

Decompression and Archive Extraction Vulnerabilities: A Taxonomy of 21 Attack Mechanisms Across 1,067 Public Cases (2001–2026)

Eugen Spivak

publications@eugenspivak.com

Independent researcher <https://orcid.org/0009-0005-7509-8609>

Research Article

Keywords: zip bomb, decompression bomb, Zip Slip, path traversal, archive extraction, vulnerability taxonomy, vulnerability classification, CVE, GitHub Security Advisory, memory safety, software supply chain security, parser differential, CWE-409

Posted Date: October 18th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11369021/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License. [Read Full License](#)

Additional Declarations: The authors declare potential competing interests as follows: The author develops commercial data-compression software. No other competing interests.

Abstract

Decompression and archive-extraction vulnerabilities span resource amplification, memory-safety faults, parser inconsistencies and unintended file writes. Prior research examines these four failure classes separately, and no unified classification covering all four was found in the studies reviewed. This paper presents a 21-mechanism taxonomy and a reusable labelled inventory of 1,067 observed included cases from the National Vulnerability Database and the GitHub Advisory Database, published 2001 to September 2026.

In the 937 cases of the fully enumerated, query-selected frame, extraction path handling (path traversal, Zip Slip) accounts for 643 (69%), and application wrappers carry 340 of those 643 (53%). Only 77 of the 937 records (8%) describe a mitigation mechanism within the record. Of 195 amplification cases, 45 describe a size check that trusted a declared size or was enforced incompletely, and 72 arrive as network messages, 63 as files or packages. The 130 sampled decoder-only cases are estimated separately. The 2026 rise (45.3 per 10,000 CVEs, against 17.4 in 2025) persists under six sensitivity checks but describes the audited record, not incidence.

Two instructed passes agreed at kappa 0.81 to 0.99 (reliability). A blinded pass by the author (80 records, sample-specific) matched the adjudicated mechanism in 96% of random cases (kappa 0.94) and exposed a shared misreading of fix language, since corrected. No external review exists; findings describe the audited records, not deployed software. The inventory, taxonomy, CWE crosswalk, benchmark protocol and a proposed, untested review checklist are released to classify new records, evaluate advisory-text classifiers and repeat the study.

1. Introduction

Software unpacks attacker-supplied data all the time: uploads, package installs, HTTP responses, RPC messages, images and documents. When that goes wrong, the public record describes the failure one advisory at a time, in the words of whoever filed it. The same mechanism appears as "zip slip", "path traversal", "directory traversal" or "arbitrary file write". A decompression bomb may be filed under resource exhaustion, uncontrolled allocation or data amplification, or under no weakness class at all. A security team that triages these records reads them one by one.

This paper turns that record into a table that others can query, update and challenge. It defines a taxonomy of 21 attack mechanisms in four strands, codes 1,067 observed included cases against it, and releases the cases, the codebook, the scripts and a machine-readable taxonomy as a versioned data record. The unit is a public case, not a vulnerability in deployed software: every count below describes the audited records.

Key findings. Each result names its denominator and its population (Section 3.3 defines the populations).

- Extraction path handling accounts for 643 of the 937 included cases of the fully enumerated frame (69%).
- Application code that wraps an extraction library carries 340 of those 643 cases (53%).

- Only 77 of the 937 records (8%) explicitly describe a mitigation mechanism within the analyzed record.
- In 45 of the 195 amplification cases the record describes a size check that trusted a declared size or was enforced incompletely.
- 72 of the 195 amplification cases arrive as network messages, against 63 as files or packages.

Three limits apply to every one of them. The 937 cases come from a query-selected frame, not from all decompression vulnerabilities. They are coded from the text of the record alone, so the mitigation figure measures description completeness, not the existence of a fix. And the coding has not been externally validated (Section 3.5).

One more result needs care. In the first nine months of 2026 the National Vulnerability Database (NVD) published 330 records in the frame, 45.3 per 10,000 CVEs published, against 17.4 in 2025. That is a change in the audited public record: more vulnerabilities found or published, changed terminology, labelling or coverage could each produce it, and Section 6 tests what the frozen data can separate.

The record matters to more than one community, and the evidence for each is specific: package installers and extraction wrappers carry 438 of the 937 cases (software supply chain security); amplification that arrives as network messages links it to network-service security; the semantic-gap strand to parser and file-format security; the mitigation and weakness-label findings to vulnerability intelligence. Modern data-ingestion pipelines that unpack uploaded archives are one application context for the same mechanisms; the records analyzed here do not measure them.

1.1 Contributions and Intended Research Uses

This paper makes four contributions:

1. a taxonomy of 21 decompression and archive-extraction mechanisms in four strands, with operational definitions and stable identifiers (Section 4);
2. a versioned, coded inventory of 1,067 observed included public cases, with explicit sampling and classification boundaries (Section 3) and a data record that others can load without the author (Section 8);
3. an empirical analysis of mechanism distributions, affected component roles and mitigation descriptions in the audited records, with sensitivity checks of the time trend (Sections 5 and 6);
4. a proposed, untested mechanism–control–verification review checklist that links each observed failure class to defensive review questions (Section 7 and Appendix A).

The taxonomy and the inventory support three uses: classifying newly published CVEs or advisories against the 21 definitions; comparing an advisory-text classifier with the published labels (evaluating a static-analysis tool needs code-level evidence and independently validated labels, which this record does not provide); and repeating the study on a later frozen frame with the same queries and definitions, which turns the per-period rates into a series.

The record is not independently validated ground truth and not a vulnerability detector; the checklist is a proposed review checklist, not a validated standard; and no control was tested, so nothing here proves that a listed control prevents exploitation. The paper reproduces no case, constructs no input and tests no software. Products appear only in the inventory, exactly as the records name them; no table ranks or counts them.

How findings are labelled. Every finding is one of three kinds. *Observed* means a count in a named population of records. *Supported interpretation* means what those records suggest, with the alternatives named. *Not established* means a claim about prevalence, incidence, loss or effectiveness that the records cannot support. Section 5.7 sets them side by side.

2. Related Work

The closest studies each answer one part of the question. Tables 1 and 2 compare them with this paper on nine dimensions. They are set out first because they decide what this paper can claim.

Table 1

No closest study combines the four mechanism strands with a case-level classification of public vulnerability records (what each examines). Entries come from each work's abstract or publisher page, read on 11 October 2026; "not stated" means that the page read does not say.

work	source type	records or systems examined	taxonomy coverage	mechanism definitions
Pellegrino et al. 2015 [Pellegrino2015]	experiments	19 implementations of 3 network protocols	amplification in network services	12 pitfalls named
Snyk, Zip Slip research, 2018 [Snyk2018]	practitioner research and disclosure	"thousands of projects"; libraries listed in a linked repository	one strand: path handling	one class named and described
You et al. 2025 [You2025]	differential fuzzing	50 ZIP parsers in 19 languages	one strand: semantic gaps	14 ambiguity types
Nallamothu 2026 [Nallamothu2026]	public CVE records and detector experiments	11 CVEs from 8 projects and the CPython library	one strand: file writes from archive and download metadata	3 failure modes
Bhuiyan et al. 2025 [Bhuiyan2025]	literature and engine review	regex engines and the ReDoS literature (a sibling resource-exhaustion class)	not decompression	not decompression
Liu et al. 2026 [Liu2026]	experiments and library analysis	85 collected archive-extraction projects in ten languages	one strand: link following and path validation in extraction	5 defense categories, 5 bypass schemes, 3 write methods
Jana and Shmatikov 2012 [Jana2012]	experiments	36 antivirus scanners	one strand: parser differentials in file processing	2 attack families
Müller et al. 2021 [Mueller2021]	experiments	28 PDF processing applications	PDF features, with denial of service as one of four classes	4 attack classes
Wang et al. 2019 [Wang2019]	app measurement	over 5 million app items from 25 markets	Android signing issues	21 anti-patterns
Momot et al. 2016 [Momot2016]	conceptual taxonomy with CVE cases	not stated	input-handling errors in general	taxonomy of LangSec errors
this paper	public NVD and GitHub advisory records	1,960 candidate records, 1,067 observed included cases	four strands, 21 mechanisms	21 definitions with decision rules

Table 2

This paper adds a case-level, versioned record and a public classification, but tests no control and has no external validation (what each establishes).

work	empirical validation	control testing	public data set	reproducible classification	longitudinal scope
[Pellegrino2015]	own experiments	yes, against implementations	not stated	not applicable	no
[Snyk2018]	disclosure to maintainers	not stated	list in a linked repository	no formal classification	no
[You2025]	differential fuzzing; artifact badges for availability, functionality and reproduction	exploit scenarios and seven proposed mitigations	artifacts reviewed by the venue	not applicable	no
[Nallamothu2026]	detector results on 11 CVEs	yes, detection rules	labelled 11-CVE corpus and detector logs	rules and corpus released	no
[Bhuiyan2025]	literature review	engine review	not stated	not applicable	literature over time
[Liu2026]	16 arbitrary file write vulnerabilities found, 2 with high-severity CVEs	yes, against libraries and tools	datasets, test cases and configuration on Zenodo	not applicable	no
[Jana2012]	45 evasion exploits against 36 scanners	yes, against scanners	not stated	not applicable	no
[Mueller2021]	evaluation of 28 applications, 26 vulnerable	yes, against applications	not stated	not applicable	no
[Wang2019]	automated measurement	tool-based detection	not stated	not applicable	evolution analysis across markets
[Momot2016]	CVE cases as examples	no	not stated	no	no
this paper	two instructed coding passes; a blinded pass	none	frame, codes, labels, scripts, taxonomy	codebook, schema, example queries	2001 to 30 September 2026, per period, with

work	empirical validation	control testing	public data set	reproducible classification	longitudinal scope
	by the author; no external review				sensitivity checks

Among the closest studies identified in this review, none combines the four mechanism strands with a case-level, versioned classification of the NVD and GitHub Advisory Database records examined here. That statement rests on the works above and on searches of OpenAlex, arXiv and the web on 11 October 2026; it is not a claim that no other work exists. The narrower papers establish things this one cannot: only experiments show that an implementation fails, and only a tool evaluation shows that a detector finds a pattern.

Four threads stand behind the matrix.

- Amplification. Pellegrino et al. [Pellegrino2015] (Table 1) are joined by studies of overlapping ZIP entries [Fifield2019], decompression quines against antivirus engines [Canet2017], medical-imaging devices [Velinov2022], recursive archives [VanMello2024], non-recursive bombs [Chen2020] and compression-conversion amplification in content delivery networks [Lin2024]. Algorithmic-complexity attacks are the parent class [Crosby2003].
- Semantic gaps. Malware scanners and end hosts parse the same file differently [Jana2012], as do PDF parsers [Carmony2016] and ZIP parsers [You2025].
- Path handling. Snyk named the class in 2018 [Snyk2018]; Nallamothu systematized 11 CVEs [Nallamothu2026]; Liu et al. traced how misunderstood symlinks in standard libraries reach path validation in extraction, with 16 arbitrary file write vulnerabilities found [Liu2026].
- Parser faults in general. The language-theoretic security literature classifies input-handling errors [Momot2016] and parser-differential anti-patterns [Ali2023]; it is not specific to compression and does not count cases.

The sibling resource-exhaustion class, regular-expression denial of service, received a systematization of knowledge in 2025 [Bhuiyan2025]. Decompression has none that this review found, which is the gap this paper takes up: not a new attack, but a common vocabulary and a queryable record.

2.1 What the weakness classes of the records do not separate

The taxonomy is finer than the weakness classes that the source records carry. Table 5 (Section 4.1) maps the mechanisms to the CWE; the counts here come from the weakness classes the records list themselves. CWE-22 is listed by 517 of the 937 included cases of the frame, which belong to 5 mechanisms: 445 are P1 (a name that escapes the folder) and 68 are P2 (a link), so one weakness class covers two failures with different controls. CWE-409 is listed by cases of 6 mechanisms, from the single-stream bomb (A1, 83

cases) through a limit that misses a path (A5, 16) to a declared-size check (A4, 5), which the taxonomy keeps apart. In the other direction one mechanism is spread over many labels: A1 cases carry 18 different weakness classes and P1 cases 28. 44 of the 937 cases list no weakness class beyond the generic NVD-CWE-Other and NVD-CWE-noinfo, or none. The CWE remains the established vocabulary; the taxonomy adds the separation of mechanism, failing component and control. The counts for the eight most frequent classes are in the data record (cwe_comparison_v1.2.md).

3. Study Design, Data Sources and Methods

3.1 Sources and frame

The sources are the NVD CVE API 2.0 [NVD] and the reviewed advisories of the GitHub Advisory Database [GHSA], both read without authentication on 6 October 2026. The cutoff is publication on or before 30 September 2026, with no lower bound. The publication date is the NVD published field of a CVE record and the advisory's published date for an advisory-only case; the year inside a CVE identifier can differ from it (Section 6).

The frame was built by queries, not by browsing:

- 32 keyword queries and one weakness-class query on the NVD;
- one query on the GitHub Advisory Database, for reviewed advisories carrying one of ten weakness classes, kept only when the text names a compression or archive term.

Each query belongs to one stratum: amplification, semantic gaps, path handling or decoder faults. The full query list and every total are in the record. A record matched by an amplification, semantic-gap or path query was coded in full. A record matched only by the decoder queries ("decompress", "inflate" and their variants) entered a stratified sample, because those words also match many records of decoder faults in general. An advisory whose CVE was already in the frame became an alias of that case.

The frame holds 1,960 candidate records: 1,442 from the NVD and 518 advisory-only candidates, reviewed advisories whose record the NVD queries did not return. 440 of the 518 name a CVE as an alias; 78 name none. Of the 1,960, 1,370 form the fully coded frame and 590 are decoder-only candidates.

3.2 Inclusion

A case is included when the record describes a weakness reached by decoding, decompressing, unpacking or extracting compressed or archived input that an attacker can influence, in any software. Exclusions carry one of seven codes:

- X1: not compressed or archived input;
- X2: compression side channels;
- X3: supply-chain compromise of a decoder's source;

- X4: markup entity expansion without compression;
- X5: the producing side only;
- X6: a rejected or duplicate record;
- X7: too little information to code.

X2 to X4 are boundary classes. They are named here and left outside the counts.

3.3 Populations and sampling

One vocabulary is used in every table, figure and sentence. Figure 1 shows the flow, and Table 3 gives the counts.

Table 3

The study separates 1,960 candidate records into a fully coded frame and a sampled population (counts of the study-selection flow).

term	count	meaning
candidate records	1,960	every record the queries returned
fully coded frame	1,370	candidates matched by an amplification, semantic-gap or path query, all coded
decoder-only candidates	590	candidates matched only by decoder queries
stratified sample	200	200 of the 590, drawn across 57 strata of publication period by format word
excluded	433 + 70 = 503	433 in the fully coded frame, 70 in the sample (418 of the 503 as X1)
fully enumerated included cases	937	included cases of the fully coded frame: exact counts
sampled included cases	130	included cases observed in the sample
observed included cases	1,067	937 + 130: a count of coded cases, never a census, never a prevalence
weighted estimate	339 of 590	the 130 sampled cases weighted by stratum size over stratum sample: an estimate for the decoder-only candidates, with a design-based 95% interval of 303 to 374

The sample of 200 was drawn with proportional allocation, at least two cases per stratum and seed 20261006. The complete frame is published so that a reader can see what was not coded. Estimates for the decoder-only candidates carry a design-based 95% interval: the estimated total is the sum over the 57 strata of the stratum size times the sampled share, the variance is the stratified variance with a finite population correction in each stratum, and the interval is the estimate plus or minus 1.96 standard errors (a normal approximation, unreliable for small counts; where no sampled case falls in a category the upper bound is

the rule of three over the 200 sampled cases). The 937 cases of the fully enumerated frame are exact counts and carry no interval. Counts of the sample are never added to counts of the frame, and no prevalence is computed across the two; 1,067 is a count of coded cases, not a census.

3.4 Coding

Each case was coded from the record alone: its description, weakness classes, product identifiers and reference URLs. Coders opened no link and ran nothing. The fields are inclusion, primary and secondary mechanism (Section 4), format, carrier (file, package, network message, image or document), component role, stated impact, the mitigation class the record states, and a mechanism description of at most 40 words taken from the record.

Example paths and field values in that description are replaced by placeholders, and exploit-tagged references are not recorded. "Mitigation stated" means stated within the analyzed record (Section 5.5).

Two coding passes were made, each from the same codebook and its own task text; the second was blind to the first. The codebook, the coder instructions and the adjudication rules are published and are the procedure of record. Agreement is Cohen's kappa [Cohen1960] per field. Kappa was 0.963 for inclusion across 1,570 cases and 0.951 for the exclusion reason across the 488 cases both passes excluded. Across the 1,057 cases both passes included, it was 0.993 for strand, 0.975 for primary mechanism, 0.962 for format, 0.883 for component and 0.812 for mitigation class. Because the two passes shared one codebook and one coder design, these figures are reliability, not agreement between independent human judges, and not validation.

Every disagreement was adjudicated under written rules. Rules R1 to R14 were fixed before the two passes were compared, from the coders' notes on hard cases. Rules R15 to R18 (component order, format naming, stated mitigations, audio and video codecs) were fixed during adjudication, before any count was made. The log of 398 adjudicated values names the rule for each. After adjudication, the 45 records coded A4 or A5 were re-read against the codebook; all 45 codes were confirmed, and the reading of each record is published.

The mitigation class was corrected after the blinded pass of Section 3.5 (Table 9 gives the corrected counts); the kappa of 0.812 above was measured before that correction.

3.5 Validation of the coding

The classification has had no external review. Two checks sit between that fact and the findings.

The two instructed passes measure reliability only, as Section 3.4 says.

A blinded coding pass by the author is the second rung. I coded 80 records from the record text alone, without the labels, in a random order, with the codebook of the two instructed passes. Fifty are included cases of the fully enumerated frame drawn at random by strand (32 path handling, 10 amplification, 5 decoder faults, 3 semantic gaps), which measures agreement with the adjudicated labels, not externally established accuracy. Thirty are challenge records drawn from the cases where the two passes disagreed

(10 each on inclusion, strand or mechanism; component; mitigation class). The seed is 20261011. The agreement figures are specific to this sample: the random part is stratified by strand and not weighted to the population, and the challenge part is deliberately difficult. The key was held by the scorer from the start; I did not see it. I wrote the codebook, so the pass is independent of the two instructed passes but not of the design, and it is not an external review. Its protocol, answers and results are in the record.

Measured before any correction, agreement with the adjudicated labels in the random sample (49 of 50 cases coded as included by both) was 0.98 for strand (Cohen's kappa 0.96), 0.96 for primary mechanism (kappa 0.94), 0.84 for component (kappa 0.80) and 0.92 for mitigation class (kappa 0.81). In the challenge sample (30 records) it was 0.96 for strand (kappa 0.93), 0.88 for mechanism (kappa 0.85), 0.92 for component (kappa 0.90) and 0.96 for mitigation class (kappa 0.92), and 28 of 30 inclusion decisions agreed. The 27 field disagreements fall on 23 records. Re-read under the written adjudication rules, four are corrections of the inventory, because a rule shows the adjudicated label to be wrong; the other 23 are boundary cases that no rule settles, and those labels stand.

The corrections exposed a shared misreading. Three were mitigation classes in the random sample, where both instructed passes had coded as nothing stated a fix that the record states (a Recommended Fix section, a released patch, a patched version in the last sentence). I then re-read the mitigation field of every included case of the fully enumerated frame coded none-stated (631) or upgrade-only (236), under rules R11 and R17, verifying each change against the record text. 55 mitigation classes changed: 48 none-stated to upgrade-only, 6 none-stated to a named mechanism and 1 upgrade-only to a named mechanism. The corrected counts are 77 mechanism stated, 283 upgrade only and 577 nothing stated, against 70, 236 and 631 before (Table 9). The agreement figures were measured before this correction and are not re-scored, because re-scoring against corrected labels would be circular. The mitigation field of the 130 sampled cases was not re-read; it enters no table. The correction rests on the same rules and the same author, so another shared misreading may remain.

An external blinded review of 50 to 100 records, a comparison of advisory text with linked patches and a test of the defensive controls are defined as a validation roadmap in Section 9. None of them has been done.

3.6 Guidance records

The records rarely say what the fix does (Section 5.5), so the guidance that readers are pointed to was also reviewed. The selection rule was fixed before any document was read:

- the weakness-class entries for the strands [CWE409; CWE22; CWE59];
- the security-considerations sections of the format and transport specifications in the inventory [RFC1950; RFC1951; RFC1952; RFC7932; RFC8878; RFC7692; RFC9110], with the ZIP specification [APPNOTE];
- the standard-library archive and compression documentation of the four ecosystems most frequent among included advisories: Go (59 cases), Python (39), Java (35) and Node.js (30) [GoZip; GoTar; PyZipfile; PyTarfile; NodeZlib; JavaZip].

All were retrieved on 6 October 2026. Each is quoted verbatim in at most 40 words, or recorded as an absence. Every quote was checked against the retrieved text.

3.7 Rates

Counts by year mix two things: how often a weakness occurs and how many records are published at all. The NVD published 48,152 CVEs in 2025 and 72,811 from January to September 2026. I therefore report NVD-sourced cases per 10,000 CVEs published in each period. Advisory-only cases are left out of the rates because they have no matching denominator; Section 6 shows what changes when they are added to the counts.

3.8 Use of AI tools

AI assistants supported evidence collection and processing, as well as manuscript preparation. I made the research and publication decisions and am responsible for the content.

4. A Taxonomy of Decompression and Archive Extraction Failures

Each strand names a different way that one decoding step goes wrong:

- Amplification: the work grows beyond what the input should cost.
- Decoder and parser faults: the decoder itself misbehaves on malformed input.
- Semantic gaps: two components read the same input differently.
- Extraction path handling: the output lands where it should not.

The 21 mechanisms and their counts are in Table 4. The machine-readable taxonomy in the data record gives every mechanism a stable identifier, a definition, a use rule and a boundary note; identifiers never change meaning. Three rules matter most for reading the counts:

- an integer error is primary over the out-of-bounds access it causes (D3 over D1 or D2);
- a large allocation sized by a header field is D6, not amplification, unless real output is produced;
- a semantic gap is coded only when the record says that two components disagree.

Table 4

Path traversal by name is the largest mechanism in the fully enumerated frame (510 of 937), and semantic gaps are almost absent (13). "Fully enumerated frame" counts are exact counts of the 937 included cases of the query-selected frame. "Decoder-only estimate" figures are weighted estimates for the 590 decoder-only candidates, from the 130 sampled cases in scope, with design-based 95% intervals (Section 3.3); they are not observed cases and are never added to the first column. Each estimate is rounded separately, so the rounded decoder-fault rows sum to 239 while the unrounded strand total of 240.7 rounds to 241; the unrounded values are in the data record.

code	mechanism	fully enumerated frame	decoder-only estimate (weighted)	95% interval of the estimate	first public case
A	Amplification	195	51	30 to 71	
A1	unbounded single-stream output	144	38	20 to 56	CVE-2012-3443 (2012)
A2	nesting and recursion	4	0	0 (none observed; up to 9)	CVE-2016-7166 (2016)
A3	shared or overlapping data	2	0	0 (none observed; up to 9)	CVE-2019-13232 (2019)
A4	declared-size trust	6	5	0 to 13	CVE-2019-16892 (2019)
A5	limit not applied on a path	39	8	0 to 16	CVE-2010-0734 (2010)
D	Decoder and parser faults	86	241	206 to 275	
D1	out-of-bounds write	23	102	76 to 129	CVE-2002-1308 (2002)
D2	out-of-bounds read	8	41	23 to 59	CVE-2014-8710 (2014)
D3	integer error	9	15	6 to 24	CVE-2007-1071 (2007)
D4	lifetime or state error	9	38	18 to 59	CVE-2008-0888 (2008)
D5	crash without memory corruption	10	6	0 to 14	CVE-2005-2459 (2005)
D6	allocation from a declared value	18	7	0 to 15	CVE-2019-13960 (2019)

code	mechanism	fully enumerated frame	decoder-only estimate (weighted)	95% interval of the estimate	first public case
D7	non-termination or excessive work	9	30	16 to 44	CVE-2007-1281 (2007)
S	Semantic gaps	13	1	0 to 3	
S1	entry ambiguity	8	0	0 (none observed; up to 9)	CVE-2007-4546 (2007)
S2	boundary ambiguity	3	0	0 (none observed; up to 9)	CVE-2015-6783 (2015)
S3	name ambiguity	0	0	0 (none observed; up to 9)	-
S4	type confusion	0	0	0 (none observed; up to 9)	-
S5	feature-support gap	2	1	0 to 3	CVE-2005-3499 (2005)
P	Extraction path handling	643	46	29 to 63	
P1	path traversal by name	510	33	16 to 51	CVE-2001-1267 (2001)
P2	link following	127	10	5 to 15	CVE-2005-1111 (2005)
P3	overwrite, permission or special file	2	3	1 to 4	CVE-2002-0759 (2002)
P4	name collision	4	0	0 (none observed; up to 9)	CVE-2021-37712 (2021)
total		937	339	303 to 374	

4.1 A crosswalk to the Common Weakness Enumeration

The taxonomy is not a replacement for CWE. Table 5 shows where the mechanisms meet it. A relationship is exact when the two definitions describe the same weakness, partial when they overlap, and contextual when the CWE names a parent, a neighbour or a consequence. Only D1 and D2 are exact. The tagged counts

show how the source records themselves labelled their cases; they are evidence of labelling practice, not of a correct mapping. The complete table is in the data record.

Table 5

Most mechanisms overlap a CWE only partially, and the only exact matches are the two out-of-bounds classes (closest CWE per mechanism group). CWE definitions read on 11 October 2026; the boundary conditions and the tagged counts of each of the 45 rows are in the data record (crosswalk_cwe_v1.1.csv).

mechanism	closest CWE	relationship
A1 unbounded output	CWE-409 (also CWE-400, CWE-770)	partial
A2–A3 nesting, overlap	CWE-409	partial
A4 declared-size trust	CWE-409	contextual
A5 limit not applied on a path	CWE-770	contextual
D1 out-of-bounds write	CWE-787 (parent CWE-119)	exact
D2 out-of-bounds read	CWE-125 (parent CWE-119)	exact
D3 integer error	CWE-190	partial
D4 lifetime or state error	CWE-416 (CWE-415, CWE-908)	partial
D5 crash without corruption	CWE-476 (CWE-617, CWE-369)	partial
D6 allocation from a declared value	CWE-789	partial
D7 non-termination	CWE-835 (CWE-674)	partial
S1–S5 semantic gaps	CWE-436 (CWE-351 for S4)	partial
P1 path traversal by name	CWE-22 (CWE-23, CWE-36)	partial
P2 link following	CWE-59 (CWE-61, CWE-62)	partial
P3 overwrite, permission, special file	CWE-732	partial
P4 name collision	CWE-178	partial

4.2 Three worked examples of difficult boundaries

Each example shows the record's words, the two candidate codes and the written rule that decides. They come from the documented disagreements and from records published after the frozen window.

Example 1

a fix stated late in the record (the shared misreading). GHSA-rgwj-5xj2-c3m3 describes a MySQL client library that inflates a compressed protocol packet with no output limit. Its last sections include a recommended fix whose code passes a limit to the inflate call: "zlib.inflate(body, { maxOutputLength: MAX_INFLATED_PACKET_SIZE })".

- *Candidate codes (mitigation class)*: none-stated, which both instructed passes and the adjudication gave, or output-size-cap.
- *Rule*: R17, a fix mechanism stated in the record sets the class. The record states one.
- *Outcome*: output-size-cap. The author's blinded pass found it, and the re-read of the mitigation field (Section 3.5) changed 55 values of this kind.

Example 2

client and server in one record. CVE-2026-106585, published after the frozen window, says there is "no check for whether the maximum packet length is exceeded during decompression of highly compressed data" in both the ssh client and the sshd server of OpenSSH.

- *Candidate codes (component)*: network-server-or-proxy, network-client or other.
- *Rule*: R15(c) assigns a protocol message to the side that decodes it; the record names both sides.
- *Outcome*: the rule does not decide. Two readers who followed it coded the component two ways and agreed on the mechanism (A5, a limit that exists but is not applied on the decompression path). A proposed refinement for the next version of the codebook is in Section 9.3.

Example 3

an allocation that is not amplification. CVE-2026-106453, also after the window, says a Java LZ4 library "uses getDecompressedLength to trust the four-byte decompressed-length header before validating the compressed input", so a five-byte input requests a very large buffer.

- *Candidate codes (mechanism)*: A4, declared-size trust, or D6, allocation from a declared value.
- *Rule*: A4 needs a limit that is checked against a declared size. Here no limit is checked and no decoded output is produced: one allocation is sized by a header field, which is D6 (the second precedence rule of Section 4).
- *Outcome*: D6; both readers agreed.

5. Results by Population

Three questions need a whole-record view: which strands dominate and how reporting practice shapes that picture (Sections 5.1, 5.2 and 6); where mitigations are stated and where guidance is silent (Sections 5.5 and 5.6); and which components carry the path-handling cases (Section 5.4). Every table names its population; counts of the fully enumerated frame (N = 937) are never pooled with counts of the sample.

5.1 The shape of the record

Of the 1,570 coded cases, 1,067 are included: 937 from the fully enumerated frame and 130 from the sample. The other 503 are excluded, 418 of them because the matched word did not concern compressed or archived input (Section 3.2).

Path handling dominates the fully enumerated frame: 643 of 937 cases (69%). That is a share of the query-selected frame, not an estimate for all decompression vulnerabilities. P1 alone, a name that escapes the target folder, accounts for 510. Amplification follows with 195 cases (21%), nearly three quarters of them A1 (144). The decoder-only population has a different shape. From the 130 sampled cases in scope, an estimated 339 of its 590 candidates would be in scope (95% interval 303 to 374; an estimate, not a set of observed cases), and most of them would be classic memory-safety faults: about 102 out-of-bounds writes (76 to 129), 41 out-of-bounds reads (23 to 59) and 38 lifetime errors (18 to 59). Semantic gaps are almost absent from the public record: 13 cases in the fully enumerated frame, and none at all for name ambiguity (S3) or type confusion (S4).

The format is not named in 243 of the 937 records (26%). Where it is named, ZIP leads in path handling (268 of 643) and in semantic gaps (11 of 13), and tar follows in path handling (165). Gzip and zlib or raw deflate together account for 51 of the 195 amplification cases (26%).

In 45 of the 195 amplification cases a size check existed but did not bound what was decoded: in 6 (A4) it trusted a size the input declared, in 39 (A5) enforcement was incomplete (5 counted compressed or wire bytes only, 3 applied after full decoding, 31 missed a codec, an API or a code path). The re-reading of all 45 records confirmed their codes.

5.2 Over time

Table 6

NVD-sourced cases of the fully enumerated frame rose to 45.3 per 10,000 CVEs in January to September 2026, after 9.6 to 18.9 in every earlier period (N = 778 cases; unit = included case; source = NVD CVE API). Cells give cases, then the rate per 10,000 in parentheses.

period	CVEs published	cases	per 10,000	A	D	S	P
2001–2009	37,230	49	13.2	0 (0.0)	7 (1.9)	1 (0.3)	41 (11.0)
2010–2014	27,192	26	9.6	3 (1.1)	11 (4.0)	1 (0.4)	11 (4.0)
2015–2019	61,400	116	18.9	10 (1.6)	17 (2.8)	1 (0.2)	88 (14.3)
2020–2022	63,544	92	14.5	9 (1.4)	3 (0.5)	1 (0.2)	79 (12.4)
2023–2024	68,769	81	11.8	14 (2.0)	3 (0.4)	0 (0.0)	64 (9.3)
2025	48,152	84	17.4	21 (4.4)	3 (0.6)	4 (0.8)	56 (11.6)
2026 (Jan-Sep)	72,811	330	45.3	102 (14.0)	28 (3.8)	5 (0.7)	195 (26.8)

From 2001 to 2025 the rate stayed between 9.6 and 18.9 cases per 10,000 CVEs. It was 45.3 in January to September 2026, with amplification at 14.0. This is a change in the audited public record. It can come from more vulnerabilities being discovered, more being published, changed terminology, changed weakness classification or changed reporting coverage, and the frame cannot fully separate them (Section 6). The figure covers nine months only and is not a measure of incidence.

5.3 Where bombs arrive

Table 7

Network messages carry more amplification cases (72 of 195) than files or packages (63), while path handling arrives almost only as files or packages (631 of 643) (N = 937; unit = carrier of the input).

strand	file or package	network message	image or document	other or unstated	total
Amplification	63	72	14	46	195
Decoder and parser faults	51	6	14	15	86
Semantic gaps	13	0	0	0	13
Extraction path handling	631	1	8	3	643

Of 195 amplification cases, 72 (37%) arrive as HTTP bodies, WebSocket messages or RPC messages, more than the 63 that arrive as files or packages; network servers, proxies and clients together carry 79. Path

handling is the opposite: 631 of its 643 cases are files or packages.

5.4 Who carries the defect

Table 8

Extraction wrappers carry 375 of 937 cases (40%), more than twice the 172 in decompression and archive libraries (N = 937; unit = component role in the record).

component	A	D	S	P	total	share
extraction-wrapper	33	2	0	340	375	40%
library-decoder	24	22	3	123	172	18%
archiver-tool	1	14	3	77	95	10%
package-installer	3	1	4	55	63	7%
network-server-or-proxy	49	10	0	4	63	7%
network-client	30	1	0	2	33	4%
media-or-document-pipeline	22	19	0	16	57	6%
security-scanner-or-gateway	12	8	1	18	39	4%
other	21	9	2	8	40	4%

An "extraction wrapper" is application code that calls a library to unpack an archive a user supplied. Wrappers carry 375 of 937 cases (40%), more than twice the 172 in decompression and archive libraries (18%). Package installers add 63, which together with wrappers makes 438 of 937 cases in code that unpacks archives on behalf of an application or a build.

Application code carries the path-handling cases. Extraction wrappers carry 340 of the 643 (53%), libraries 123 (19%), archiver tools 77 and package installers 55. The pattern is consistent with the guidance: two of the standard-library documents leave the path check to the caller or behind an opt-in switch (Section 5.6). An application that unpacks user archives with a library that leaves the check to the caller must write its own, and the records show how often such checks fail: path-handling records state code execution as the impact in 156 of 643 cases (24%). Amplification records state denial of service in nearly all cases.

Experiments on the tool side point the same way: Liu et al. found 16 arbitrary file write vulnerabilities through symlink-based bypasses of path validation in 85 extraction projects [Liu2026]. The 127 P2 cases of the frame are consistent with link handling being a recurring weak point; this is an interpretation, not a count of records.

5.5 What the record says about the fix

This section measures mitigation description completeness: what a record states about its fix, within the text analyzed. Linked pages, patches and vendor notes were not opened; a record that states nothing may still be fixed.

Table 9

Only 77 of 937 included records in the fully enumerated frame (8%) explicitly describe a mitigation mechanism within the analyzed record (unit = record; mitigation as the record states it).

strand	mechanism stated	upgrade only	nothing stated	total	mechanism stated (share)
Amplification	29	80	86	195	15%
Decoder and parser faults	7	19	60	86	8%
Semantic gaps	4	3	6	13	31%
Extraction path handling	37	181	425	643	6%
all	77	283	577	937	8%

Most records say that a fixed version exists, or say nothing about the fix: 77 describe a mechanism (8%), 283 state an upgrade or a fixed version only and 577 state nothing. The most frequent stated mechanisms are a canonical path check (31) and an output-size cap (21). Path handling, the largest strand, states a mechanism least often (37 of 643, 6%).

5.6 What the guidance says

Seven of the 15 mechanism groups have no guidance record in the set read: A2, A3, A5, D4 and D5, D7, S2 to S5, and P4 (the map of groups to records is in the data record, `guidance_map_v1.2.md`). The 21 guidance records (identifiers G01 to G21, with the gist and the verbatim quote of each, retrieved 6 October 2026) are in `guidance_quotes_v1.2.md`; the identifiers below refer to them.

Three things stand out:

- Among the 21 guidance records reviewed, the deflate, zlib and gzip specifications speak only of integrity, while the newer brotli and Zstandard specifications name amplification and bounds checking. The comparison is limited to these records and to the protections examined.
- The weakness class for decompression bombs, CWE-409, has no mitigation section of its own, and its observed examples are XML entity expansion, a class this paper excludes.
- Where standard libraries offer path protection, it is opt-in (Go), left to the caller (the experimental ZIP archive API of Node.js), or partial by the library's own account (Python's tarfile filters; Go's tar reader validates names "not link targets").

Where the two gaps overlap. The records describe a fix mechanism in only 8% of cases (Table 9), and the guidance covers eight of the 15 mechanism groups. The two gaps overlap where the risk is growing:

- A5, a limit that does not cover every path. This is the largest amplification mechanism after A1 (39 cases), and 32 of them were published in 2026. No guidance document in the set warns that a limit must cover

streaming, asynchronous or alternative codec paths. One documents a limit that covers only some methods (G18).

- A2 and A3, nesting and overlapping entries. These are the constructions research describes best [Fifield2019; Canet2017], yet they have no guidance at all. They are also rare in the record (4 and 2 cases).
- P2 and P4, links and name collisions. There are 127 and 4 cases. The only library-level statement on links is that they are not validated (G17).
- Semantic gaps. Only one document takes a position on which of two disagreeing ZIP records is authoritative (G19). The ZIP specification constrains what writers store, not what readers accept (G13).

Where guidance exists, it is mostly about A1 and P1, the two mechanisms with the most cases.

5.7 What the findings establish

Table 10

Each headline finding is an observed count in a named population; the interpretation and the limits are stated beside it.

finding	observed	supported interpretation	not established
path handling dominates	643 of 937 included cases of the frame (69%)	where extracted files land is the most reported failure in these records	its share among all decompression vulnerabilities
wrappers carry path handling	340 of 643 path-handling cases (53%)	application code carries more reported defects than libraries (123 of 643)	how often wrappers fail in deployed software
mitigation completeness	77 of 937 records describe a mechanism (8%); 283 an upgrade only; 577 nothing	records seldom say what a fix does	that fixes are undocumented elsewhere, or absent
limits that do not bound decoding	45 of 195 amplification cases (A4 6, A5 39)	a limit that exists is often not applied on every path or trusts a declared size	how common such limits are in deployed libraries
network carriers	72 of 195 amplification cases arrive as network messages, 63 as files or packages	amplification is a network-service problem as much as a file problem	the relative frequency of the two carriers in practice
the 2026 rise	330 NVD-sourced cases, 45.3 per 10,000 CVEs, against 17.4 in 2025	the audited record changed; the rise persists under every check of Section 6	any change in the number or incidence of vulnerabilities
sparse semantic gaps	13 cases; none for S3 and S4	the public record undercounts this strand [You2025]	the true frequency of semantic gaps

6. Robustness and Reporting Sensitivity

The 2026 count depends on which words records use, which weakness classes they carry, which source publishes them and how a date is defined. This section runs the checks the frozen data allow; every table is generated by a script of the record (sensitivity.py) from the published inventory. A check the data cannot support is marked as such.

The NVD changed its operations in April 2026: from 15 April it gives immediate enrichment only to CVEs in the known-exploited catalog, CVEs for software used in the federal government and CVEs for critical software, marks the rest "Lowest Priority", and moves backlogged records published before 1 March 2026 to a "Not Scheduled" category; all submitted CVEs still appear in the NVD [NIST2026]. This concerns enrichment, which includes weakness-class and product data that the frame partly uses; the record does not store whether a weakness tag came from the numbering authority or from the NVD, so nothing separates its effect.

6.1 Reporting practice and mechanism shifts

Table 11

Classification practice changed faster than the vocabulary: amplification records tagged CWE-409 rose from 7 of 28 (2020–2024) to 85 of 130 (2026) (all sources, fully enumerated frame; unit = case).

period	A cases	A tagged CWE-409	A with the word "bomb"	A arriving as a network message	A with a declared-size check or incomplete limit enforcement (A4, A5)	P cases	P naming "Zip Slip"	P through links (P2)
2001–2014	3	0	0	2	1	52	0	7
2015–2019	10	1	7	4	1	90	14	18
2020–2024	28	7	15	8	3	194	45	26
2025	24	12	12	7	3	68	18	11
2026 (Jan-Sep)	130	85	57	51	37	239	37	65

Three readings follow from the table:

- Classification practice changed. Amplification records tagged CWE-409 went from 7 of 28 (2020–2024) to 85 of 130 (2026); the word "bomb" did not rise in the same way (15 of 28, then 57 of 130). A frame built from words and classes finds more cases when records name the class more often, so part of the 2026 rise belongs to the record.
- The mechanism mix shifted, not only the volume. In 2026, 37 of the 130 amplification cases (28%) involved a declared-size check or incomplete limit enforcement, against 3 of 28 (11%) in 2020–2024, and 51 of 130

arrived as network messages. These shares are less exposed to a vocabulary change than the volume is, but not immune to it.

- The public record under-reports semantic gaps. It holds 13 such cases, while one study found 14 ambiguity types across 50 ZIP parsers and reported three CVEs [You2025]; many gaps surface outside CVEs, so the count is a floor.

The rise in links is real in the same sense. P2 cases went from 7 of 52 path cases (2001–2014) to 65 of 239 (2026).

How a change in weakness labelling or enrichment would move the frame. Two parts of the frame depend on weakness classes. On the advisory side the filter selects reviewed advisories that carry one of ten weakness classes: all 82 advisory-only cases of 2026 entered through it, so a change in how advisories are labelled adds or removes such cases directly. On the NVD side the cases are found by words, and the one weakness-class query (CWE-409) returned 4 records; but 283 of the 330 NVD-sourced cases of 2026 (86%) carry one of the ten classes, against 62 of 84 (74%) in 2025, so a change in labelling or enrichment changes which records the coding, Table 11 and the shares of Tables 12 and 13 see, without any change in the software. The frame cannot tell the two apart; it can show only that the keyword-only records, which do not depend on those labels, also rose, by a smaller factor (1.4 times; Check 2).

6.2 Sensitivity checks of the 2026 rise

Six checks and one descriptive count follow, each changing one thing.

Table 12
The 2026 rate is between 2.2 and 3.2 times the 2025 rate whichever query stratum is left out (NVD-sourced cases per 10,000 CVEs; unit = included case of the fully enumerated frame).

left out	2025	2026 (Jan-Sep)	ratio 2026 / 2025
nothing	17.4	45.3	2.6
path only	6.2	20.1	3.2
amplification	12.2	27.5	2.2
semantic	16.6	44.4	2.7
decoder with path	17.2	44.1	2.6

Check 1, by query stratum. Table 12 leaves out each group of the frame's queries in turn. The ratio stays between 2.2 and 3.2, so the rise is not the product of one query family. Amplification-matched cases rose most (5.2 to 17.9 per 10,000 CVEs), path-only cases from 11.2 to 25.3.

Table 13

The rise is concentrated in records that carry a listed weakness class (3.0 times the 2025 rate), not in keyword-only records (1.4 times) (NVD-sourced cases per 10,000 CVEs; unit = included case).

period	with a listed weakness class	keyword-only	share keyword-only	without cases that depend on recent vocabulary
2001–2009	10 (2.7)	39 (10.5)	80%	49 (13.2)
2010–2014	12 (4.4)	14 (5.1)	54%	26 (9.6)
2015–2019	87 (14.2)	29 (4.7)	25%	115 (18.7)
2020–2022	85 (13.4)	7 (1.1)	8%	92 (14.5)
2023–2024	75 (10.9)	6 (0.9)	7%	81 (11.8)
2025	62 (12.9)	22 (4.6)	26%	83 (17.2)
2026 (Jan-Sep)	283 (38.9)	47 (6.5)	14%	313 (43.0)

Check 2, keyword-matched against weakness-class-matched. Table 13 separates records that carry one of the ten weakness classes of the advisory filter (CWE-409, 22, 59, 400, 405, 770, 776, 789, 674 and 835) from records found by keyword alone. The first group rose 3.0 times, the second 1.4 times. Part of the rise is therefore labelling (Section 6.1).

Check 3, without recent vocabulary. A keyword query counts as recent vocabulary when at least 75% of the candidate records it matched were published in 2025 or 2026 (all candidate records: 58%); 10 of the 33 keyword queries qualify. A case depends on it when every query that matched it is such a query. Leaving out the 17 cases of 2026, one of 2025 and one of 2015 to 2019 that do (last column of Table 13) leaves a ratio of 2.5 against 2.6: terminology that became common in 2025 and 2026 does not explain the rise alone.

Check 4, NVD-only against combined sources. Advisory-only cases have no denominator, so no rate is given for them: they are 82 of the 412 included cases of 2026 against 16 of 100 in 2025. The ratio of 2026 to 2025 is 2.6 with NVD-sourced cases alone and 2.7 with all sources divided by the same NVD totals (a ratio of unlike things that shows only that the shape holds).

Check 5, publication year against identifier year. 90 of the 778 NVD-sourced cases carry a CVE identifier from a different year than their publication date. Without denominators by identifier year only counts can be compared (the table of sources and dates is in the data record, sensitivity_v1.1.md): 318 by identifier year against 330 by publication date, so the date definition moves the 2026 total by 12 cases.

Check 6, a matched window. The full year 2025 sets nine months of 2026 against twelve. In January to September 2025 the NVD published 35,366 CVEs and 58 NVD-sourced cases fall in the window (16.4 per 10,000); in January to September 2026 it published 72,811 and 330 cases fall in it (45.3 per 10,000), a ratio

of 2.8 against 2.6 for the full year 2025. Advisory-only cases number 9 against 82. A later read gives 72,810 CVEs for the 2026 window, one record rejected since; the tables keep the count of 6 October. The matched window does not weaken the rise.

Check 7, months and products (descriptive). NVD-sourced cases per month in 2026 were 19, 26, 30, 32, 26, 43, 47, 50 and 57 from January to September: the counts climb, a single step at one month such as April is not visible, and counts cannot separate causes (no monthly NVD totals are in the record, so no monthly rates). One product does not explain the rise: the most frequent product in 2026 has 8 of 330 cases (2%), against 3 of 84 (4%) in 2025.

What the checks do not do. The 2026 count survives changes to the frame, the matching rule, the source mix and the date definition. The checks cannot tell whether more vulnerabilities were found, published or labelled; that needs a frame that does not depend on record wording, for example one built from fix commits (Section 9). The conclusion stays a change in the audited public record, not a measured change in incidence.

7. Implications for Security Engineering

The ten recommendations below turn the measured findings into actions. Each names who should act, the existing guidance that maps to the mechanism, the first step and how to check it. Each rests on its own evidence in the tables of Section 5; no two share one. None proposes a new design, and none was tested for effect (Section 9). They show where the record points, not what a fix achieves, and the paper offers no estimate of prevented vulnerabilities, savings or effectiveness. Appendix A turns the taxonomy into a proposed, untested mechanism–control–verification review checklist that the recommendations draw on.

Each recommendation carries an evidence status. Documented behavior means a library's or a standard's own documentation states the relevant behavior, with source, version and retrieval date. Inferred means a control follows from a specification, a weakness entry or the records, and no document describes an implementation of it. The five strongest actions come first, in the order that the evidence supports.

Table 14

Five actions come first: consistent decoding limits, testable extraction confinement, documented extraction defaults, mitigation mechanisms in advisories and weakness guidance (the ten recommendations at a glance).

#	audience	mechanism	evidence	recommended action	verification	evidence status
1	library maintainers	A4, A5	45 of 195 amplification cases (Table 4)	make each resource limit cover every decoding entry point	the library's tests exercise each limit on each entry point	gap documented (G18); control inferred
2	application integrators	P1–P4	340 of 643 path-handling cases in wrappers (Table 8)	document and test extraction confinement per site	no entry is written outside the destination root, under the actual library	library behaviors documented (G15–G17, G20); control inferred
3	library maintainers	P1–P4	123 of 643 path-handling cases in libraries; 127 link cases	publish what the default extraction checks, mechanism by mechanism	the table is published and each "enabled by default" row has a test	model documented (G15); control inferred
4	advisory writers	all	77 of 937 records describe a mechanism (Table 9)	state what the fix does in one sentence	new records name the mechanism, not only the version	observed in the records; action inferred
5	weakness-class maintainers	A1–A5	85 of 130 amplification records of 2026 tagged CWE-409 (Table 11)	give CWE-409 its own mitigations and compression examples	a later CWE version lists them	entry content documented (G01); action inferred
6	network-service teams	A5	72 of 195 amplification cases arrive as network messages (Table 7)	limit decoded content where network input is decoded	every handler applies its limits during decoding	inferred from guidance (G08, G09)
7	format maintainers	A1	51 of 195 amplification cases in gzip, zlib or raw deflate	name amplification and decoder limits in the security considerations	the next text names them	specification text documented (G04–G09); action inferred

#	audience	mechanism	evidence	recommended action	verification	evidence status
8	format maintainers	P1	268 of 643 path-handling cases in ZIP	give readers the rule that writers already have	a revision states a reader-side rule	specification text documented (G13); action inferred
9	library maintainers	S1–S2	13 semantic-gap cases; 14 ambiguity types in research [You2025]	publish an advisory when an ambiguity fix has a substantiated security impact	security-relevant fixes carry an identifier	inferred
10	advisory writers	all	243 of 937 records do not name the format	name the format and how the input arrives	new records name both	observed in the records; action inferred

7.1 The ten recommendations

The check of each recommendation is the verification column of Table 14.

Library maintainers

1. Make each resource limit cover every decoding entry point, and say which limit it is.

- *Basis*: in 45 of the 195 amplification cases a size check existed but did not bound what was decoded. Six trusted a size the input declared (A4). In 39 the enforcement was incomplete (A5): 5 counted compressed or wire bytes only, 3 were applied after full decoding, and 31 missed a codec, an API or a code path. 37 of the 45 were published in 2026 (Table 4; the re-reading of the 45 records is in the data record).
- *With*: G09; G18 (a limit that covers convenience methods only); G10 (a declared size smaller than the real output).
- *First step*: list every public entry point that decodes (synchronous, asynchronous, streaming, each codec). Mark for each which limits apply: input size, decoded output, memory, nesting depth and processing time. Note whether each limit counts the bytes actually produced or a size the input declares.

Application integrators

2. Make archive extraction confinement documented and tested.

- *Basis*: application code that wraps a library carries 340 of the 643 path-handling cases (53%). Those cases state code execution as the impact in 156 of 643 (Section 5.4).

- *With*: the library safeguards where they cover the mechanism: Python's tarfile filters (G15), Go's opt-in insecure-path errors (G16, G17); CWE-22 (G02); a library documentation that leaves the check to the caller (G20).
- *First step*: list every place that extracts an archive received from a user. For each, record which of these the library handles and which the application must handle:
 - a controlled destination root, with file operations relative to that root where the platform offers them;
 - absolute and parent-relative names;
 - a policy for symbolic links, hard links and special files;
 - name collisions under the target file system's rules;
 - overwriting existing files;
 - time-of-check to time-of-use races between validation and write;
 - restoring permissions and ownership.
- *Check*: string canonicalization alone does not cover links, collisions or races, so tests run against the library's actual behavior, not its documentation alone, show that no entry is written outside the destination root and that rejected entries are handled as the documented policy says.

Library maintainers

3. Publish what the default extraction checks, mechanism by mechanism.

- *Basis*: libraries carry 123 of the 643 path-handling cases, and links account for 127 (P2). One standard library states that "Only file names are validated, not link targets" (G17). Another checks names only behind an opt-in switch (G16) (Table 8).
- *With*: the model of G15, which documents its filters and says plainly: "None of the available filters blocks all dangerous archive features."
- *First step*: add a table to the extraction documentation with one row per mechanism: names (P1), links (P2), permissions and special files (P3), name collisions (P4). Mark each row "enabled by default", "optional", "partial" or "caller's responsibility".

Advisory writers and weakness-class maintainers

4. State what the fix does, in one sentence.

- *Basis*: within the analyzed records, 577 of 937 state nothing about the fix and 283 state only a fixed version. 77 (8%) name a mechanism (Table 9). This is a finding about the records, not proof that the fixes are undocumented elsewhere.
- *With*: the mechanisms the records themselves use when they state one: a canonical path check (31), an output-size cap (21), strict parsing, no link following, a limit applied on all paths (Table 9).
- *First step*: add a line "The fix:" to the advisory template.

5. Give CWE-409 its own mitigations and compression examples.

- *Basis*: amplification records tagged CWE-409 rose from 7 of 28 (2020–2024) to 85 of 130 (2026), so more readers land on that entry (Table 11). The entry lists no mitigation of its own, and its observed examples are XML entity-expansion cases (G01).
- *With*: G09 and G18, and the amplification cases of the published inventory as candidate observed examples.
- *First step*: submit a content suggestion to the CWE team proposing compression examples and the quoted guidance as mitigations.

Network-service teams

6. Limit decoded content where network input is decoded, and name which resource each limit bounds.

- *Basis*: 72 of 195 amplification cases arrived as HTTP, WebSocket or RPC messages, and 79 sat in network servers, proxies and clients (Tables 7 and 8). Adjudication rule R7 assigns A5 to records whose limit counts compressed bytes only.
- *With*: G09 and G08.
- *First step*: list every handler that accepts compressed bodies or messages (content codings, compressed RPC or WebSocket messages). Note for each whether it limits decoded output size, memory and processing work, and whether the limits apply before, during or after decoding. A decoded-byte limit alone does not bound chained encodings, item counts or per-connection work.

Format maintainers

7. Deflate, zlib and gzip: name amplification and decoder limits in the security considerations.

- *Basis*: gzip and zlib or raw deflate carry 51 of the 195 amplification cases (26%). Their specifications discuss integrity only (G04-G06).
- *With*: the Zstandard text on decoder limits (G09) and the brotli statement of the risk (G08).
- *First step*: propose a security-considerations paragraph that cites G08 and G09, for the next revision of each document or for the format's own documentation where the specification is not revised.

8. ZIP: give readers the rule that writers already have.

- *Basis*: ZIP is the format of 268 of the 643 path-handling cases. The specification says that "The path stored MUST NOT contain a drive or device letter, or a leading slash" (G13), as a rule for writers only.
- *With*: G13 itself and CWE-22 (G02).
- *First step*: ask the specification's publisher to add, under the same section, what a reader should do with a stored name that breaks the rule.

Library maintainers

9. Publish an advisory when an ambiguity fix has a substantiated security impact.

- *Basis*: the record holds 13 semantic-gap cases, while a single study found 14 ambiguity types across 50 ZIP parsers and reported three CVEs [You2025] (Section 6.1).
- *With*: G19, a library stating which of two disagreeing ZIP records it treats as authoritative.
- *First step*: when a parser change alters which entries, boundaries or names are read, decide whether the ambiguity it resolves has a substantiated security impact and meets the applicable disclosure criteria. If it does, request a vulnerability identifier. If it does not, describe the change in the ordinary release notes.

Advisory writers

10. Name the format and how the input arrives.

- *Basis*: 243 of 937 records (26%) do not name the compressed format (Section 5.1).
- *With*: the format and carrier vocabulary of Section 3.4.
- *First step*: add "format" and "input arrives as" to the advisory template.

Recommendations 4, 5 and 10 improve what records say. They do not, by themselves, show that fewer vulnerabilities occur.

7.2 What adoption takes

The documentation-only actions (recommendations 3, 4, 5 and 10) need a table or a template field and, for default protections, one test per row marked "enabled by default". Extraction review (recommendation 2) and decoding limits (recommendations 1 and 6) need existing tests extended and new inputs that exceed each budget, and engineering only where an entry point is found unprotected; archives with links or unusual names, and large legitimate responses, may be rejected after the change. These are assumptions stated for planning, not measurements: no control was implemented or timed in this study, and no savings, return or prevention rate is claimed. The scenario table is in the data record (adoption_scenarios_v1.2.md).

8. Research Artefacts and Reuse

8.1 What is released

The data record is a dated, versioned release of the inventory and everything needed to rebuild the paper's numbers: the frame of 1,960 candidate records with the matching facts of each; the sampling strata, weights and seed; all 1,570 coded records, with 40-word mechanism texts and an uncertainty flag per case; the two coding passes, the adjudication rules and log; the codebook, the machine-readable taxonomy with stable identifiers, a schema and a data dictionary; the CWE crosswalk, the guidance records (with their verbatim quotes) and the full proposed, untested review checklist; a benchmark protocol with a deterministic split; the NVD totals, the rates, the sensitivity checks and the design-based intervals; the scripts that generate every table, figure and audit (Python standard library only; the figures need matplotlib) and an example that reproduces the key findings; the protocol and results of the blinded pass; a changelog and a correction log. It contains no test inputs and no exploit or reproduction material.

8.2 The five-minute path

1. Download the record and check it: `sha256sum -c SHA256SUMS`.
2. Run `python example_queries.py`: it reproduces the five key findings of Section 1 and exits with status 1 if a number differs.
3. Rebuild Table 6: `python scripts/rates.py inventory/inventory_v0.1.csv inventory/nvd_totals_by_year.csv rates.md`.
4. Code one new record: read `coding/codebook_for_coders_v0.1.md` once, choose the primary mechanism from `taxonomy_v1.1.json` and compare it with the crosswalk rows of that mechanism.

The scripts need Python 3 and nothing else. To evaluate an advisory-text classifier against the labels, follow the benchmark protocol of Section 8.5 and state the validation status of the labels (Section 3.5). To repeat the study on a later window, re-run `scripts/harvest.py` with a new cutoff and the same queries and compare with Table 6.

8.3 Versions, identities and citation

The record is version 1.0, frozen at records published to 30 September 2026, and is uploaded as `decompression_extraction_vulnerabilities_data_record_v1.0.zip` (1,492,450 bytes, SHA-256 `e7e037cc40fb9f9078b3bb647b22b21d3924d99067037a8e543f12f68cbc75f6`). Data, tables and documentation are CC BY 4.0, scripts MIT, and the quoted record and guidance text stays under its owners' terms. Its DOI is <https://doi.org/10.5281/zenodo.23300061> (version 1.0; the concept DOI [10.5281/zenodo.23300060](https://doi.org/10.5281/zenodo.23300060) resolves to the latest version). The record is a supplement to the paper of the same title (DataCite relation `IsSupplementTo`) and the paper is supplemented by the record (`IsSupplementedBy`); `CITATION.cff` names the record and prefers the paper; `CHANGELOG.md` and `CORRECTIONS.md` list changes and corrections.

An error is reported through the contact details on the record's landing page, with the record identifier, the field, the proposed value and the supporting passage of the record text. Accepted corrections are listed with their evidence in the correction log of the next version; codes and denominators are never changed silently, and identifiers never change meaning.

8.4 Research questions the record opens

Table 15 lists six questions, each tied to a gap the record shows. All six are untested, and none is a finding of this paper. Advisory-text classification can use the released labels; questions about static analysis also need code-level evidence and independently validated labels.

Table 15
Six open research questions follow from the gaps in the record (all untested).

observed gap	data-set evidence	open research question	required validation	available artefact
wrappers carry most path-handling cases	340 of 643 cases; guidance G16, G17, G20	how accurately can static analysis detect unsafe archive-extraction wrappers?	detector runs on code-level evidence, with independently validated labels of wrappers and their patches	labelled P1 and P2 cases, component roles
limits that miss an entry point	45 of 195 amplification cases	which output-limit enforcement strategies cover every decoding entry point?	benign tests per entry point (Protocol B, Section 9)	A4 and A5 cases, the checklist rows
semantic gaps surface outside the CVE record	13 cases; 14 ambiguity types in research [You2025]	how often do scanners and consumers interpret the same archive differently?	differential testing of parser pairs	S1 to S5 definitions and cases
mitigation descriptions are rare	77 of 937 records	which reporting fields most improve mitigation classification?	comparison with linked patches and vendor notes	the mitigation classes and the case-level labels
classification rests on two instructed passes and one author	kappa 0.81 to 0.99; the blinded pass of Section 3.5	how reproducibly can individual mechanisms be classified from advisory text?	external blinded coders on a stratified plus challenge sample	codebook, schema, example queries, uncertainty flags
weakness labels may follow practice, not mechanism	85 of 130 amplification records of 2026 tagged CWE-409; Table 13	do weakness classifications change when linked patches are inspected rather than the advisory text?	patch inspection of a sample	the crosswalk and the matching facts

8.5 Benchmark use and evaluation protocol

The record carries a protocol (BENCHMARK_PROTOCOL.md) for using the labels as a benchmark, so that results of different groups can be compared. It defines two tasks: mechanism classification of an included case from its record text (the 21 mechanisms or the four strands; 1,067 cases), and mitigation-description classification (the three completeness categories or the 14 classes; the 937 cases of the fully enumerated frame, because the mitigation field of the sampled cases was not re-read). The split is deterministic. Related advisories form a group (the same product in the same publication year); a group goes to the development set when the first byte of the SHA-256 of its key is below 179 and otherwise to the evaluation set. That gives 751 groups, 793 development and 274 evaluation cases for the first task and 704 and 233 for the second, and a published script rebuilds it. Every case carries the status of its label: agreed by both passes,

adjudicated, or corrected after the blinded pass. Users report macro and per-class results on all evaluation cases and separately on the agreed ones, with intervals that resample groups, not records. The labels are adjudicated, not externally verified ground truth.

9. Limitations and Future Validation

9.1 Limitations

- The frame depends on words and classes. Cases described in other terms are missed, and changes in wording move the counts (Section 6). Shares such as 643 of 937 describe this query-selected frame, not all decompression vulnerabilities. The inventory is not exhaustive.
- The decoder-fault strand is sampled. Its figures are weighted estimates from 200 of 590 cases, of which 130 were observed in scope. Observed sample counts are never added to the counts of the frame, and the 130 sampled cases enter no mitigation table.
- Coverage of advisories is uneven. The GitHub database reviews advisories only for the ecosystems it covers; distribution advisories enter only through their CVE identifiers.
- 2026 is incomplete, sources change and dates have definitions. The 2026 figures cover January to September. NVD and GitHub records are edited over time, so the frame retrieved on 6 October 2026 is the reference; the publication date is not the year in the CVE identifier (Section 6).
- Coding is from record text only. No case was reproduced and no linked page was opened, so the inventory inherits what a record gets wrong or leaves silent. Mitigation figures measure description completeness.
- The classification is not externally validated. The two instructed passes shared one codebook. The mitigation field carried a shared bias towards none-stated, which only the author's blinded pass exposed and the re-read of Section 3.5 removed. Both rest on the author's rules, so another shared misreading may remain.
- No recommendation or control was tested. The recommendations and the controls in Appendix A rest on the record and the guidance, not on tests. The checklist is a proposed, untested review checklist, not a validated standard.

These limits do not change three readings of the audited records: path handling dominates them in every period and application code carries most of it; records rarely state what their fixes do; the guidance reviewed is silent on seven mechanism groups.

9.2 What remains to be tested

The inventory makes these problems visible but cannot answer them:

- Control effectiveness. The study tested no software and built no test input; building the inputs such tests need is outside its scope. The record holds three protocols for later work (extraction containment, decoded-output limits and parser consistency; `validation_protocols_v1.2.md`).

- The semantic-gap strand cannot be measured from records. 13 cases is a floor, and research shows the mechanisms are widespread [You2025; Jana2012]. A measure needs a source that captures disagreements that never receive an identifier.
- Separating incidence from the record. The 2026 rise cannot be split into discovery, publication, terminology, classification and coverage. A frame not dependent on record wording, for example one built from fix commits, would help.
- Mechanisms without guidance. Seven of 15 mechanism groups have no guidance in the documents read; whether guidance elsewhere covers them needs a wider document survey.
- Efficacy. With 8% of records describing the fix, the record cannot show which mitigations work; that needs fix analysis.
- Coverage beyond two databases. Distribution trackers, vendor bulletins and bug trackers hold cases that receive neither a CVE nor a reviewed advisory.

9.3 A validation roadmap (a plan, not an executed study)

Four steps would turn the audited record into a validated one. None has been carried out.

1. External review. Independent blinded coders check a stratified sample of 50 to 100 included cases plus a challenge sample of hard cases (mechanism boundaries, mitigation descriptions, cases where the two passes disagreed). The protocol, disagreements, corrections, confusion matrices, precision and recall are published.
2. Records against remediation evidence. The mitigation and mechanism codes are compared with linked patches and vendor explanations, where accessible and ethically appropriate.
3. Benign control tests. Protocols A to C of Section 9.2, run by teams on their own software with harmless inputs.
4. Boundary review. Outside reviewers examine the decision boundaries of the taxonomy and the crosswalk.

Two refinements of the codebook wait for the next version: a record that names a client and a server together is coded by the side that decompresses untrusted input, and by network-server-or-proxy if both (rule R15(c) leaves the choice open, as with the OpenSSH record of Section 4.2); and a fix named only by a commit or a pull request counts as upgrade-only, as the re-read of Section 3.5 treated it. Independent blinded validation by outside reviewers is the first step of this roadmap and has not been carried out.

Three candidates for later papers follow from the findings: a cross-library evaluation of default extraction protections, a cross-API evaluation of decoding limits, and a study of advisory completeness against patch evidence.

10. Conclusion

Observed in the audited records: extraction path handling accounts for 643 of the 937 included cases of the fully enumerated frame, and extraction wrappers carry 340 of those 643. Amplification increasingly arrives over the network (72 of 195 cases), and in 45 of its 195 cases a size check existed but rested on a declared size or was enforced incompletely. Only 77 of the 937 records describe a mitigation mechanism.

The supported interpretation is that the failures concentrate in application code around the libraries, and that the record seldom says what a fix does. Not established: the frequency of any mechanism in deployed software, the incidence of the vulnerabilities, the effectiveness of any control, and the cause of the 2026 rise in records, which survives every sensitivity check of Section 6 but remains a change in the audited public record.

Two gaps should be closed first. Advisories should state what the fix does, and specifications and library documentation should state decoder limits and reader-side path rules. The next useful step is to validate the labels outside the author's hands (Section 9.3), and to repeat the inventory each year with the same published frame and codebook, as separately versioned updates, so that the rates in Table 6 become a series rather than a single reading.

Declarations

Data availability

The inventory is published as a dated data record, version 1.0; its DOI is <https://doi.org/10.5281/zenodo.23300061> (concept DOI 10.5281/zenodo.23300060). The window, records published from 2001 to 30 September 2026, is frozen: later records will form separately versioned updates. The record is a supplement to the paper of the same title (IsSupplementTo) and the paper is supplemented by the record (IsSupplementedBy). Section 8 lists the contents. The record contains no test inputs and no exploit or reproduction material.

CVE description text is quoted under the CVE Terms of Use, Copyright © 1999-2026, The MITRE Corporation. This product uses data from the NVD API but is not endorsed or certified by the NVD. Text from the GitHub Advisory Database is quoted under CC BY 4.0, with a link to each record.

Competing interests

The author develops commercial data-compression software. No other competing interests.

References

1. [Ali2023] S, Ali, Smith SW (2023) A Survey of Parser Differential Anti-Patterns. IEEE Secur Priv Workshops (SPW). 10.1109/SPW59333.2023.00016
2. [APPNOTE] PKWARE Inc. APPNOTE.TXT,.ZIP File Format Specification.
<https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT> (retrieved 2026-10-06)

3. [Bhuiyan2025] MHM, Bhuiyan B, Çakar EH, Burmane JC, Davis (2025) and C.-A. Staicu. SoK: A Literature and Engineering Review of Regular Expression Denial of Service (ReDoS). In ACM Asia Conference on Computer and Communications Security (ASIA CCS), arXiv:2406.11618
4. [Canet2017] M, Canet A, Kumar C, Lauradoux M-A, Rakotomanga, Safavi-Naini R (2017) Decompression Quines and Anti-Viruses. In ACM Conference on Data and Application Security and Privacy (CODASPY). 10.1145/3029806.3029818
5. [Carmony2016] C, Carmony M, Zhang X, Hu AV, Bhaskar, Yin H (2016) Extract Me If You Can: Abusing PDF Parsers in Malware Detectors. In Network and Distributed System Security Symposium (NDSS). 10.14722/ndss.2016.23483
6. [Chen2020] M, Chen (2020) and M. Fan. Detection Algorithm for Non-recursive Zip Bombs. In SECURWARE
7. [Cohen1960] J, Cohen (1960) A Coefficient of Agreement for Nominal Scales. Educ Psychol Meas 20(1):37–46. 10.1177/001316446002000104
8. [Crosby2003] SA, Crosby (2003) and D. S. Wallach. Denial of Service via Algorithmic Complexity Attacks. In 12th USENIX Security Symposium. <https://www.usenix.org/conference/12th-usenix-security-symposium/denial-service-algorithmic-complexity-attacks>
9. [CWE22] MITRE CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal'). CWE 4.20. <https://cwe.mitre.org/data/definitions/22.html> (retrieved 2026-10-06)
10. [CWE409] MITRE CWE-409: Improper Handling of Highly Compressed Data (Data Amplification). CWE 4.20. <https://cwe.mitre.org/data/definitions/409.html> (retrieved 2026-10-06)
11. [CWE59] MITRE CWE-59: Improper Link Resolution Before File Access ('Link Following'). CWE 4.20. <https://cwe.mitre.org/data/definitions/59.html> (retrieved 2026-10-06)
12. [Fifield2019] D (2019) Fifield. A better zip bomb. In 13th USENIX Workshop on Offensive Technologies (WOOT). <https://www.usenix.org/conference/woot19/presentation/fifield>
13. [GHSA] GitHub GitHub Advisory Database, reviewed advisories, REST API. <https://api.github.com/advisories> (retrieved 2026-10-06)
14. [GoTar] The Go Authors Package archive/tar. <https://pkg.go.dev/archive/tar> (retrieved 2026-10-06)
15. [GoZip] The Go Authors Package archive/zip. <https://pkg.go.dev/archive/zip> (retrieved 2026-10-06)
16. [Jana2012] S, Jana, Shmatikov V (2012) Abusing File Processing in Malware Detectors for Fun and Profit. In IEEE Symposium on Security and Privacy. 10.1109/SP.2012.15
17. [JavaZip] O Java SE 21 API, java.util.zip.ZipInputStream and java.util.zip.ZipFile. <https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/zip/ZipInputStream.html> (retrieved 2026-10-06)
18. [Lin2024] Z, Lin Z, Lin et al (2024) Unveiling the Bandwidth Nightmare: CDN Compression Format Conversion Attacks. In International Conference on Big Data and Privacy Computing (BDPC). 10.1109/BDPC59998.2024.10649043
19. [Liu2026] Y, Liu L, Zhang Y, Zhao, He Y (2026) Wormholes in the File System: Understanding the Misunderstanding of Symlinks. In 35th USENIX Security Symposium, pages 5213–5229.

- <https://www.usenix.org/conference/usenixsecurity26/presentation/liu-yongheng> (artifact: 10.5281/zenodo.20325582)
20. [Momot2016] F, Momot S, Bratus SM, Hallberg, Patterson ML (2016) The Seven Turrets of Babel: A Taxonomy of LangSec Errors and How to Expunge Them. IEEE Cybersecur Dev (SecDev). 10.1109/SecDev.2016.019
 21. [Mueller2021] J, Müller D, Noss C, Mainka V, Mladenov, Schwenk J (2021) Processing Dangerous Paths: On Security and Privacy of the Portable Document Format. In Network and Distributed System Security Symposium (NDSS). 10.14722/ndss.2021.23109
 22. [Nallamothu2026] P, Nallamothu (2026) Trusting the Filename: A Systematization and Detection Study of File-Write Attacks from Untrusted Archive and Download Metadata. Preprint, Zenodo. 10.5281/zenodo.22288349
 23. [NIST2026] National Institute of Standards and Technology NIST Updates NVD Operations to Address Record CVE Growth. News release, 15 April 2026 (updated 17 April 2026). <https://www.nist.gov/news-events/news/2026/04/nist-updates-nvd-operations-address-record-cve-growth> (retrieved 2026-10-11)
 24. [NodeZlib] OpenJS Foundation Node.js v26.10.0 documentation, Zlib (including the experimental ZIP archive API added in v26.8.0). <https://nodejs.org/api/zlib.html> (retrieved 2026-10-06)
 25. [NVD] National Institute of Standards and Technology, National Vulnerability Database CVE API 2.0. <https://services.nvd.nist.gov/rest/json/cves/2.0> (retrieved 2026-10-06)
 26. [Pellegrino2015] G, Pellegrino D, Balzarotti S (2015) Winter and N. Suri. In the Compression Hornet's Nest: A Security Study of Data Compression in Network Services. In 24th USENIX Security Symposium. <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pellegrino>
 27. [PyTarfile] Python Software Foundation tarfile: Read and write tar archive files. Python 3.14 documentation. <https://docs.python.org/3/library/tarfile.html> (retrieved 2026-10-06)
 28. [PyZipfile] Python Software Foundation zipfile: Work with ZIP archives. Python 3.14 documentation. <https://docs.python.org/3/library/zipfile.html> (retrieved 2026-10-06)
 29. [RFC1950] P, Deutsch, Gailly J-L (1996) ZLIB Compressed Data Format Specification version 3.3. RFC 1950
 30. [RFC1951] P, Deutsch (1996) DEFLATE Compressed Data Format Specification version 1.3. RFC 1951
 31. [RFC1952] P, Deutsch (1996) GZIP file format specification version 4.3. RFC 1952
 32. [RFC7692] T (2015) Yoshino. Compression Extensions for WebSocket. RFC 7692
 33. [RFC7932] J, Alakuijala, Szabadka Z (2016) Brotli Compress Data Format RFC 7932
 34. [RFC8878] Y, Collet, Kucherawy M (2021) Zstandard Compression and the 'application/zstd' Media Type. RFC 8878
 35. Fielding [RFC9110]R, Nottingham M, Reschke J (2022) HTTP Semant RFC 9110
 36. [Snyk2018] Snyk. Zip Slip Vulnerability. Security research and disclosure, 5 (2018) June <https://security.snyk.io/research/zip-slip-vulnerability> (retrieved 2026-10-11)
 37. [VanMello2024] (2024) Van Mello and Audenaert. A Generator for Recursive Zip Files. Appl Sci. 10.3390/app14219797

38. [Velinov2022] A, Velinov, Mileva A (2022) POSTER: Launching a ZIP Bomb on the DICOM-enabled Devices. In European Interdisciplinary Cybersecurity Conference (EICC). 10.1145/3528580.3532995
39. [Wang2019] H, Wang H, Liu X, Xiao G, Meng, Guo Y (2019) Characterizing Android App Signing Issues. In IEEE/ACM International Conference on Automated Software Engineering (ASE). 10.1109/ASE.2019.00035
40. [You2025] Y, You J, Chen Q, Wang, Duan H (2025) My ZIP isn't your ZIP: Identifying and Exploiting Semantic Gaps Between ZIP Parsers. In 34th USENIX Security Symposium, pages 431–450. <https://www.usenix.org/conference/usenixsecurity25/presentation/you>

Figures

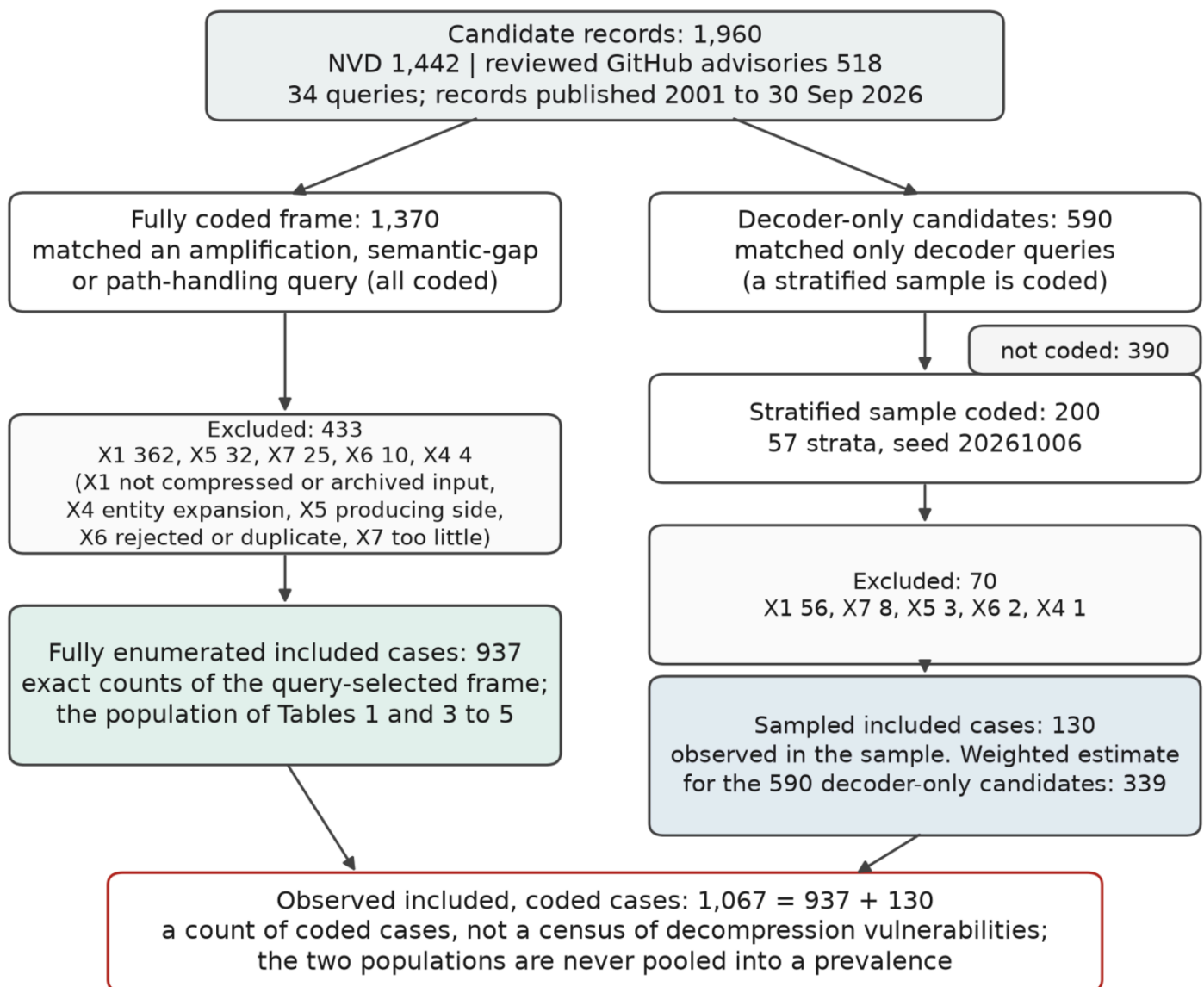


Figure 1

The 1,067 observed included cases come from two populations, 937 enumerated and 130 sampled, and are never pooled into a prevalence. N = 1,960 candidate records (NVD and reviewed GitHub advisories, published 2001 to 30 September 2026); unit = record; source = the record's population flow file.

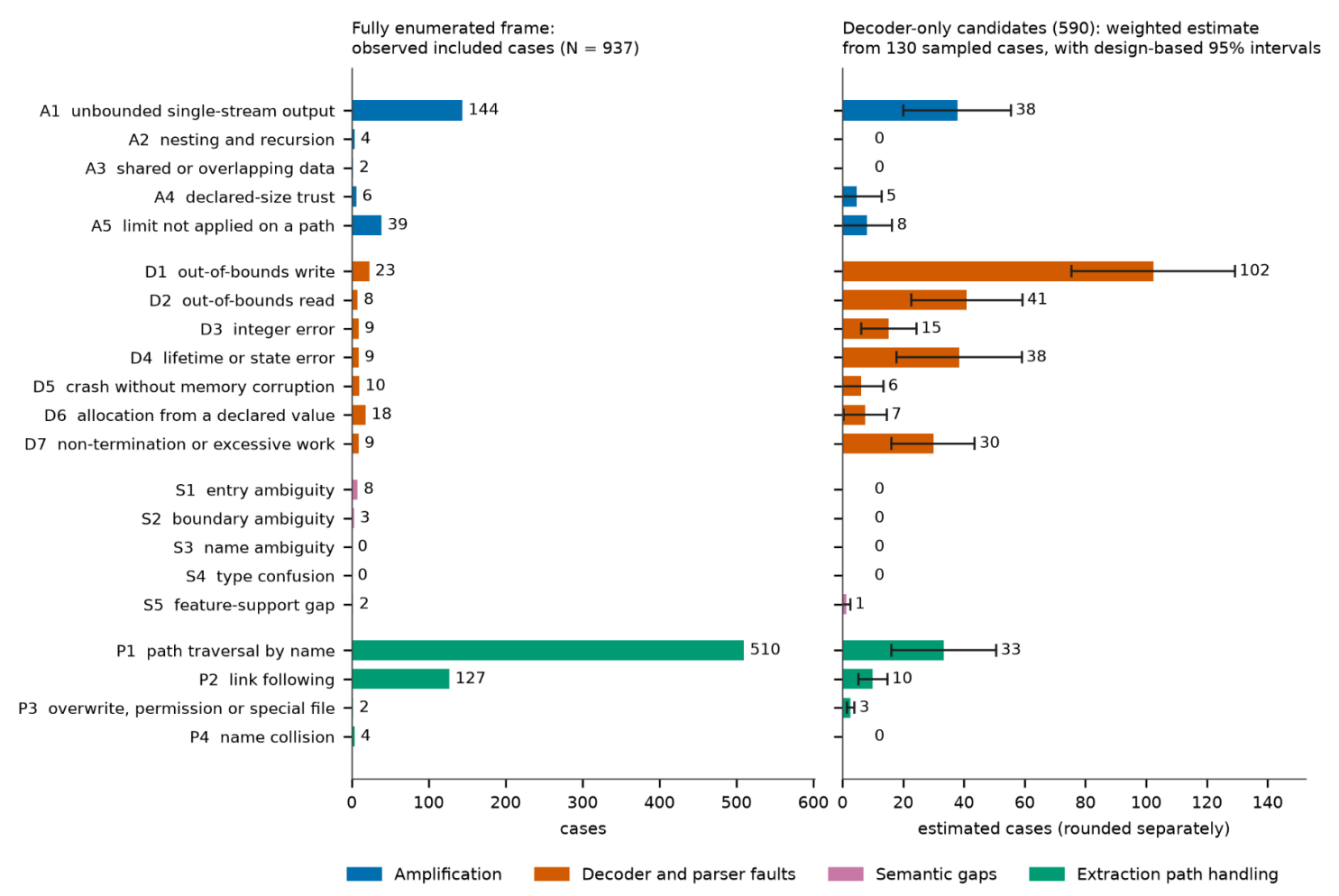


Figure 2

Path traversal by name dominates the fully enumerated frame (510 of 937), while the decoder-only estimate is led by out-of-bounds writes (102 of 339). Left panel: observed included cases, N = 937. Right panel: weighted estimates for the 590 decoder-only candidates, from 130 sampled cases, with design-based 95% intervals. The panels are never added.

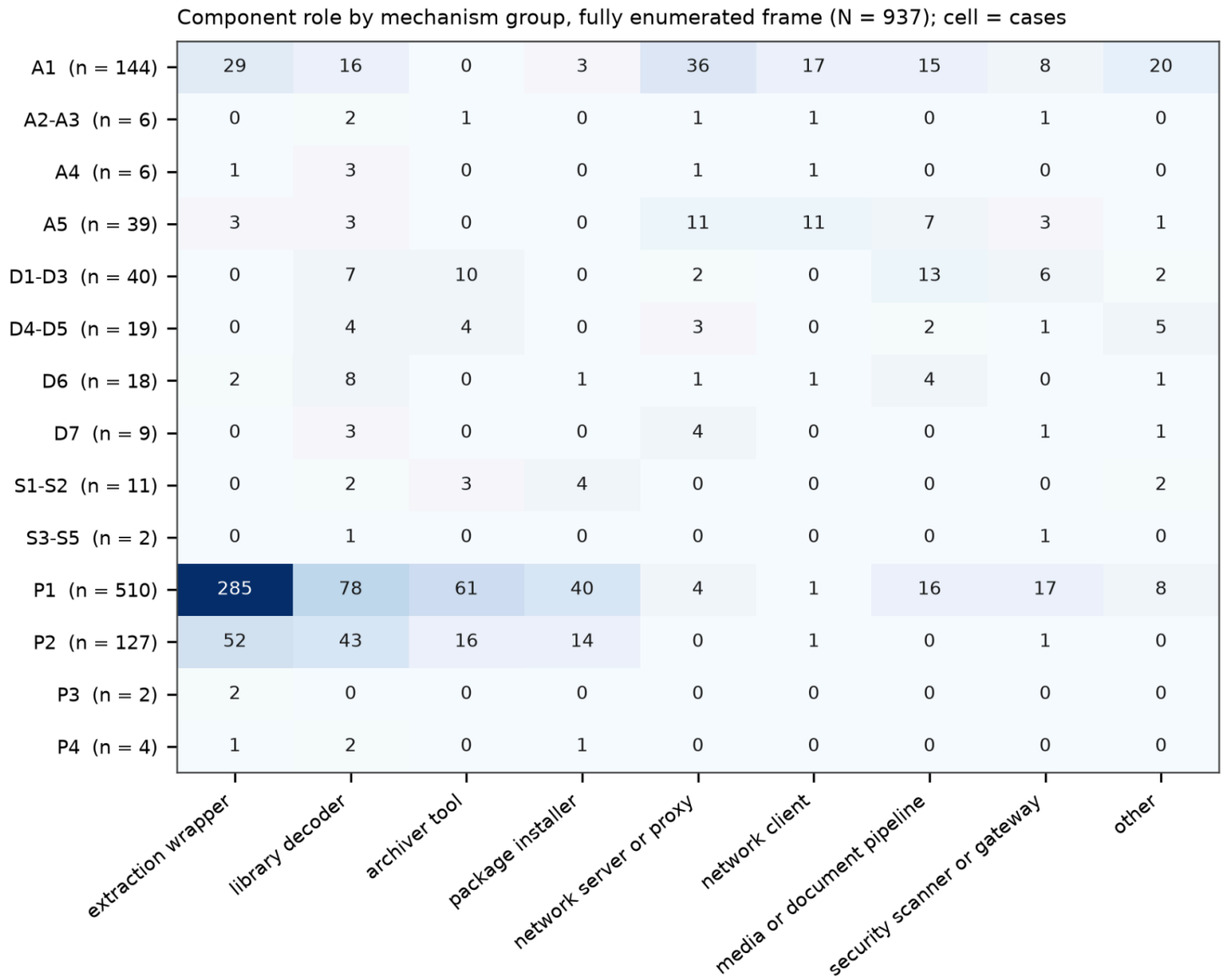


Figure 3

Extraction wrappers carry most path-handling cases, 285 of the 510 coded P1. N = 937 included cases of the fully enumerated frame; unit = case; cell = number of cases; source = the coded inventory.

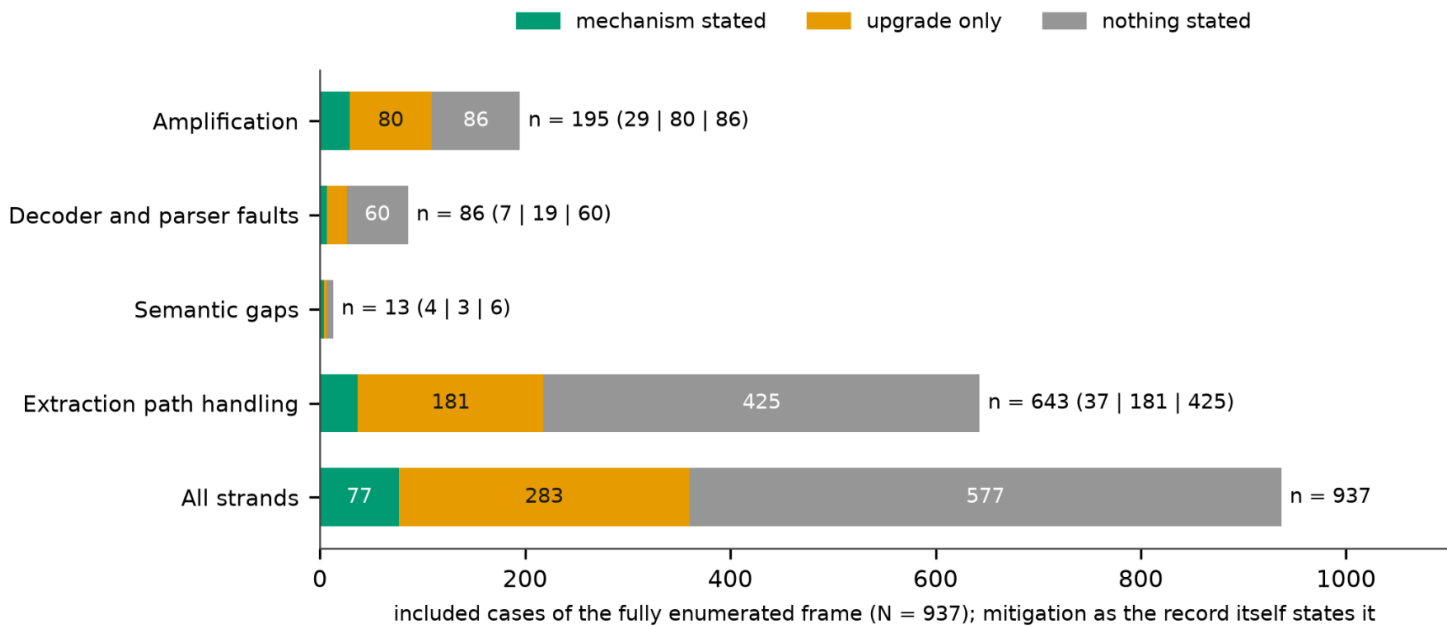


Figure 4

Path handling states a mitigation mechanism least often, 37 of 643, and the record states nothing in 425 of them. N = 937 included cases of the fully enumerated frame; unit = case; categories as the record states them; source = the coded inventory.

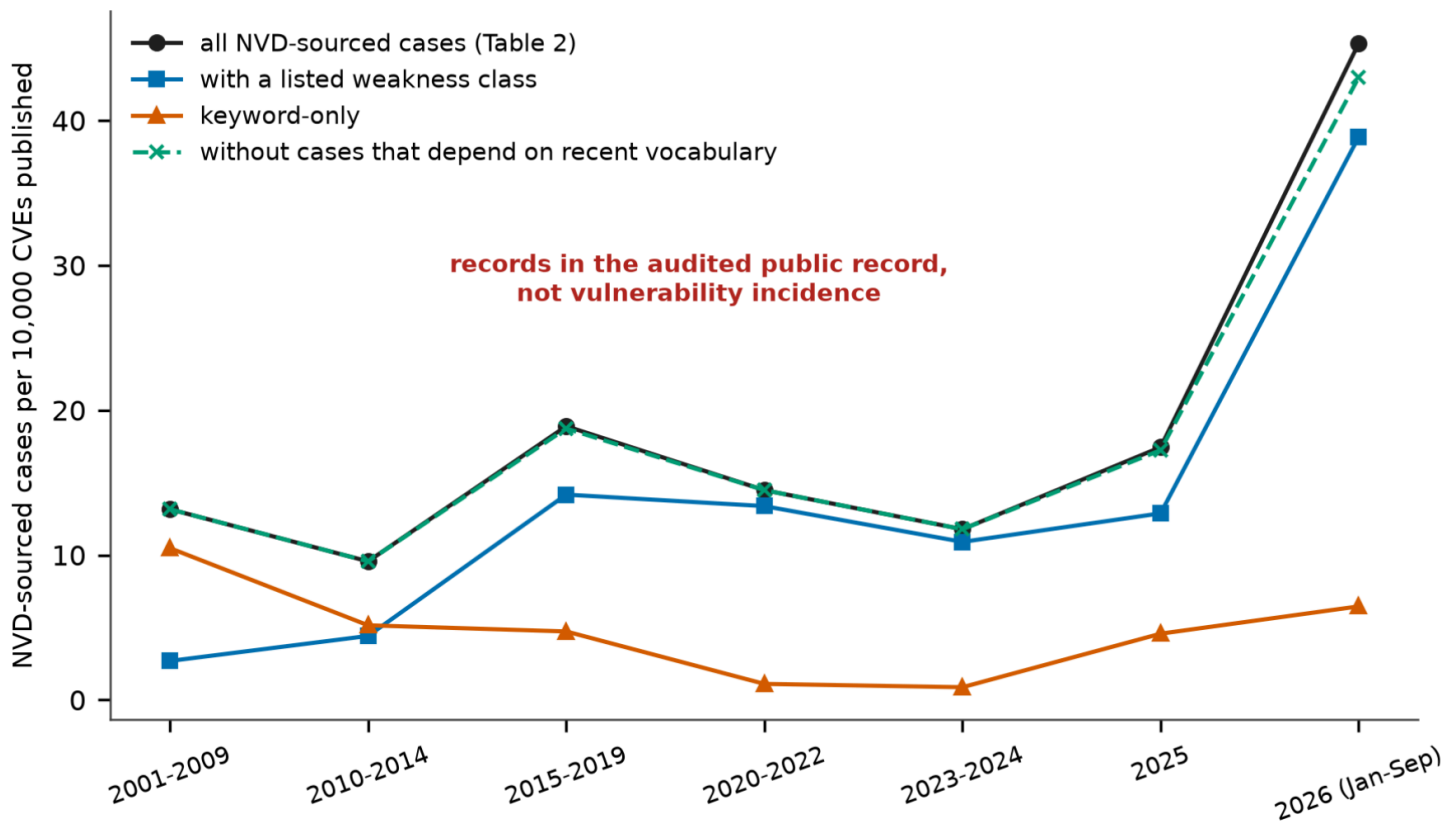


Figure 5

The 2026 rise holds with or without records that depend on recent vocabulary, but sits in records that carry a listed weakness class. Rate per 10,000 CVEs published, NVD-sourced included cases of the fully enumerated frame (N = 778); lines show the checks of Tables 12 and 13; the figure describes records in the audited public record, not vulnerability incidence.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [AppendixA.docx](#)