Zombie ZIP: Archive Metadata Desync Defeats AV at Scale

CVE-2026-0866 Exposes a Structural Blind Spot Across 65 of 66 Security Engines on VirusTotal

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-12

Key Takeaways

A newly disclosed technique called "Zombie ZIP" (CVE-2026-0866) exploits a structural inconsistency in ZIP archive metadata to conceal malicious payloads from antivirus and endpoint detection and response (EDR) engines. By declaring an incorrect compression method in the archive's Local File Header while encoding the actual payload with a different algorithm, an attacker can deliver content that security scanners read as unintelligible compressed noise rather than recognizable malware signatures. Testing against VirusTotal on the date of public disclosure found that only one of 66 scanning engines detected the technique — an evasion rate of approximately 98% [1][2].

The vulnerability was disclosed by Christopher Aziz of Bombadil Systems on March 9–10, 2026, following coordinated vendor notification through CERT/CC beginning January 12, 2026 [3]. Kingsoft was the sole engine to detect the proof-of-concept payload; all major commercial security products — including Microsoft Defender, Avast, Bitdefender, ESET, Kaspersky, Sophos, and Trend Micro — failed to flag the malformed archive [1][2]. As of the date of this publication, no vendor patch or remediation update from a major AV or EDR provider had been publicly announced [3].

Zombie ZIP is not an exploit in the traditional sense. The malformed archive cannot be opened by standard tools — 7-Zip, WinRAR, and Python's `zipfile` library all fail with errors — which means it requires a purpose-built custom loader on the attacker's side to execute the payload. This characteristic limits widespread commodity malware adoption in the immediate term but does not diminish its relevance to sophisticated, staged delivery campaigns by advanced threat actors. The technique requires as few as six lines of standard `zlib` code on the decoding side, making integration into existing loader frameworks relatively low-effort. Furthermore, the underlying vulnerability class has direct precedent going back to 2004 without systemic remediation [4], and the 60-day coordinated disclosure window elapsed with virtually no substantive public vendor response.

Security teams should treat this disclosure as a signal about a class of risk — not a single CVE to patch and move on. Archive-based delivery has become a primary malware staging mechanism precisely because it offers multiple layers of obfuscation control, and Zombie ZIP joins a growing inventory of techniques that exploit parser discrepancies in widely deployed archive handlers.

Background

The ZIP Format and the Trust Problem

The ZIP format, originally released by Phil Katz in 1989, stores metadata about each archived file in two locations: a Local File Header (LFH) immediately preceding the compressed data for each file, and a Central Directory (CD) record appended at the end of the archive [5]. The CD is the authoritative index used by standard archive tools to enumerate contents, while the LFH is the record that a decompressor consults to determine how to handle the bytes that follow. Most conformant ZIP implementations treat these records as consistent; many security scanners rely on the CD or LFH metadata to determine how to parse content rather than independently verifying the actual data structure.

This trust assumption is the architectural basis for the Zombie ZIP technique. The ZIP specification requires that the `Compression Method` field in the LFH accurately describe how the following data was encoded. A value of `0` indicates that data is stored verbatim — no compression, raw bytes. A value of `8` indicates DEFLATE compression. When these fields are consistent, everything works as expected. When they diverge, the behavior of different ZIP consumers varies widely, and that variance creates a gap that defenders can fall into.

Prior Art and a 22-Year Pattern

The vulnerability class that Zombie ZIP inhabits is not new. CERT/CC documented related archive scanning weaknesses in 2004 (VU#968818) [12], and CVE-2004-0935 specifically identified that ESET NOD32 failed to scan certain malformed archives under similar conditions [4]. SANS ISC analyst research characterized CVE-2026-0866 as "a new primitive — method field desynchronization — within the same vulnerability class" as the 2004 disclosures, noting that the underlying design flaw — trusting archive metadata without independent validation of actual content structure — has persisted across two decades of security product development without systemic remediation [4]. That continuity is significant: it suggests that metadata trust is treated as a performance optimization or standards-compliance assumption baked deeply into scanner architectures, not an edge case that individual vendors are likely to address in isolation without external pressure.

Security Analysis

The Zombie ZIP Mechanism

The Zombie ZIP technique exploits three interrelated structural inconsistencies in a crafted ZIP archive [1][2][4]. The first is a compression method mismatch: the archive's LFH declares `Compression Method = 0` (STORED), instructing any consumer reading that field to treat the following bytes as uncompressed, raw data, while the actual payload is encoded using DEFLATE compression (Method 8). When an AV scanner reads the LFH and sees `Method = 0`, it scans the raw bytes as-is. Those bytes are DEFLATE-compressed binary data — effectively structured noise — which contains no recognizable plaintext signatures, strings, or PE headers.

The second inconsistency involves the CRC-32 integrity field. ZIP archives include a CRC-32 checksum for each file; in a Zombie ZIP, this checksum is set to match the decompressed payload rather than the raw DEFLATE-encoded bytes. When a standard ZIP library attempts to open the file — reading it as STORED per the declared method — the checksum fails to validate against the raw bytes, and the library throws an error. This error behavior serves dual purposes: it prevents automated sandbox detonation environments that rely on standard decompression libraries from successfully unpacking the payload, and it reinforces the appearance that the file is simply corrupt rather than malicious.

The third indicator is a size field discrepancy. In a legitimately STORED ZIP entry, the compressed and uncompressed size fields must be equal because no compression is applied. Zombie ZIPs present differing values — for example, 70 bytes compressed against 68 bytes uncompressed [11] — which is a reliable static heuristic that security scanners could, in principle, use for detection. The near-universal failure to detect this inconsistency indicates that the overwhelming majority of current scanners do not perform this consistency check [2][3].

A custom loader developed by the attacker — the public proof-of-concept requires approximately six lines of standard `zlib` code — ignores the declared method field and passes the raw bytes directly to a DEFLATE decompressor, successfully recovering the payload. The malware is not stored in any form that current AV signatures can reliably recognize; the DEFLATE compression layer ensures that bytes on disk bear no resemblance to the executable in memory as seen by scanners relying on signature-based detection [1].

Evasion Results and Vendor Landscape

Testing of the public proof-of-concept archive against VirusTotal on the day of disclosure produced a stark contrast [1]:

Sample	Description	Detection Rate
`baseline.zip`	Structurally valid ZIP containing the same payload	55/67 engines
`method_mismatch.zip`	Zombie ZIP – identical payload, desynchronized header	1/66 engines

Note: One engine that successfully processed the baseline archive returned an error – rather than a clean or detection verdict – on the Zombie ZIP sample and is excluded from the malformed-archive denominator, accounting for the one-engine difference between denominators.

Kingsoft, a Chinese antivirus vendor, was the only engine to detect the Zombie ZIP sample. All other major commercial security products – including Microsoft Defender, Avast, Bitdefender, ESET, Kaspersky, McAfee/Trellix, Sophos, and Trend Micro – returned clean results [1][2]. CERT/CC's coordinated disclosure advisory listed 28 additional vendors as "unknown" status, indicating that no public statement had been received from those organizations in the two months between initial vendor notification and public disclosure [3].

Cisco, the developer of the widely deployed ClamAV open-source scanner, acknowledged awareness of the issue in its CERT/CC response but classified it as "not considered a vulnerability, but rather a hardening suggestion" for future releases, stopping short of committing to a CVE-tracked remediation [3]. This classification is notable because it means Cisco may not issue a CVE-tracked fix, and organizations with contractual SLAs tied to vulnerability response should assess whether this outcome aligns with their security requirements. No confirmed patches from major AV or EDR vendors had been announced publicly as of the date of this publication.

Threat Actor Relevance and Deployment Context

Zombie ZIP is not a technique available to commodity malware authors in its current form. A ZIP archive that fails to open with 7-Zip, WinRAR, unzip, and Python's `zipfile` library cannot be deployed through simple phishing attachments or drive-by downloads that rely on the victim's standard tooling to unpack it. Successful weaponization requires a separately delivered custom loader that is capable of

parsing the malformed archive and decompressing its contents — a staged delivery architecture (a two-phase approach where a first-stage dropper enables delivery and execution of a separate final payload) more consistent with targeted intrusion campaigns than mass-distribution malware [2].

This constraint is meaningful but not reassuring. Advanced persistent threat actors routinely deploy multi-stage loaders, and the low implementation complexity of the decoding side — standard `zlib` DEFLATE with one field ignored, achievable in approximately six lines of code — makes integration into existing loader frameworks straightforward. The technique is best understood as a payload-staging primitive: a method of storing a final-stage implant on a compromised host or delivering it through a controlled channel in a form that post-compromise AV scanning will not identify. Given that many organizations run AV or EDR scanning on file shares, email attachments, and cloud storage as a secondary detection layer after initial access, the ability to store an implant in a form that evades secondary scanning represents a meaningful capability advantage for a threat actor already inside the perimeter, particularly in environments that rely on file-share or cloud storage scanning as a post-access detection layer.

The 98% evasion rate against VirusTotal also means that hash-sharing and threat intelligence feeds will not provide early warning at scale. A baseline Zombie ZIP sample containing a known payload will receive clean verdicts from 65 of 66 engines, so defenders cannot rely on reputation-based controls to catch variants until signatures are explicitly developed for the structural anomaly rather than the payload.

Archive Evasion as an Expanding Attack Surface

The past 18 months have produced several high-profile examples of archive format manipulation as a malware delivery and evasion primitive, suggesting the technique is gaining traction among threat actors. Zombie ZIP should be understood in this context, as the examples below illustrate a pattern of deliberate exploitation of ZIP parser variance across multiple distinct threat actor operations.

GootLoader malware, analyzed in January 2026, pioneered a related but distinct approach: concatenating 500–1,000 ZIP archives into a single file and deliberately truncating the End of Central Directory record by two bytes. WinRAR and 7-Zip reject the resulting file; Windows Explorer's built-in ZIP handler is permissive enough to extract it, delivering a JavaScript payload. GootLoader's operators additionally randomize non-critical archive header fields per-victim — a technique called "hashbusting" — ensuring every delivered file carries a unique hash and defeats signature-based detection entirely [6].

The Head Mare threat group has been documented using polyglot files simultaneously valid as both a Windows PE executable and a ZIP archive, embedding the PhantomPyramid backdoor in an archive that also contains a harmless ZIP visible to casual inspection [13]. Because ZIP central directory records are located at the file's end, arbitrary data prepended to a valid ZIP is ignored by ZIP parsers while remaining

executable by the operating system loader [7]. Most AV engines appear to scan the first identified format rather than treating the file as potentially multi-format, based on observed detection failures against polyglot files [7].

Additional recent precedents in the archiver vulnerability category include CVE-2025-0411 (7-Zip, bypassing Mark-of-the-Web propagation to extracted files, January 2025), CVE-2025-31334 (WinRAR, Mark-of-the-Web bypass via symbolic link, 2025), and CVE-2024-8811 (WinZip, Mark-of-the-Web stripping from archive and extracted files, 2024) [8][9][10]. Each of these vulnerabilities reflects the same underlying pattern: archive handling software is complex, the specification leaves implementation details to vendor discretion, and the attack surface created by that complexity continues to yield high-severity CVEs and evasion techniques, suggesting that security investment in archive parser hardening has not kept pace with exploitation. The persistence of CERT/CC VU#968818's core finding from 2004 [12] through CVE-2026-0866 in 2026 underscores that pattern.

Recommendations

Immediate Actions

Security teams should not wait for vendor patches before taking action. The Zombie ZIP mechanism introduces a structurally detectable anomaly — a STORED-declared ZIP entry where compressed and uncompressed sizes differ — that can be detected independently of signature databases. Organizations with the capability to deploy custom YARA rules or similar content-matching logic in their detection stack should develop and deploy a rule targeting the specific structural inconsistency: a LFH declaring `Compression Method = 0` combined with unequal `Compressed Size` and `Uncompressed Size` fields. This structural heuristic will produce some false positives on legitimately malformed archives but provides meaningful coverage in the absence of vendor-native detection.

Security operations teams should also revisit the alert triage logic applied to archive files that generate "unscannable" or "corrupted archive" results from their scanning infrastructure. In the absence of Zombie ZIP-specific detection, a scanner that encounters this file type will either return a clean verdict (because it scans the DEFLATE bytes as uncompressed noise and finds no signatures) or generate an error (because the CRC-32 validation fails). Both outcomes currently result in the file being passed through. A policy of quarantining or manually triaging files that generate archive parsing errors from security tooling — rather than treating parsing errors as equivalent to clean verdicts — would reduce the blind spot in the near term.

Short-Term Mitigations

At the platform and gateway level, security teams should review the archive inspection depth and error-handling behavior of their email security, web proxy, and cloud storage scanning tools. Vendors that have not yet issued guidance on CVE-2026-0866 should be contacted directly to request confirmation of their detection status and expected remediation timelines. Organizations with contractual SLAs tied to vulnerability response should assess whether the coordinated disclosure timeline — 60 days between vendor notification and minimal substantive public response from most major vendors, despite at least one vendor (Cisco) explicitly declining to classify the issue as a vulnerability — is consistent with their security requirements.

For cloud environments specifically, teams should audit any automated pipelines that accept or process user-uploaded archives. An attacker who can deliver a Zombie ZIP to a file processing endpoint in a cloud application may be able to trigger execution of the payload in a context where the surrounding cloud infrastructure provides additional attack surface. Content inspection at cloud ingress points should be augmented with structural validation that goes beyond metadata trust.

Network-level controls should be reviewed to ensure that archive files delivered through channels outside of email — including chat platforms, cloud storage sharing links, and developer artifact repositories — pass through inspection infrastructure rather than bypassing it. The staged delivery requirement for Zombie ZIP means that a first-stage dropper must reach the target before the malformed archive can be deployed, which creates network visibility opportunities that organizations should ensure they are capturing.

Strategic Considerations

The Zombie ZIP disclosure should prompt a broader assessment of dependency on metadata-trust assumptions in security tooling. The two-decade persistence of the underlying vulnerability class — from CERT/CC VU#968818 [12] in 2004 to CVE-2026-0866 in 2026 — reflects a systemic pattern in how AV and EDR products are built and validated. The near-universal detection failure suggests that products are typically tested against well-formed archives containing known-malicious payloads, with malformed archives treated as edge cases outside scanning scope rather than as an evasion surface requiring dedicated hardening. Until security products are routinely tested against structurally inconsistent or malformed archives as part of their acceptance criteria, this class of evasion will continue to yield high success rates against the commercial scanner market.

Organizations evaluating endpoint security products should specifically ask vendors about their approach to malformed archive handling. Questions worth posing include: Does the product validate compression method consistency between LFH and CD records? Does it detect size field inconsistencies

in STORED-method entries? Does it treat archive parsing errors as a security signal or as a scanning exception? What is the product's response timeline for CVE-2026-0866 specifically? Vendors that cannot answer these questions directly or that have no public advisory on file should be assessed as lower maturity from an archive evasion coverage perspective.

CSA Resource Alignment

MAESTRO: Agentic AI Threat Modeling

The Cloud Security Alliance's MAESTRO framework for agentic AI threat modeling identifies file and artifact integrity as a concern in environments where AI agents process user-submitted content. An AI agent configured to process uploaded archive files — for example, a code analysis agent or a document processing pipeline — that relies on downstream AV scanning as a trust gate is exposed to Zombie ZIP in precisely the way described above: the scanning layer returns a clean result, and the agent proceeds to process content that may contain malicious payloads. MAESTRO's guidance on input validation at agent boundaries applies here: AI-mediated file processing pipelines should not treat AV scan results as a binary trust signal without understanding the structural limitations of the scanning approach.

CCM: Cloud Controls Matrix

The CSA Cloud Controls Matrix addresses threat and vulnerability management in control domain TVM-01 through TVM-10. Relevant controls include: TVM-02 (Malware Protection), which calls for anti-malware tools that are regularly updated to address new threats; TVM-05 (Vulnerability Management), which requires timely remediation of identified vulnerabilities; and TVM-07 (External Vulnerability Disclosures), which addresses monitoring of external disclosures. CVE-2026-0866 falls squarely within the TVM-02 and TVM-07 scope, and the absence of vendor patches for major AV products creates a gap that CCM-aligned organizations should formally document in their risk registers pending remediation.

Zero Trust Guidance

CSA Zero Trust guidance emphasizes that file inspection and content validation should not be treated as a one-time gate at the network perimeter but as a continuous verification applied at each processing step. Zombie ZIP's ability to evade perimeter scanning reinforces this principle: organizations that perform AV scanning only at email ingress or network boundary may have no additional visibility into a

Zombie ZIP that reaches a file share, cloud storage bucket, or endpoint. Layered inspection at the endpoint, in the cloud workload, and in application-layer processing pipelines is consistent with Zero Trust principles and provides defense-in-depth against techniques designed to evade any single scanning layer.

AI Organizational Responsibilities

CSA's AI Organizational Responsibilities guidance calls for organizations deploying AI systems to maintain awareness of security threats specific to AI-integrated workflows. To the extent that organizational AI deployments include file ingestion pipelines — code analysis, document summarization, data processing — the Zombie ZIP technique represents a concrete threat that should be addressed in AI system security assessments. AI pipelines that trust AV scan verdicts as a precondition for processing user-submitted files should be reviewed in light of this disclosure and updated to include structural archive validation as a supplementary check.

References

[1] C. Aziz (Bombadil Systems), "Zombie ZIP — PoC Repository and VirusTotal Results," GitHub, March 9, 2026. https://github.com/bombadil-systems/zombie-zip

[2] BleepingComputer, "New 'Zombie ZIP' technique lets malware slip past security tools," March 10, 2026. https://www.bleepingcomputer.com/news/security/new-zombie-zip-technique-lets-malware-slip-past-security-tools/

[3] CERT/CC, "Vulnerability Note VU#976247 — ZIP archive metadata desynchronization allows malware to evade antivirus scanning (CVE-2026-0866)," Carnegie Mellon University Software Engineering Institute, March 2026. https://www.kb.cert.org/vuls/id/976247

[4] SANS Internet Storm Center, "Analyzing 'Zombie ZIP' Files (CVE-2026-0866)," ISC Diary, March 11, 2026. https://isc.sans.edu/diary/32786

[5] PKWARE Inc., "APPNOTE.TXT — .ZIP File Format Specification," Version 6.3.10, 2022. https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT

[6] The Hacker News, "GootLoader Malware Uses 500–1,000 Concatenated ZIP Archives to Evade Detection," January 2026. https://thehackernews.com/2026/01/gootloader-malware-uses-5001000.html

[7] WNE Security, "What Are Polyglot Files and How Do Hackers Use Them," 2025. https://wnesecurity.com/what-are-polyglot-files-and-how-do-hackers-use-them/

[8] NIST National Vulnerability Database, "CVE-2025-0411 — 7-Zip Mark-of-the-Web bypass," January 2025. https://nvd.nist.gov/vuln/detail/CVE-2025-0411

[9] NIST National Vulnerability Database, "CVE-2025-31334 — WinRAR Mark-of-the-Web bypass via symbolic link," 2025. https://nvd.nist.gov/vuln/detail/CVE-2025-31334

[10] NIST National Vulnerability Database, "CVE-2024-8811 — WinZip Mark-of-the-Web stripping from archive and extracted files," 2024. https://nvd.nist.gov/vuln/detail/CVE-2024-8811

[11] Iron Castle Systems / IBVL, "Analyzing CVE-2026-0866 (Zombie ZIP)," March 11, 2026. https://www.ironcastle.net/analyzing-zombie-zip-files-cve-2026-0866-wed-mar-11th/

[12] CERT/CC, "Vulnerability Note VU#968818 – Multiple antivirus products fail to scan archives," Carnegie Mellon University SEI, 2004. https://www.kb.cert.org/vuls/id/968818

[13] Kaspersky Securelist, "Head Mare and Twelve: Together Against a Common Enemy," 2025. https://securelist.com/head-mare-twelve-collaboration/