14 (Privileged Access Management) are directly implicated in the active stream hijacking variant documented by Unit 42 [3]. The technique requires that an adversary hold bucket-deletion permissions. Organizations implementing least-privilege IAM should treat deletion permissions on logging and telemetry destination buckets as high-value entitlements warranting the same controls applied to administrative access—separate role assignments, multi-party approval for deletion operations, and periodic access review. Separation of the identity that manages logging configurations from the identity that holds deletion permissions on logging destinations provides a meaningful mitigation.

LOG-05 (Audit Log Protection) and LOG-06 (Log Review) are directly threatened by data stream hijacking attacks that divert audit logs to attacker-controlled storage. Organizations should ensure that the audit log infrastructure is itself subject to integrity monitoring and that logging destinations are validated against an organizational account inventory on a regular basis—not only configured once and left unmonitored.

TVM-09 (Threat Intelligence) supports the integration of namespace hijacking research findings into organizational risk assessments. The Checkmarx, watchTowr, Unit 42, and Aqua Security research all provide actionable intelligence about specific attack mechanisms, affected service configurations, and vulnerable patterns. Security teams should track this body of research and update their threat models and control frameworks as new findings emerge.

MAESTRO and Agentic AI Considerations

As organizations deploy AI agents that interact with cloud storage resources—for data retrieval, artifact management, model artifact storage, and pipeline orchestration—the namespace hijacking risk acquires additional dimensions relevant to CSA's MAESTRO threat modeling framework for agentic AI systems.

At MAESTRO Layer 2 (Hosting and Infrastructure), AI agents that write logs, model artifacts, or pipeline outputs to cloud storage buckets create telemetry reference patterns analogous to the logging sinks described in the Unit 42 research. An AI agent's data pipeline may write training logs, inference logs, or pipeline artifacts to an S3 bucket configured at deployment time. If that bucket is subsequently deleted and re-registered by an attacker, the agent's output—including potentially sensitive inference data, retrieved documents, or tool outputs—flows to attacker-controlled infrastructure without any error visible to the agent or its operator.

At MAESTRO Layer 6 (Data Pipeline Security), AI agents that retrieve data from cloud storage buckets as part of retrieval-augmented generation workflows or tool execution are vulnerable to a specific variant of the data poisoning attack: if the bucket providing tool resources or context data is hijacked, an adversary can serve malicious content that the agent processes as authoritative retrieved information. This is a variant of prompt injection via data pipeline manipulation, using the namespace hijacking mechanism as the delivery vector rather than direct prompt manipulation.

Organizations deploying agentic AI infrastructure should apply the same bucket reference auditing and pre-deletion control disciplines to AI-specific storage resources—model artifact buckets, agent logging sinks, retrieval corpus buckets—as they apply to general cloud infrastructure.

CSA STAR and Assurance Guidance

Organizations seeking to demonstrate cloud security assurance through the CSA STAR program should treat namespace hijacking controls as part of their evidence base for supply chain security and data governance attestations. The Cloud Security Assessment approach applied to storage infrastructure should include verification of pre-deletion procedures, audit of DNS records for dangling storage references, and documentation of controls restricting logging and telemetry destinations to organization-owned resources. These are verifiable, evidence-producible controls that align with STAR Level 1 (self-assessment) documentation and Level 2 (third-party audit) examination procedures.

Conclusions

Global cloud namespace hijacking is not a novel attack technique, but its full scope and systemic implications came into focus only through a convergence of research published in 2024 and 2025. The watchTowr supply chain demonstration, the Checkmarx bignum analysis, the Aqua Security CDK disclosure, and the Unit 42 active stream hijacking research collectively document a risk that spans all major cloud providers, multiple distinct attack paths, and a threat surface that extends backward through years of accumulated organizational history and forward through the continued growth of cloud-native software ecosystems.

The foundational problem—that cloud storage bucket names are globally unique but not persistently tied to the accounts that created them—is architecturally difficult to fully correct at the provider level. AWS's March 2026 account regional namespace feature represents a meaningful structural fix for newly created buckets, but it is prospective and opt-in only [6][7]. The legacy global namespace and all references to names deleted within it remain exposed. GCS soft delete provides a bounded protection window against the active stream hijacking variant but does not address the supply chain or infrastructure-as-code attack paths. Azure's account-scoped container naming and robust soft-delete protections provide the strongest structural mitigations of the three major providers, but do not eliminate the risk entirely.

For security practitioners, the practical implication is that cloud storage deletion is an irreversible act with a long tail of distributed consequences that cannot be fully inventoried at the time of deletion. Treating bucket names as durable identifiers—subject to the same lifecycle discipline applied to domain names, certificate common names, and service endpoints—is the appropriate conceptual shift. Pre-deletion auditing, account-scoped naming conventions, data perimeter enforcement for telemetry destinations, and systematic scanning for dangling DNS and configuration references are the concrete controls that translate that conceptual shift into organizational protection.

The watchTowr finding that eight million automated requests reached attacker-controlled infrastructure from government agencies, military networks, and Fortune 500 companies—in a demonstration that cost less than five hundred dollars to execute—illustrates how dramatically the offensive economics favor the attacker in this threat class. Parity requires organizations to treat abandoned bucket name risk as a standing posture question, not a one-time remediation project.

References

[1] watchTowr. "Abandoned S3 Buckets: Millions of Requests to Re-Registered Cloud Infrastructure." *The Register*, February 4, 2025.

[2] watchTowr / SecurityWeek. "Abandoned Amazon S3 Buckets Could Have Enabled Attacks Against Governments, Big Firms." *SecurityWeek*, February 2025.

[3] Palo Alto Networks Unit 42. "The Global Namespace Risk: Universal Bucket Hijacking Technique for Cloud Data Exfiltration." *Unit 42 Threat Research*, 2025.

[4] Aqua Security. "AWS CDK Risk: Exploiting a Missing S3 Bucket Allowed Account Takeover." *Aqua Security Blog*, 2024.

[5] The Hacker News. "AWS Cloud Development Kit Vulnerability Exposes Users to Potential Account Takeover Risks." *The Hacker News*, October 2024.

[6] AWS. "Amazon S3 Introduces Account Regional Namespaces for General Purpose Buckets." *AWS What's New*, March 12, 2026.

[7] AWS News Blog. "Introducing Account Regional Namespaces for Amazon S3 General Purpose Buckets." *AWS News Blog*, March 2026.

[8] Checkmarx. "Hijacking S3 Buckets: New Attack Technique Exploited in the Wild by Supply Chain Attackers." *Checkmarx Security Blog*, 2023.

[9] Google Cloud. "Soft Delete | Cloud Storage." *Google Cloud Documentation*, 2024.

[10] Google Cloud Blog. "Understanding Cloud Storage's New Soft Delete Feature." *Google Cloud Blog*, 2024.

[11] AWS Security Blog. "Threat Tactic Spotlight: Subdomain Takeover." *AWS Security Blog*, 2024.

[12] Char49. "Subdomain Takeover via Abandoned Amazon S3 Bucket." *Char49 Security Research*, 2024.

[13] Microsoft Azure. "Azure Storage Account Overview." *Microsoft Azure Documentation*, 2024.