ISSUE #010 · BLIND SPOTS

Blind Spots.

You defend what you can see. Three stories this week each take away a piece of that. Attackers turn your cloud logs off and then read the pipeline for their own recon. A GCP field your detection keys on shows up in the console and never reaches your SIEM. And Azure ransomware writes a log line that reads exactly like a normal upload.

> ATTACK

Yahav Festinger at Unit 42 · 2026-06-09 · **Blinding the Watchmen: Abusing Cloud Logging Services for Defense Evasion and Visibility**

You land a role with rights over the logging plane, and you have options before you touch anything noisy. Call `StopLogging` on the CloudTrail trail and the writes stop. Or `DeleteTrail`, or empty and delete the S3 bucket under it with `s3:DeleteObject` then `s3:DeleteBucket`. Prefer something quieter: repoint the trail at a KMS key you control with `update-trail`, then revoke CloudTrail's access to that key, and delivery fails so new logs stop landing at all. Quieter still, if the trail has log file integrity validation off, pull the log objects with `s3:GetObject`, edit the JSON to drop your own calls, and put them back with `s3:PutObject`. The GCP side is the same moves under different names: `DeleteSink`, `logging.buckets.delete`, a CMEK swap with `gcloud logging buckets update --cmek-kms-key-name`.

The half most defenders miss is the visibility flip. Broader logging permissions, the ones that create and reconfigure trails rather than only stop them, let you stand up a fresh trail or a logging sink that ships to a bucket you own, or repoint the existing trail's `--s3-bucket-name` with `update-trail`, and now the victim's control-plane activity streams to you. **Logging is a two-way mirror. The rights to reshape it are the rights to sit behind it and watch the response.**

▸ SHIP THIS WEEK

Lock down who can change the logging plane. Scope `cloudtrail:StopLogging`, `DeleteTrail`, `UpdateTrail`, `CreateTrail`, and `s3:DeleteBucket` / `s3:PutBucketPolicy` on the trail's bucket; on GCP, lock `logging.sinks.update`, `logging.sinks.delete`, and `logging.buckets.delete`. Turn on CloudTrail log file integrity validation so the quiet edit-and-replace move breaks the digest chain. The stop, delete, or re-key call is the canary, so alert on any of them as high severity, because in a healthy account they almost never fire. Watch for a second trail or sink you didn't create: a new sink pointed at an unfamiliar bucket is exfiltration wearing a logging label.

> DEFENDER

Art Ukshini at Permiso · 2026-06-16 · **Mind the Gap: GCP serviceData in Logs Explorer vs. Exported Logs**

Verify the field your GCP audit-logging detections depend on reaches your SIEM at all. `serviceData` is a deprecated GCP audit-log field, and for some services it still carries the only copy of the signal you need. When someone disables audit logging through `SetIamPolicy` on `cloudresourcemanager.googleapis.com`, the field that says this was a removal, the `policyDelta` with `action: REMOVE`, `service: allServices`, lives inside `serviceData.policyDelta`. That field renders in Logs Explorer. For this event, GCP drops it from the exported log record, so it never reaches your analytics platform. Same `insertId`, same method, the discriminating field gone. **For your engineer: take your top GCP detections, find each that reads a `serviceData` sub-field, and confirm that field survives export into your SIEM. Where it lands only in Logs Explorer, rebuild the rule on a field that survives.**

Aidan Steele · 2026-06-19 · **CloudTrail in CloudWatch isn't very good** — Know what you lose before you move CloudTrail off Lake

The same failure shows up in AWS. AWS deprecated CloudTrail Lake for new customers on June 1 and points you to CloudWatch instead. Steele ran the migration and found the CloudWatch destination drops enrichment the trail and Lake carried. Lake enriched events with resource tags and global condition keys, and that enrichment is gone in CloudWatch, so a rule scoped on a tag or an `aws:` condition key has nothing left to match. Events also land about 8 hours late where a trail delivers in minutes. **Before you migrate, list every detection that reads a tag or a condition key and confirm it still fires on the CloudWatch copy.**

> RULE OF WEEK

Jonah Feldman at Datadog Security Labs · 2026-06-15 · **Holding blobs for ransom: Four methods for Azure Storage ransomware**

Azure resource logs don't capture the `x-ms-encryption-key-sha256` header, so a customer-provided-key (CPK) `PutBlob` and an ordinary `PutBlob` look identical in the log. You can't key on the header, so key on the behavior around it. Alert when one principal overwrites a large number of existing blobs in a short window, and raise the severity when that burst pairs with control-plane moves that lock the data: a new encryption scope, a storage account default-encryption (CMK) change, or a Key Vault key or vault deleted minutes after the writes. **The bulk-overwrite-then-delete-the-key shape is the ransom, even when each individual `PutBlob` reads clean.**

> AGENT BENCH

The blind spots above live in your logs. The other one is in code you depend on and have never read. Reading a dependency's source for injection and unsafe deserialization by hand is slow, and most teams never get to it. An agent can run the first pass and hand you somewhere to start. Scope the target precisely: the repo URL, the exact commit SHA or release version, the language and framework, and the paths to exclude. Ask it to hunt the code-audit classes, injection, auth bypass, SSRF, path traversal, unsafe deserialization, and run secrets-in-code as a separate deterministic scan, since a grep-based scanner beats a reading pass for that. For every hit, require a source-to-sink trace: the reachable entrypoint, the attacker-controlled input, and the guard conditions the model believes are missing. A finding without that trace is a guess. **This is the oss-security-audit workflow. Agents fabricate reachability, invent control flow, and miss the mitigating check three lines up, so your verification is the gate. Try it this week: pick one dependency you've never read, pin the version, run the audit, and verify the top finding's source-to-sink trace yourself.**

> RADAR

Katie Knowles at Datadog Security Labs · 2026-06-11 · **Entra Agent ID: The blueprint blast radius**

Entra's new agent-identity model hangs many identities off one object. A blueprint is an app registration for AI agents, and it is the single trusted source of authentication for everything beneath it: a blueprint principal in each tenant, up to 250 agent identities per tenant under that, plus optional agent users. Add one credential to the blueprint and you can mint tokens that impersonate any identity it anchors, through a `client_credentials` exchange with an `fmi_path` naming the target. Where a blueprint deploys agents across tenants, that one credential reaches all of them, the shape of the Midnight Blizzard third-party compromise Knowles draws the line to. The blast radius of a single secret is the part you can't see from inside one agent. Worth tracking as agent identities multiply faster than the IAM model around them.

> RECON ROLES

Chainguard, Senior Security Engineer (AI Platform). US + Canada remote. A brand-new individual-contributor role that reads less like infrastructure security and more like governing the AI tooling the rest of the company runs on. The scope: administer Claude and ChatGPT at the org level, manage console settings through Git, run API key lifecycle, build anomaly detection for AI spend by team, write MCP servers and agentic tooling in Python or TypeScript, and keep sensitive data out of prompts. Five-plus years and hands-on enterprise Claude or ChatGPT administration expected, on GCP. The US listing posts $130,000 to $160,000 USD, modest for a senior security title at a company with Chainguard's name; the Canada listing shows no range. Worth a look if AI platform posture is the job you want before every org writes the same req. Verified open as of Mon 2026-06-22. Roles can close anytime. No affiliation or payment.

> SIGNOFF

The cloud doesn't hide things from you out of malice. The defaults do it: a deprecated field that stops at the console, a header that never reaches a log, a logging plane any sufficiently privileged role can switch off. Assume your visibility has holes, then go find which detection breaks when the one field you counted on never arrives. **Full post with source links: defensive.works/recon/issue-10**

SOURCES

Unit 42 / unit42.paloaltonetworks.com (Blinding the Watchmen, Yahav Festinger)
Permiso / permiso.io/blog (Mind the Gap: GCP serviceData) · Aidan Steele / awsteele.com
Datadog Security Labs / securitylabs.datadoghq.com (Azure Storage ransomware; Entra Agent ID)
Chainguard / job-boards.greenhouse.io/chainguard