S3 bucket misconfigurations.

Amazon S3 (Simple Storage Service) is the cloud storage AWS customers use for almost everything: website files, backups, logs, datasets. A misconfigured S3 bucket is still one of the most common ways company data ends up public. AWS has added safer defaults over the years, but new mistakes keep happening, because the permission settings are complex and one wrong line can expose everything.

HOW IT WORKS

01 What misconfigurations show up in production?

From engagements over the past year, the patterns we see most:

- Public read on a bucket that was never meant to be public. Often a one-line policy added for a quick job and never removed.
- Public list permission. Even if the files cannot be read, an `s3:ListBucket` open to everyone tells an attacker exactly what to ask for next.
- World-writable buckets. Rare now, but bad when found. An attacker can plant content that real users will load.
- Leaked pre-signed URLs. A pre-signed URL is a credential on its own. A leaked one is a public file until the URL expires.
- Cross-account access wider than intended. A policy that grants `arn:aws:iam::*:role/*` when the team meant one specific account.
- Replication to a weaker bucket. Data copied to a destination bucket with looser permissions than the source.
- Encryption turned off. The bucket accepts uploads without encryption, leaving data unprotected if the storage is reached another way.
- Unlabeled sensitive data. Production database backups sitting in a 'misc' bucket nobody has classified.

02 How do attackers find your S3 misconfigurations?

Three methods do most of the work.

HOW TO DEFEND

- Turn on Block Public Access at the account level. One setting that overrides bucket-level mistakes. On by default for new accounts since 2023; check older ones.
- Start every bucket with deny-all. Then grant only the minimum each user needs.
- Inventory and label your buckets. A tool like AWS Macie, or your own classifier, finds buckets holding sensitive data and flags them for tighter controls.
- Require encryption. Enforce server-side encryption on every upload through the bucket policy.
- Check continuously. A posture-monitoring tool catches drift, like someone turning Block Public Access off for a quick task and forgetting. Pair it with a recurring pentest that confirms the controls really hold.

SOURCES

[1] AWS S3 Block Public Access

[2] AWS S3 Security Best Practices

[3] CIS AWS Foundations Benchmark

- Guessing bucket names. S3 names are global. Attackers brute-force common patterns from your company name, brand, environment, and region (`acme-prod-backups`, `acme-dev-logs`, `acme-uploads`). DNS answers whether the name exists, so they quickly learn where to look.
- Public code and forums. Developers paste bucket names into examples, post errors that leak them, or commit them to public repos.
- Referrer leaks. A bucket used to host images often shows up in HTTP referrer headers when those images load on other sites.

Once a bucket is found, the attacker tries the basics: anonymous read, list, and write. The whole check is automated and takes seconds per bucket.

03 How does SecureLayer7 test S3 configurations?

Every AWS engagement runs four passes over your buckets.

- Inventory. List every bucket the account owns, with its policy, ACL, Block Public Access setting, encryption defaults, replication targets, and a sample of how sensitive its contents are.
- External probe. From outside the account, try anonymous read, list, and write on each bucket. Confirm what is actually exposed.
- Cross-account probe. From a separate AWS account we control, try the same, to test for over-broad cross-account trust.
- Content check. For any readable bucket, sample the files and classify them (personal data, credentials, backups, source code). Findings include the data type, so you can prioritize.

The report maps findings to the CIS AWS Foundations Benchmark, with a fix for each bucket.

Audit your S3 inventory end to end.
`securelayer7.net/learn/cloud-security/s3-misconfig`

Open online