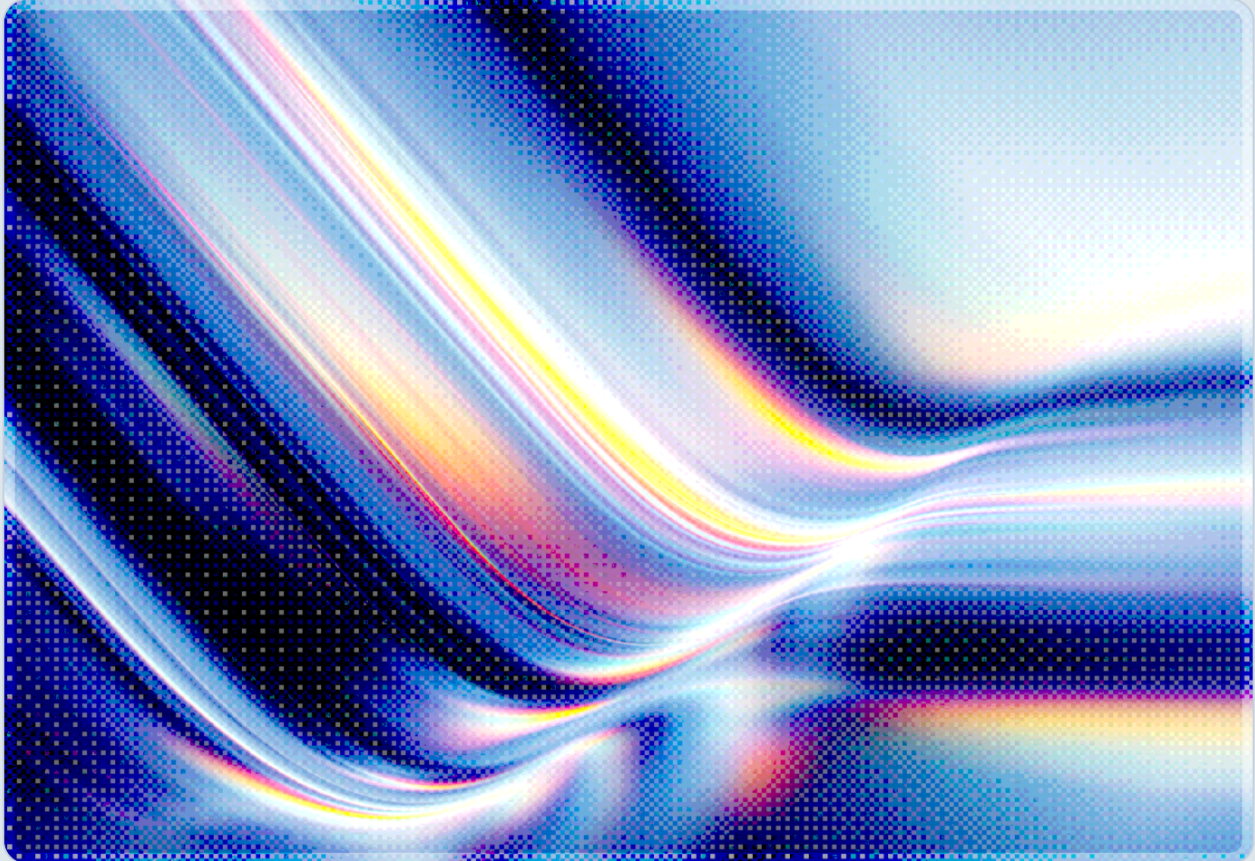


543,699 Still-Valid GitHub Credentials: Implications for Agent- Era Risk

Security Implications and Guidance (credentials tested July
2026)

2026-10-05

 AI-assisted Rapid Research



© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

Truffle Security reported on October 1, 2026 that 543,699 unique credentials found in public GitHub repositories still authenticated successfully when it tested them on July 27-28, 2026 [1][2]. The median working credential had been public for 784 days, and about one in ten was more than 6.3 years old, so many of these secrets have been exposed for years rather than being freshly leaked [1]. Because the tests ran more than two months before publication, some of these credentials may have been rotated since.

GitHub's default push protection appears to have reduced exposure of the secret types it covers by roughly 53 percent, but 51.8 percent of the live credentials belong to formats it does not block, including database connection strings and Google API keys [1][2]. Survival rates also varied enormously by credential type, from 0.001 percent for npm tokens to 88 percent for Postgres connection strings, and the authors and press coverage attribute much of that spread to whether the issuing provider revokes leaked secrets automatically [1][2].

The figures describe exposure, not observed misuse. The claim that AI agents could harvest this pool is our inference from agent capabilities and from documented credential-stealing malware, and it should be read as a risk assessment rather than a measured trend.

Background

On October 1, 2026, Truffle Security published an analysis of public GitHub code that it describes as finding 543,699 working credentials that nobody had revoked [1]. The researchers worked from The Stack v3, a public code corpus of roughly 58.4 billion files drawn from 224,553,295 repositories, with a crawl that runs through August 7, 2025 [1][2]. They matched candidate secrets by pattern and then tested them against the issuing services on July 27-28, 2026, about eleven months after the crawl [1]. The resulting 543,699 unique credentials appeared in 1,103,438 separate exposures, because the same secret is often copied across files and forks [1][2].

The method has limits that affect how the number should be read. The corpus covers default branches only, so secrets that were force-pushed away, reverted, or committed to other branches are outside the sample, and Truffle Security says the real population is therefore larger [1]. Several categories were also

excluded because they cannot be verified or are not sensitive: bare hex or base64 strings, private keys that need host infrastructure to test, and Google API keys that only worked for non-sensitive services such as Maps or Firebase [1]. The figure is a lower bound on a specific slice of GitHub as it existed in August 2025, checked against provider state eleven months later.

The result sits within a broader pattern. GitGuardian's State of Secrets Sprawl 2026 report counted 29 million new hardcoded secrets in public GitHub commits during 2025, a 34 percent increase over the prior year, and found that 64 percent of valid secrets leaked in 2022 had still not been revoked [3][8]. It also reported that incidents tied to AI services rose 81 percent year over year, that eight of the ten fastest-growing secret categories were AI-related, and that Model Context Protocol (MCP) configuration files exposed 24,008 unique secrets in their first year of use [3][8]. The two datasets measure different things, new leaks per year in one case and surviving valid credentials in the other, but both are consistent with the view that leaked secrets are being created faster than they are retired.

Security Analysis

What is in the pool

The composition of the 543,699 credentials indicates what access an attacker could gain. The largest single category is Google Cloud service account credentials at 69,041, followed by 51,067 MongoDB connection strings and 33,343 Google API keys, of which 31,374 are Gemini keys [1][2]. Service account keys and database connection strings can grant direct access to cloud resources or data stores, depending on their scope and network reachability, and Gemini keys let a holder consume paid model inference at the owner's expense. The dataset does not show how many of the 543,699 carry meaningful privileges. The oldest verified credential dates to 2009, reportedly an AWS key that still worked in 2026, and 2,636 live credentials predate 2015 [1][2].

Credential type	Live count	Note
Google Cloud service account	69,041	Largest category; 54% survival rate after exposure [1]
MongoDB connection string	51,067	Not covered by default push protection [1]
Google API key	33,343	Of which 31,374 are Gemini keys [1][2]

Credential type	Live count	Note
All verified credentials	543,699	1,103,438 exposures including duplicates [1]

Why the credentials stay alive

The age profile suggests that detection is not the main problem. The median working credential had been exposed for 784 days, and a tenth were more than 6.3 years old [1]. GitHub's secret scanning has alerted issuers for public repositories since February 2023, yet 245,959 of the live credentials predate that program, and a further 97,897 were committed while push protection was available but optional [1]. Of the total, 199,843 credentials, or 36.8 percent, were committed after push protection became the default on February 29, 2024 [1][2]. Truffle Security's comparison suggests that push protection reduced exposure of covered types, measuring a 53 percent reduction after the rollout against a 7 percent decline for formats it does not cover [1]. Even so, 51.8 percent of the live credentials fall into the uncovered group, which includes database connection strings, Google API keys, and private keys [1].

A second finding concerns revocation. Survival rates are strongly associated with whether the issuing provider revokes leaked secrets automatically. Only 0.001 percent of exposed npm tokens and 0.36 percent of GitHub tokens still worked, whereas 54 percent of Google Cloud service accounts, 75 percent of MySQL strings, and 88 percent of Postgres strings remained valid [1]. Truffle Security and SecurityWeek attribute the difference to revocation pipelines, noting that GitHub alerts issuers but does not require them to revoke [1][2]. The dataset does not isolate that mechanism, and token expiry, credential design, and provider policy beyond secret scanning may also contribute. In Truffle Security's words, push protection "stops secrets at the door" but "has nothing to say about the 543,699 already inside" [2].

Why agents may change the economics

Credential theft from public repositories is not new, and automated scanners have long watched GitHub for fresh commits. What capable AI agents could change is the cost of the later steps. A stale credential is only useful if someone can work out what it unlocks, test it safely, chain it to other resources, and act within the available permissions. For a service account key that means enumerating a cloud project, and for a MongoDB string it means finding and interpreting the data. Post-authentication steps of this kind have generally required more human effort than the initial validity check, which tools already automate, and we infer that this effort has acted as a practical constraint on exploiting low-profile credentials. An agent that can read documentation, adapt to an unfamiliar API, and iterate on errors would lower the

marginal cost of working through a long tail of such credentials. We regard this as a plausible shift in attacker economics, but we are not aware of published measurements of agent-driven harvesting of this specific pool, and it should be treated as an inference.

Adjacent evidence shows that credential theft and reuse can already be chained automatically. CSA's research on the Miasma and IronWorm npm worms documents automated worms that harvest a related set of credentials, including API keys for several AI providers, npm publish tokens, GitHub write access, and CI/CD OIDC tokens, and then use them to spread to additional packages and repositories without further operator action [4]. That campaign worked from compromised developer environments rather than from public repositories, and its credential types overlap only partly with the pool described here. We have found no evidence that Truffle Security's verified list has been published, but the underlying corpus is public and the method is reproducible, so an actor could assemble a comparable pool without compromising any developer machine.

AI coding assistants also bear on the exposure side of this problem. GitGuardian reported that commits co-authored by Claude Code leaked secrets at a 3.2 percent rate against a 1.5 percent baseline [8][9]. This is one vendor's measurement of one tool, and the sample sizes and controls should be checked in the primary report before the number is generalized, but it would be consistent with the concern that agent-assisted development can increase the rate at which new credentials reach repositories. Secrets held in an agent's own configuration are likely to appear in future leaks, which is why the MCP configuration findings above deserve attention.

Limits of the evidence

Several caveats apply to any operational conclusion. The dataset is a snapshot of default branches, so it neither measures how many of these credentials have been used by third parties nor ranks them by the damage they could do, and a credential that "still works" may be scoped to a sandbox or a free tier. The claim that the pool is attractive to agent-driven attackers is plausible, but it rests on capability reasoning plus adjacent incidents rather than on direct observation. Organizations should treat the 543,699 as evidence about how long exposed secrets survive and about the gaps in provider revocation, and treat the agent-harvesting scenario as a planning assumption.

Recommendations

Immediate Actions

Treat any credential that has ever been committed to a public repository as compromised, and rotate it before cleaning up the repository. Removing a file or rewriting history does not undo exposure, because forks, caches, and mirrors persist. Organizations should run a full-history scan of every repository they own or have owned, including forks and archived projects, rather than relying on current branch contents. This matters in particular for anything committed before February 2024, since those commits never passed through default push protection [1]. Priority should go to the categories that make up the largest share of the live pool and that default push protection does not cover, namely database connection strings and Google API keys, together with cloud service account keys, which are the largest category in the pool [1].

Organizations should also review the credentials held by AI coding assistants, agent frameworks, and MCP servers. Configuration files for these tools were already a notable source of leaks in GitGuardian's 2026 data [3][8], so scanning for them in both repositories and developer workstations is a reasonable early step.

Short-Term Mitigations

Enable secret scanning and push protection across every organization repository, and extend coverage to non-provider patterns by adding custom patterns for internal token formats and database connection strings. Because default push protection does not block more than half of the live credentials in Truffle Security's sample, teams should add a pre-commit or CI scanner that validates findings against the issuing service, which cuts false positives and prioritizes secrets that still work [1]. Where possible, replace long-lived static secrets with short-lived credentials issued through workload identity federation or OIDC, so that a leaked value expires on its own.

Ownership matters as much as tooling. Inventory the non-human identities behind each credential, including the agent or pipeline that uses it and the owner who can approve revocation, so that rotation does not stall on unclear responsibility. For credentials issued by your own organization, build or adopt an automated revocation path and register with GitHub's secret scanning program where applicable, because the survival data suggest that automated revocation, more than detection alone, shapes how long a leaked secret remains valid [1][2].

Strategic Considerations

Providers and platform operators carry part of this problem. The gap between npm and GitHub tokens, which almost never remain valid, and database strings, which largely persist, indicates that a revocation pipeline can work at scale where one exists [1]. Enterprises that buy or build services that issue credentials should treat automatic revocation on public exposure as a design criterion, with a defined revocation window, and should favor vendors who offer it. Security teams should plan for a threat model in which automated agents test every discovered credential quickly and at low cost, which argues for least-privilege scoping, network-level restrictions such as IP allow lists and private endpoints, and anomaly monitoring on high-privilege machine identities.

Finally, measure secrets hygiene with outcome metrics, such as median time from exposure to revocation and the share of leaked secrets still valid after 30 days, rather than counting alerts alone. GitGuardian found that 64 percent of valid secrets leaked in 2022 had still not been revoked [3][8]. That statistic concerns revocation rather than alert volume, but we infer from it that alert counts without remediation tracking can leave a false sense of progress.

CSA Resource Alignment

CSA's research note on the Miasma and IronWorm npm worms is the closest prior work on how stolen AI and developer credentials are put to use by automated tooling. It documents the harvesting of AI provider keys and CI/CD tokens and the practice of treating AI tool configuration directories as trust boundaries, which connects directly to the recommendation above to scan and restrict credentials held by coding assistants and agents [4].

CSA's analysis of the CISA "Private-CISA" repository exposure addresses a closely related failure: a public GitHub repository maintained by a contractor exposed administrative credentials for cloud servers, and the keys reportedly stayed valid for roughly 48 hours after the repository was taken offline [5]. It illustrates the pattern in the Truffle Security data, where exposure and revocation are separate events, and it adds the dimension of third-party and contractor governance.

CSA's whitepaper on non-human identity and agentic AI governance supplies the identity-management framing for the strategic recommendations, including inventories of non-human identities, credential rotation, and rapid revocation [6]. For control mapping, the AI Controls Matrix (AICM) v1.1, a superset of the Cloud Controls Matrix, includes controls in the Identity and Access Management and Cryptography, Encryption and Key Management domains that apply to secret lifecycle management, and the Application and Interface Security domain that applies to code-level secret hygiene [7].

References

- [1] Truffle Security. "[543,699 Credentials Still Working on GitHub, and Nobody Revoked Them](#)." Truffle Security Blog, October 1, 2026.
- [2] SecurityWeek. "[500,000 Active Credentials Left Exposed on GitHub](#)." SecurityWeek, October 1, 2026.
- [3] The Hacker News (sponsored content from GitGuardian). "[The State of Secrets Sprawl 2026: 9 Take aways for Security Leaders](#)." The Hacker News, March 2026.
- [4] Cloud Security Alliance. "[Miasma and IronWorm: Self-Replicating Worms Targeting AI Credentials](#)." CSA Labs, June 2026.
- [5] Cloud Security Alliance. "[Private-CISA: GovCloud Leak and the Hollowing of U.S. Cyber Defense](#)." CSA Labs, May 2026.
- [6] Cloud Security Alliance. "[Non-Human Identity and Agentic AI Governance](#)." CSA Labs, May 2026.
- [7] Cloud Security Alliance. "[AI Controls Matrix \(AICM\) v1.1](#)." Cloud Security Alliance, June 22, 2026.
- [8] GitGuardian. "[The State of Secrets Sprawl 2026](#)." GitGuardian (PDF mirrored by the Non-Human Identity Management Group), March 2026.
- [9] SC World. "[AI coding assistants twice as likely to leak secrets as overall leaks rise 34%](#)." SC World, March 2026.