Network Working Group
Internet-Draft
Intended status: Informational
Expires: 11 October 2026

J. Carroll
The Clearing Room

9 April 2026

Triageable Evidence Format (TEF)
draft-tcr-tef-00

Abstract

This document defines the Triageable Evidence Format (TEF), a YAML-based format for structured vulnerability evidence submission. TEF provides a standard way for security researchers to describe software defects so that triage systems, whether human or automated, can classify them without ambiguity.

TEF combines YAML data structure with a Given/When/Then evidence model. Each submission carries structured metadata, one or more evidence scenarios with preconditions, triggers, and observed outcomes, location data identifying where the defect exists, and reproduction artefacts proving its presence.

TEF is an intake format. It does not replace vulnerability publication formats (CVE, CSAF, OSV) or scan output formats (SARIF). It fills the gap between discovery and classification.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 October 2026.

Copyright Notice

Table of Contents

1. Introduction

Vulnerability disclosure relies on structured formats at the publication stage (CVE records, CSAF advisories, OSV entries) and at the scanning stage (SARIF). There is no equivalent standard for the intake stage: what a researcher submits when reporting a software defect to a coordination body, vendor, or disclosure platform.

Most intake systems accept free text. This creates ambiguity, inconsistency, and friction. A researcher describes their finding in prose. A triager interprets the prose, potentially misunderstanding preconditions, confusing the trigger with the outcome, or missing evidence entirely. At scale, this process does not work.

TEF addresses this gap by defining a structured, machine-parseable, human-readable format for vulnerability evidence.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Format Overview

A TEF document is a YAML file containing two required top-level keys: "defect" and "evidence".

The "defect" key contains structured metadata about the affected product, vendor, target type, weakness classification, affected versions, and defect manifestation pattern.

The "evidence" key contains one or more scenarios, each describing a distinct piece of evidence using a Given/When/Then structure, an optional location block, and an optional reproduction block.

3. Schema

3.1. Defect Block (REQUIRED)

The "defect" block MUST contain the following fields:

* "product" (string, REQUIRED): The name of the affected product.

* "vendor" (string, REQUIRED): The vendor domain or identifier.

* "type" (string, REQUIRED): The target type. See Section 4.

* "cwe" (integer, REQUIRED): The CWE identifier for the weakness class.

* "versions" (array of strings, REQUIRED): The affected version range.

* "presence" (string, REQUIRED): The defect manifestation pattern. See Section 5.

The "defect" block MAY contain:

* "os" (string): The operating system or platform.

* "description" (string): Freeform context.

* "cvss" (string): CVSS vector or score.

* "references" (array of strings): URLs, CVE IDs, advisory links.

* "reproduction_rate" (string): For probabilistic or timing-dependent defects (e.g., "15/100").

* "environment" (string): Specific environment if relevant.

3.2. Evidence Block (REQUIRED)

The "evidence" block MUST contain one or more scenario entries. Each scenario MUST contain:

* "title" (string, REQUIRED): A short name for the scenario.

* "given" (array of strings, REQUIRED): Preconditions that must be true before the trigger.

* "when" (array of strings, REQUIRED): The action or input that reveals or triggers the defect.

* "then" (array of strings, REQUIRED): The observed outcome that proves the defect exists.

Each scenario MAY contain:

* "location": A block identifying where the defect exists. See Section 6.

* "reproduction": A block carrying proof artefacts. See Section 7.

4. Target Types

The "type" field in the "defect" block MUST be one of the following values: "web", "api", "mobile_android", "mobile_ios", "mobile", "desktop", "server", "os", "embedded", "network", "cloud", "ot", "ai_ml", "supply_chain".

Producers SHOULD use "mobile_android" or "mobile_ios" rather than "mobile" when the defect is platform-specific. The "desktop" type refers to the application, not the operating system. The "server" type covers software running as a service. Web applications running on server software are "web".

5. Presence Patterns

The "presence" field MUST be one of: "static_artefact", "static_config", "behavioural_deterministic", "behavioural_stateful", "behavioural_timing", "probabilistic", "environmental", "design_flaw", "supply_chain".

"static_artefact": The defect exists in the artefact itself and can be confirmed without execution.

"static_config": Observable in configuration, headers, DNS, or policy without exploitation.

"behavioural_deterministic": Requires execution but same input always produces same result.

"behavioural_stateful": Requires specific application state or multi-step interaction.

"behavioural_timing": Depends on timing or concurrency. Submissions SHOULD include "reproduction_rate".

"probabilistic": Same input may produce different outcomes because the system is non-deterministic. Distinct from timing-dependent. Submissions SHOULD include "reproduction_rate".

"environmental": Only manifests in specific environments. Submissions SHOULD include comparison.

"design_flaw": The design itself is insecure. No single code fix resolves it.

"supply_chain": Inherited from a third-party component or dependency.

6. Location Block

The "location" block identifies where the defect exists. It MAY appear on any evidence scenario.

* "type" (string, REQUIRED): One of "endpoint", "file", "function", "binary_offset", "register", "port", "config", "path".

* "value" (string, REQUIRED): The specific location.

* "context" (string, OPTIONAL): Broader context.

7. Reproduction Block

The "reproduction" block carries raw proof artefacts. All fields are OPTIONAL.

* "commands" (array of strings): Exact commands executed.
* "code" (string): Vulnerable code snippet or exploit script.
* "request" (string): Raw protocol request.
* "response" (string): Raw protocol response.
* "logs" (string): Relevant log entries.
* "output" (string): Tool output, crash dumps, extraction results.
* "files" (array of objects): References to attached evidence files, each with "name" (string), "hash" (string, SHA-256), and "type" (string).

8. Defect Classes

TEF recognises 19 defect classes across two categories. The defect class is not required but is RECOMMENDED. Producers MAY use the "bugClass" field.

Code defects: injection, auth, crypto, data_exposure, access_control, deserialization, file_handling, ssrf, memory, logic, supply_chain, ai_ml.

Configuration defects: config_tls, config_headers, config_secrets, config_permissions, config_logging, config_dns, config_default.

The "ai_ml" class covers defects in AI and machine learning systems including prompt injection, training data poisoning, model extraction, adversarial inputs, excessive agent permissions, RAG corpus manipulation, and model supply chain compromise.

9. Conformance

A TEF producer MUST include all REQUIRED fields. A TEF consumer MUST accept documents with unknown fields and MUST NOT reject them.

A TEF document is valid if: (1) it is valid YAML, (2) the "defect" block contains all REQUIRED fields, (3) the "evidence" block contains at least one scenario, (4) each scenario contains "title", "given", "when", and "then", (5) the "type" value is from Section 4, (6) the "presence" value is from Section 5.

10. Relationship to Other Formats

TEF is an intake format. It captures what a researcher observed, structured for triage. After triage, findings are published as CVE records, CSAF [CSAF] advisories, or correlated with SARIF [SARIF] output. The Common Weakness Enumeration [CWE] provides the defect taxonomy. TEF does not duplicate these formats. It precedes them in the vulnerability lifecycle.

11. Security Considerations

TEF documents may contain offensive security content: exploit code, injection payloads, shell commands, memory corruption details, and proof-of-concept material. Systems processing TEF documents MUST treat all user-supplied content as untrusted input.

TEF documents SHOULD NOT contain plaintext credentials or unredacted sensitive data belonging to third parties. Producers SHOULD redact sensitive values while preserving enough information for verification.

The "reproduction" block may reference attached files by SHA-256 hash. Consumers MUST NOT execute attached files without explicit user authorisation.

12. IANA Considerations

This document has no IANA actions. A media type registration for "application/tef+yaml" may be requested in a future version.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[CWE] MITRE, "Common Weakness Enumeration", <https://cwe.mitre.org/>.

[SARIF] OASIS, "Static Analysis Results Interchange Format (SARIF) Version 2.1.0", 2020, <https://docs.oasis-open.org/sarif/sarif/v2.1.0/sarif-v2.1.0.html>.

[CSAF] OASIS, "Common Security Advisory Framework (CSAF) Version 2.0", 2022, <https://docs.oasis-open.org/csaf/csaf/v2.0/csaf-v2.0.html>.

Acknowledgments

The TEF format was developed through implementation experience at The Clearing Room, an independent vulnerability disclosure clearing house. Feedback from the security research community, particularly on presence pattern classification and AI/ML defect coverage, shaped the current specification.

Examples

Complete TEF examples covering all target types and presence patterns are maintained at the TEF template library: https://theclearingroom.io/templates

Author’s Address

John Carroll
The Clearing Room
Email: jc@thecontractor.io