GCVE-BCP-12 - Sighting Format

GCVE.eu

GCVE.eu

Contents

0.1 GCVE-BCP-12 - Sighting Format

1 Sighting Format

- **Version**: 0.9
- **Status**: Draft (for Public Review)
- **Date**: 2026-07-30
- **Authors**: GCVE Working Group
- **BCP ID**: BCP-12

1.1 Introduction

A vulnerability sighting is a structured observation associating a real-world signal with a vulnerability identifier.

Sightings complement vulnerability advisories by recording information that may evolve independently from the vulnerability record itself. Examples include:

- a vulnerability being mentioned or discussed;
- the publication of a proof of concept;
- analyst confirmation or rejection of a vulnerability report;
- observed exploitation;
- verification that a vulnerability was not exploited in a particular environment;
- successful or unsuccessful patching.

A sighting is an assertion made by an observer at a particular point in time. It is not necessarily an authoritative or globally valid statement about the vulnerability.

Different observers may publish multiple sightings for the same vulnerability. These sightings may overlap, complement one another, or appear to conflict because they were produced from different environments, evidence, time periods, or analytical perspectives.

This Best Current Practice defines:

1. a minimal and interoperable core sighting object;
2. the meaning of the existing sighting types;
3. compatibility requirements for the existing Vulnerability-Lookup implementation;
4. an extension mechanism that does not modify the core sighting object;
5. recommendations for corrections, retractions, and future format evolution.

The core format defined by this BCP is based on the existing Vulnerability-Lookup sighting format. Existing Vulnerability-Lookup implementations and sighting-generation tools can continue to operate without modification.

1.2 Design Principles

BCP-12 follows these principles.

1.2.1 Preserve Existing Implementations

The existing Vulnerability-Lookup sighting object is the BCP-12 core format.

BCP-12 does not rename existing fields, alter their data types, add mandatory fields, change the sighting type enumeration, or permit additional properties inside the core object.

1.2.2 Keep the Core Format Small

The core sighting object represents the minimum information necessary to associate an observation with:

- a vulnerability;
- an observer;
- an origin;
- a sighting type;
- a timestamp;
- an optional source and description.

More detailed information, such as confidence, evidence, observation scope, affected assets, geographic information, or external identifiers, belongs in optional extension fragments.

1.2.3 Treat Sightings as Assertions

A sighting records what an observer reported. It does not establish universal ground truth.

For example, an `exploited` sighting means that exploitation was reported by the observer. It does not mean that every vulnerable system was exploited.

Similarly, a `not-exploited` sighting means that exploitation was not found in the context examined by the observer. It does not disprove exploitation reported by another observer or in another environment.

1.2.4 Keep Core Sightings Stable

Once published, a sighting should be treated as an immutable observation.

Corrections, additional context, retractions, and superseding information should be expressed separately rather than silently rewriting the original observation.

1.2.5 Permit Independent Evolution

Future extensions must not require changes to existing implementations.

Structured extensions are therefore published as separate objects linked to the UUID of the core sighting. Existing consumers may ignore extension fragments while continuing to process the core sighting normally.

1.3 Terminology

1.3.1 Sighting

A structured assertion associating an observation with a vulnerability identifier.

1.3.2 Core Sighting

The minimal JSON object defined by this BCP and compatible with the existing Vulnerability-Lookup sighting schema.

1.3.3 Observer

The person, organisation, service, sensor, automation tool, or other entity responsible for the observation.

1.3.4 Author

The identity, represented by a UUID, under which the sighting was published by the origin instance.

The author may represent a human user or an automation account.

1.3.5 Origin

The Vulnerability-Lookup instance that created or originally published the sighting.

The origin is identified by the `vulnerability_lookup_origin` UUID.

1.3.6 Source

A human-readable or machine-readable indication of where the observation originated.

A source may be a URI, document identifier, MISP event UUID, tool name, feed name, case identifier, or another string.

1.3.7 Extension Fragment

A separate JSON object that provides structured information related to a core sighting without modifying the core sighting itself.

1.3.8 Producer

A system or person that creates or publishes sightings.

1.3.9 Consumer

A system or person that imports, validates, analyses, displays, or otherwise processes sightings.

1.4 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 RFC 2119 and RFC 8174 when, and only when, they appear in uppercase.

1.5 Relationship to Vulnerability Records

A vulnerability record describes the identity and known characteristics of a vulnerability.

A sighting describes an observation related to that vulnerability.

Sightings MUST remain separate from the identity of the vulnerability because:

- observations can occur multiple times;
- different observers can reach different conclusions;
- exploitation and patching status can change over time;
- observations can be limited to a specific environment;
- a sighting may exist before a complete vulnerability advisory is available.

The `vulnerability` field links the sighting to the vulnerability record.

The identifier MAY be a GCVE, CVE, GHSA, PYSEC, GSD, vendor advisory identifier, or another vulnerability identifier supported by the producer and consumer.

A sighting MUST NOT be interpreted as changing the content of the referenced vulnerability record.

1.6 Relationship to GCVE-BCP-07

GCVE-BCP-07 defines a format for publishing Known Exploited Vulnerability assertions.

An `exploited` sighting and a BCP-07 KEV assertion are related but distinct concepts:

- a sighting represents an individual observation;
- a KEV assertion represents a conclusion or declaration that a vulnerability is known to be exploited;
- one or more sightings may be used as evidence supporting a KEV assertion;
- an individual `exploited` sighting does not automatically make a vulnerability a KEV;
- a KEV publisher remains responsible for assessing the evidence and publishing the BCP-07 assertion.

Sightings can therefore provide evidence or operational context for BCP-07 without replacing the KEV format.

1.7 Core Sighting Format

A core sighting is a single JSON object as defined by ECMA-404. Collections of sightings MAY be transported as NDJSON, with exactly one complete core sighting object per line. Blank lines SHOULD NOT be included. The order of records MUST NOT convey semantic meaning.

The core object has the following fields:

Field	Type	Required	Description
`uuid`	UUID	yes	Stable identifier of the sighting.
`vulnerability_lookup_origin`	UUID	yes	Identifier of the Vulnerability-Lookup instance that originated the sighting.
`author`	UUID	yes	Identifier of the author or automation account that created the sighting.
`vulnerability`	string	yes	Vulnerability identifier associated with the sighting.
`type`	string enumeration	yes	Semantic type of the observation.
`creation_timestamp`	date-time	yes	Timestamp assigned when the sighting was created or recorded by its origin.
`source`	string, maximum 2048 characters	no	Source from which the sighting was derived.
`content`	string	no	Optional free-form description or source content.

The core sighting object is closed.

Properties other than those defined above MUST NOT appear in the core object.

In particular, extension fields, including fields prefixed with `x_`, MUST NOT be inserted into the core sighting object.

1.7.1 `uuid`

The `uuid` field uniquely identifies the sighting.

The value:

- MUST be a valid UUID;
- SHOULD be a UUID version 4;
- MUST remain stable after publication;
- MUST NOT be reused for a different sighting.

Two sightings with different UUIDs are distinct assertions, even when their other fields are identical.

Implementations MAY detect and suppress operational duplicates, but duplicate detection does not change the identity rules defined by this BCP.

1.7.2 `vulnerability_lookup_origin`

The `vulnerability_lookup_origin` field identifies the Vulnerability-Lookup instance where the sighting originated.

The field name is retained for compatibility with existing implementations.

The value:

- MUST be a valid UUID;
- SHOULD correspond to the stable instance UUID configured by the originating Vulnerability-Lookup instance;
- MUST identify the original producer rather than an instance that merely replicated or imported the sighting;
- MUST NOT be replaced when the sighting is redistributed.

A consumer that republishes an imported sighting SHOULD preserve both the original `uuid` and `vulnerability_lookup_origin`.

1.7.3 `author`

The `author` field identifies the account that created the sighting at the origin instance.

The value:

- MUST be a valid UUID;
- MAY identify a human user;

- MAY identify an automation account, service, collector, or import process;
- does not need to be globally resolvable outside the originating instance.

The author UUID SHOULD be treated as a pseudonymous identifier unless the origin explicitly provides additional identity information.

API responses MAY provide an enriched representation of the author, such as a login or display name, outside the canonical core sighting object. Such API-specific representations are not part of the BCP-12 core format.

1.7.4 `vulnerability`

The `vulnerability` field contains the identifier of the vulnerability associated with the sighting.

The value:

- MUST be a non-empty string;
- SHOULD use the canonical representation and casing of the relevant identifier namespace;
- SHOULD reference a vulnerability record when one is available;
- MAY reference a reserved, preliminary, unpublished, or locally recognised identifier when the producer supports such identifiers.

Consumers SHOULD compare identifiers according to the rules of their respective namespaces.

Consumers MUST NOT assume that every sighting references a publicly available advisory.

1.7.5 `type`

The `type` field describes the semantic category of the sighting.

The value MUST be one of:

- `seen`
- `confirmed`
- `not-confirmed`
- `published-proof-of-concept`
- `exploited`
- `not-exploited`
- `patched`
- `not-patched`

The enumeration is closed in the BCP-12 core format.

New experimental or domain-specific classifications MUST be expressed through extension fragments rather than by adding values to the core enumeration.

1.7.6 `creation_timestamp`

The `creation_timestamp` field contains the time at which the sighting was created or recorded by its origin.

The value:

- MUST be a date-time compatible with RFC 3339;
- SHOULD include a timezone;
- SHOULD be normalised to UTC;
- SHOULD use the `Z` suffix when represented in UTC.

The timestamp does not necessarily represent the time at which the underlying event occurred.

For example, a social-network message may have been published several hours before a collector created the sighting.

When the time of the observed event is known and differs from `creation_timestamp`, it SHOULD be represented in a structured extension field such as `observed_at`.

Some historical producers may have used `creation_timestamp` to store the timestamp of the underlying event. Consumers processing historical data MUST therefore avoid treating `creation_timestamp` as a precise forensic event timestamp unless the producer's behaviour is known.

1.7.7 `source`

The optional `source` field indicates where the observation came from.

The value:

- MUST be a string;
- MUST NOT exceed 2048 characters;
- MAY be a URI;
- MAY be a tool or feed name;
- MAY be an external object identifier;
- MAY be a MISP event UUID;
- MAY be an incident, case, or internal reference;

- MAY be an empty string.

A consumer MUST NOT assume that `source` is a URL.

A consumer MUST NOT automatically retrieve or execute content referenced by `source` without applying appropriate security controls.

When a public proof of concept is reported, the producer SHOULD place the public reference in `source` when possible.

1.7.8 `content`

The optional `content` field contains free-form information associated with the sighting.

The value:

- MUST be a string;
- MAY contain a description, excerpt, message body, analyst note, or other textual information;
- MAY be an empty string;
- MUST be treated as untrusted input by consumers.

`content` is not a structured extension mechanism.

Producers SHOULD NOT place JSON-encoded extension objects in `content`. Structured information should be published as an extension fragment.

Consumers MUST safely escape or sanitise `content` before rendering it as HTML, Markdown, SVG, or another active format.

1.8 Sighting Types

1.8.1 `seen`

The vulnerability was mentioned, discussed, referenced, or otherwise observed by the reporter.

This is the broadest sighting type.

A `seen` sighting does not imply that:

- the vulnerability was independently confirmed;
- a proof of concept exists;
- exploitation occurred;
- an affected system was identified.

Examples include:

- a social-network post mentioning a vulnerability;
- a public mailing-list discussion;
- a security blog post;
- a reference appearing in a source-code repository;
- an automated tool observing the identifier in collected data.

1.8.2 `confirmed`

The observer validated the vulnerability or the associated claim from an analytical perspective.

The exact validation scope may vary between observers.

A `confirmed` sighting does not imply exploitation.

When the validation method or scope is important, it SHOULD be expressed in `content` or an extension fragment.

1.8.3 `not-confirmed`

The observer could not validate the vulnerability or expresses doubt about the associated claim.

This type does not withdraw or invalidate the referenced vulnerability record.

It represents the observer's conclusion based on the evidence available to that observer.

The source, context, and validation limitations SHOULD be provided when possible.

1.8.4 `published-proof-of-concept`

A publicly accessible proof of concept has been published for the vulnerability.

The proof of concept may be:

- exploit code;
- a demonstration;
- a test case;
- a scanner template;
- a reproducible technical description.

The existence of a proof of concept does not necessarily mean that exploitation has occurred in operational environments.

The `source` field SHOULD reference the proof of concept or the publication describing it.

1.8.5 `exploited`

The observer reports that exploitation of the vulnerability was observed.

The observation may originate from:

- incident response;
- forensic analysis;
- telemetry;
- honeypots;
- network sensors;
- endpoint detection;
- vendor reporting;
- another credible source.

An `exploited` sighting is scoped to the evidence and environment available to the observer.

It MUST NOT be interpreted as a universal claim that every vulnerable system has been exploited.

Where possible, evidence type, confidence, observation time, and scope SHOULD be expressed in an extension fragment.

1.8.6 `not-exploited`

The observer checked for exploitation and did not find evidence of exploitation within the examined scope.

This is a scoped negative observation.

It MUST NOT be interpreted as proof that exploitation has never occurred elsewhere or at another time.

A useful `not-exploited` sighting SHOULD describe:

- the environment examined;
- the period covered;
- the detection or investigation method;
- relevant limitations.

This context SHOULD be expressed through an extension fragment when structured representation is needed.

1.8.7 `patched`

The observer reports that the vulnerability was successfully patched or otherwise remediated within the examined environment.

This does not mean that every affected system or product deployment has been patched.

A `patched` sighting SHOULD describe the relevant product, asset, version, environment, or remediation when that information can be shared.

1.8.8 `not-patched`

The observer reports that the vulnerability was checked and was not successfully patched or remediated within the examined environment.

This may indicate:

- the patch was not installed;
- the remediation failed;
- the vulnerable version remained present;
- a mitigation was incomplete;
- verification could not confirm successful remediation.

The scope and verification method SHOULD be supplied when possible.

1.9 Independence of Sighting Types

The sighting types do not form a strict progression or hierarchy.

For example:

- `confirmed` does not imply `exploited`;
- `published-proof-of-concept` does not imply `exploited`;
- `seen` does not imply `confirmed`;
- `patched` does not invalidate an earlier `exploited` sighting;
- `not-exploited` from one observer does not invalidate `exploited` from another observer;
- `not-confirmed` does not delete or reject the underlying vulnerability record.

Multiple sighting types may legitimately be published for the same vulnerability.

Consumers SHOULD evaluate each sighting using its source, author, origin, timestamp, and available context.

1.10 Canonical Core Sighting Example

```
{
  "uuid": "d292fe1c-b3b8-4d88-984d-aaa3680c92ff",
  "vulnerability_lookup_origin": "1a89b78e-f703-45f3-bb86-59eb712668bd",
  "author": "9f56dd64-161d-43a6-b9c3-555944290a09",
  "vulnerability": "CVE-2026-3323",
  "type": "seen",
  "source": "https://example.org/security/advisories/example-2026-01",
  "content": "The vulnerability was referenced in a newly published security advisory.",
  "creation_timestamp": "2026-05-04T09:16:09.070432Z"
}
```

1.11 Minimal Submission Example

The canonical core sighting contains all required fields.

An API submission profile MAY permit server-managed fields to be omitted. For example, an implementation may assign:

- `uuid`;
- `vulnerability_lookup_origin`;
- `author`;
- `creation_timestamp`.

A minimal submission accepted by such an implementation may resemble:

```
{
  "vulnerability": "CVE-2026-3323",
  "type": "seen",
  "source": "https://example.org/security/advisories/example-2026-01"
}
```

Submission payloads are API-specific and are not themselves canonical BCP-12 sighting objects.

After acceptance, the server MUST produce a complete core sighting object before publication or export.

1.12 Core JSON Schema

The following JSON Schema describes the BCP-12 core sighting object.

It intentionally retains JSON Schema Draft 7 and a closed object model for compatibility with the existing Vulnerability-Lookup sighting schema.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "https://gcve.eu/schemas/bcp-12-sighting-core-1.0.schema.json",
  "type": "object",
  "title": "GCVE-BCP-12 core sighting format",
  "description": "A minimal and legacy-compatible vulnerability sighting object.",
  "required": [
    "uuid",
    "vulnerability_lookup_origin",
    "author",
    "vulnerability",
    "type",
    "creation_timestamp"
  ],
  "additionalProperties": false,
  "properties": {
    "uuid": {
      "type": "string",
      "format": "uuid",
      "description": "Stable UUID of the sighting.",
      "readOnly": true
    },
    "vulnerability_lookup_origin": {
      "type": "string",
      "format": "uuid",
      "description": "UUID of the Vulnerability-Lookup instance where the sighting originated.",
      "readOnly": true
    },
    "author": {
      "type": "string",
      "format": "uuid",
      "description": "UUID of the author or automation account that created the sighting.",
      "readOnly": true
    },
    "source": {
      "type": "string",
      "maxLength": 2048,
      "description": "Source of the sighting, such as a URI, feed, tool, MISP event UUID, or external identifier."
    },
    "creation_timestamp": {
      "type": "string",
      "format": "date-time",
      "description": "Timestamp assigned when the sighting was created
```

```
            or recorded by the origin.",
      "readOnly": true
    },
    "type": {
      "type": "string",
      "enum": [
        "seen",
        "published-proof-of-concept",
        "exploited",
        "not-exploited",
        "confirmed",
        "not-confirmed",
        "patched",
        "not-patched"
      ],
      "default": "seen",
      "description": "Semantic type of the sighting."
    },
    "vulnerability": {
      "type": "string",
      "description": "Vulnerability identifier associated with the
          sighting."
    },
    "content": {
      "type": "string",
      "description": "Optional free-form content describing the
          sighting.",
      "default": ""
    }
  }
}
```

1.13 Extensions

1.13.1 General Principle

The core sighting format is intentionally closed.

Structured extensions MUST NOT be added directly to the core sighting object because existing implementations reject unknown properties.

An extension MUST instead be represented as a separate extension fragment linked to the core sighting through its UUID.

This approach allows:

- existing implementations to remain unchanged;

- legacy consumers to process core sightings without understanding extensions;
- multiple organisations to publish independent enrichment;
- different extension schemas to evolve independently;
- corrections and retractions to be represented without silently changing historical objects.

1.13.2 Extension Fragment Fields

A BCP-12 extension fragment contains the following fields:

Field	Type	Required	Description
`uuid`	UUID	yes	Stable identifier of the extension fragment.
`sighting_uuid`	UUID	yes	UUID of the core sighting being extended.
`namespace`	URI	yes	Identifier of the extension schema or semantic namespace.
`version`	string	yes	Version of the extension namespace.
`creation_timestamp`	date-time	yes	Time when the extension fragment was created.
`relation`	enumeration	no	Relationship between the fragment and the core sighting or previous fragments.
`author`	string	no	Identifier of the fragment publisher or author.
`supersedes`	array of UUIDs	no	Previous extension fragments replaced by this fragment.
`content`	object	yes	Extension-specific structured content.

1.13.3 Extension Relations

The optional `relation` field may contain:

- `supplements`
- `corrects`
- `supersedes`
- `retracts`

1.13.3.1 `supplements`

The fragment adds information without replacing or invalidating the core sighting.

1.13.3.2 `corrects`

The fragment identifies and explains an error in the core sighting.

The original core object remains unchanged for provenance purposes.

1.13.3.3 `supersedes`

The fragment replaces one or more earlier extension fragments.

The `supersedes` field SHOULD contain the UUIDs of the replaced fragments.

1.13.3.4 `retracts`

The publisher withdraws the assertion represented by the core sighting or an earlier fragment.

A retraction SHOULD provide a human-readable reason in `content`.

A retraction does not delete the historical object. Consumers that support extension processing SHOULD display or otherwise account for the retracted state.

1.13.4 Extension Fragment Example

The following extension adds structured context to an `exploited` sighting:

```
{
  "uuid": "18fa47c7-2833-49d9-a496-85274427827c",
  "sighting_uuid": "4cb3de64-d5a8-4db6-b471-09a4277fda6f",
  "namespace": "https://gcve.eu/extensions/sighting-context",
  "version": "1.0",
  "creation_timestamp": "2026-07-30T06:45:00Z",
  "relation": "supplements",
  "author": "national-csirt",
  "content": {
    "observed_at": "2026-07-29T18:42:00Z",
    "confidence": 0.9,
    "evidence_type": [
      "incident-response",
      "forensic-analysis"
    ],
```

```
    "scope": {
      "asset_exposure": [
        "internet-facing"
      ],
      "sector": [
        "telecommunications"
      ]
    },
    "notes": "Scope describes the investigated environment and is not a
        global exploitation estimate."
  }
}
```

1.13.5 Extension Update Example

An extension fragment can supersede a previous fragment without changing the core sighting:

```
{
  "uuid": "fd6ff675-742a-41bc-a970-0cf13e68b376",
  "sighting_uuid": "4cb3de64-d5a8-4db6-b471-09a4277fda6f",
  "namespace": "https://gcve.eu/extensions/sighting-context",
  "version": "1.0",
  "creation_timestamp": "2026-07-30T10:12:00Z",
  "relation": "supersedes",
  "author": "national-csirt",
  "supersedes": [
    "18fa47c7-2833-49d9-a496-85274427827c"
  ],
  "content": {
    "observed_at": "2026-07-29T18:37:00Z",
    "confidence": 0.95,
    "evidence_type": [
      "incident-response",
      "forensic-analysis",
      "endpoint-telemetry"
    ],
    "scope": {
      "asset_exposure": [
        "internet-facing"
      ],
      "sector": [
        "telecommunications"
      ]
    },
    "notes": "The observation time and confidence were updated
       following additional forensic analysis."
  }
}
```

1.13.6 Retraction Example

```
{
  "uuid": "1104c317-9b83-4148-92c9-34f808d8280a",
  "sighting_uuid": "4cb3de64-d5a8-4db6-b471-09a4277fda6f",
  "namespace": "https://gcve.eu/extensions/sighting-lifecycle",
  "version": "1.0",
  "creation_timestamp": "2026-07-31T08:20:00Z",
  "relation": "retracts",
  "author": "national-csirt",
  "content": {
    "reason": "The observation was attributed to a different
       vulnerability after additional analysis."
  }
}
```

1.13.7 Extension Fragment JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "https://gcve.eu/schemas/bcp-12-sighting-extension-1.0.schema.
     json",
  "type": "object",
  "title": "GCVE-BCP-12 sighting extension fragment",
  "description": "A separate extension fragment associated with a BCP
     -12 core sighting.",
  "additionalProperties": false,
  "required": [
    "uuid",
    "sighting_uuid",
    "namespace",
    "version",
    "creation_timestamp",
    "content"
  ],
  "properties": {
    "uuid": {
      "type": "string",
      "format": "uuid",
      "description": "Stable UUID of the extension fragment."
    },
    "sighting_uuid": {
      "type": "string",
      "format": "uuid",
      "description": "UUID of the core sighting being extended."
    },
    "namespace": {
      "type": "string",
```

```
        "format": "uri",
        "description": "URI identifying the extension schema or semantic
            namespace."
      },
      "version": {
        "type": "string",
        "minLength": 1,
        "description": "Version of the extension namespace."
      },
      "creation_timestamp": {
        "type": "string",
        "format": "date-time",
        "description": "Timestamp when the extension fragment was created
            ."
      },
      "relation": {
        "type": "string",
        "enum": [
          "supplements",
          "corrects",
          "supersedes",
          "retracts"
        ],
        "default": "supplements",
        "description": "Relationship of the extension fragment to the
            core sighting or earlier extension fragments."
      },
      "author": {
        "type": "string",
        "minLength": 1,
        "description": "Identifier of the extension fragment author or
            publisher."
      },
      "supersedes": {
        "type": "array",
        "uniqueItems": true,
        "items": {
          "type": "string",
          "format": "uuid"
        },
        "description": "UUIDs of extension fragments superseded by this
            fragment."
      },
      "content": {
        "type": "object",
        "description": "Structured content defined by the extension
            namespace.",
        "additionalProperties": true
      }
    }
  }
```

1.14 Publishing Extensions

Extension fragments MAY be distributed:

- through a separate API endpoint;
- in a dedicated feed;
- as NDJSON;
- in an archive alongside core sightings;
- in a bundle containing separate `sightings` and `extensions` collections;
- through another transport agreed upon by producers and consumers.

Transport and discovery mechanisms are outside the mandatory core format.

When core sightings and extension fragments are transported together, they MUST remain logically separate.

For example:

```
{
  "sightings": [
    {
      "uuid": "4cb3de64-d5a8-4db6-b471-09a4277fda6f",
      "vulnerability_lookup_origin": "1a89b78e-f703-45f3-bb86-59
         eb712668bd",
      "author": "9f56dd64-161d-43a6-b9c3-555944290a09",
      "vulnerability": "CVE-2026-3323",
      "type": "exploited",
      "source": "incident-response-case-2026-184",
      "content": "Exploitation was observed during incident response.",
      "creation_timestamp": "2026-07-30T06:40:00Z"
    }
  ],
  "extensions": [
    {
      "uuid": "18fa47c7-2833-49d9-a496-85274427827c",
      "sighting_uuid": "4cb3de64-d5a8-4db6-b471-09a4277fda6f",
      "namespace": "https://gcve.eu/extensions/sighting-context",
      "version": "1.0",
      "creation_timestamp": "2026-07-30T06:45:00Z",
      "relation": "supplements",
      "content": {
        "observed_at": "2026-07-29T18:42:00Z",
        "confidence": 0.9
      }
    }
  ]
}
```

The transport wrapper shown above is illustrative and is not itself a replacement for the core sighting format.

An existing Vulnerability-Lookup implementation can import the item from `sightings` and ignore the `extensions` collection.

1.15 Corrections and Lifecycle

1.15.1 Immutability

A published core sighting SHOULD NOT be modified in place.

The UUID identifies the original assertion and its historical content.

Changing the meaning of an existing UUID can lead to inconsistent replicas, unverifiable historical analyses, and conflicting cached data.

1.15.2 Correcting a Sighting

When a sighting contains an error, the producer SHOULD use one of the following approaches:

1. publish a correction extension fragment;
2. retract the original sighting and create a new corrected sighting with a new UUID;
3. delete the local object when necessary and publish a retraction through a distribution mechanism capable of propagating lifecycle information.

The same UUID MUST NOT be reused for the corrected sighting.

1.15.3 Deletion

Deletion is a local storage operation and does not automatically communicate a retraction to other systems.

Once a sighting has been distributed, producers SHOULD publish a retraction fragment when consumers need to know that the assertion was withdrawn.

1.15.4 Conflicting Sightings

Conflicting sightings SHOULD be preserved as separate assertions.

Consumers SHOULD NOT automatically delete or merge sightings solely because their types appear contradictory.

For example, these sightings may all be valid:

- one organisation reports `exploited`;
- another reports `not-exploited`;
- a vendor reports `patched`;
- a researcher publishes `published-proof-of-concept`.

Their scopes, timestamps, and evidence may differ.

1.16 Compatibility Profiles

1.16.1 BCP-12 Core Consumer

A BCP-12 core consumer:

- MUST accept valid core sighting objects;
- MUST reject or safely isolate unknown core properties;
- MAY ignore extension fragments;
- MUST NOT require extension support.

1.16.2 BCP-12 Extended Consumer

A BCP-12 extended consumer:

- MUST support the core sighting format;
- MAY process one or more recognised extension namespaces;
- MUST ignore unsupported extension namespaces without rejecting the associated core sighting;
- MUST preserve extension fragments it republishes unless local policy requires otherwise;
- SHOULD retain extension provenance and lifecycle relationships.

1.16.3 Vulnerability-Lookup Compatibility Profile

A Vulnerability-Lookup-compatible producer:

- MUST use the existing core field names;

- MUST use one of the eight existing sighting types;
- MUST NOT add properties to the core object;
- SHOULD allow the server to assign read-only fields when submitting through the Vulnerability-Lookup API;
- MUST publish structured extensions separately;
- SHOULD preserve the original UUID and origin when redistributing sightings.

A core sighting complying with this BCP can therefore be stored and processed by the existing Vulnerability-Lookup implementation without changes to its sighting database model or core JSON schema.

1.17 Format Evolution

1.17.1 Core Version 1

The format defined in this document is the BCP-12 core version 1 format.

No explicit version field is included in the core object because adding such a field would break compatibility with existing strict validators.

The core version is identified through:

- the BCP version;
- the schema URI;
- the API or feed documentation;
- an external transport media type or profile.

1.17.2 Adding New Information

New information SHOULD first be introduced through an extension namespace.

Examples include:

- observation timestamps;
- confidence;
- evidence categories;
- detection methods;
- geographic scope;
- affected sectors;
- asset exposure;

- patch versions;
- remediation verification;
- TLP markings;
- signatures;
- licence information;
- relationships to incidents, cases, KEV assertions, or other sightings.

1.17.3 Adding New Sighting Types

New values MUST NOT be added to the BCP-12 version 1 core enumeration.

A producer requiring a more specific type SHOULD:

1. select the closest valid core type;
2. publish the more specific classification through an extension fragment.

For example, a producer observing mass scanning without confirmed compromise may use:

```
{
  "type": "seen"
}
```

and publish an extension containing:

```
{
  "classification": "mass-scanning"
}
```

A future major revision of BCP-12 MAY define additional core types, but it MUST provide a documented mapping to the version 1 enumeration when backward compatibility is required.

1.17.4 Future Major Revisions

A future incompatible core format MUST use a new schema URI.

It SHOULD provide:

- an explicit compatibility profile;
- conversion rules to BCP-12 core version 1;
- guidance for existing Vulnerability-Lookup implementations;
- handling rules for unknown or unmappable fields and types.

1.18 Validation and Processing Recommendations

Producers SHOULD validate sightings before publication.

Consumers SHOULD:

- validate UUID and timestamp syntax;
- enforce the `source` length limit;
- validate the sighting type enumeration;
- treat `source` and `content` as untrusted input;
- retain the original UUID and origin;
- avoid silently modifying imported objects;
- preserve timestamps as supplied;
- avoid inferring global conclusions from context-limited sightings;
- record import or processing timestamps separately from `creation_timestamp`.

Implementations MAY apply additional identifier validation based on the vulnerability namespaces they support.

Such local validation restrictions SHOULD be documented.

1.19 Duplicate Detection

BCP-12 does not define two sightings as duplicates solely because they have the same:

- vulnerability;
- source;
- type;
- author;
- day;
- content.

Different UUIDs identify different assertions.

Implementations MAY use duplicate-detection rules to reduce repeated automated observations. Such rules are implementation-specific and SHOULD be documented.

Deduplication SHOULD NOT silently combine sightings from different origins or authors when preserving independent provenance is important.

1.20 Security Considerations

1.20.1 Untrusted Content

The `source`, `content`, vulnerability identifier, and all extension content MUST be treated as untrusted input.

Consumers MUST escape or sanitise these values before inserting them into:

- HTML;
- Markdown rendered as HTML;
- SVG;
- JavaScript;
- SQL queries;
- shell commands;
- log formats;
- templates.

1.20.2 Source Retrieval

Consumers MUST NOT automatically fetch a `source` value merely because it resembles a URI.

Implementations that retrieve sources SHOULD:

- use a URL-scheme allowlist;
- prevent access to local and restricted network resources;
- apply timeouts and size limits;
- prevent unintended redirects;
- avoid executing downloaded content;
- record retrieval provenance separately.

1.20.3 Sensitive Information

Sightings may contain operational, incident-response, or victim information.

Producers SHOULD minimise sensitive information in public sightings and extensions.

Information such as the following SHOULD NOT be included unless appropriate for the intended sharing community:

- personal data;

- customer names;
- internal hostnames;
- IP addresses associated with victims;
- credentials;
- forensic artefacts exposing confidential data;
- active incident identifiers;
- information that could facilitate additional compromise.

Traffic Light Protocol markings or equivalent sharing restrictions SHOULD be used when sightings are distributed in restricted communities.

1.20.4 Negative Assertions

Negative types such as `not-exploited`, `not-confirmed`, and `not-patched` can be misleading when their scope is omitted.

Producers SHOULD provide sufficient scope and methodology to prevent a local negative observation from being interpreted as a universal conclusion.

1.20.5 Author Privacy

Author UUIDs may constitute pseudonymous personal data.

Producers and consumers SHOULD apply applicable privacy and retention requirements.

1.20.6 Trust and Provenance

Consumers SHOULD evaluate sightings using:

- origin;
- author;
- source;
- timestamp;
- evidence;
- confidence;
- scope;
- publication history.

No sighting type should be accepted as authoritative solely because it is syntactically valid.

1.21 Privacy Considerations

Sightings are intended to support vulnerability analysis and operational coordination, but they may indirectly reveal:

- that an organisation uses an affected product;
- that an organisation experienced exploitation;
- that patching has or has not occurred;
- the existence of an active investigation;
- the capabilities or visibility of a monitoring system.

Producers SHOULD assess these risks before publication.

Aggregated or generalised scope information SHOULD be used when detailed information is unnecessary.

1.22 Implementation Guidance

1.22.1 Producers

Producers SHOULD:

- create one core object per independent observation;
- use stable UUIDs;
- preserve the original origin;
- select the most appropriate core sighting type;
- provide a meaningful source when possible;
- avoid using `content` as a structured data container;
- publish detailed context through extension fragments;
- issue corrections or retractions without mutating historical objects.

1.22.2 Consumers

Consumers SHOULD:

- retain provenance;
- permit multiple sightings for the same vulnerability;
- display negative assertions with their scope;
- distinguish sighting time from observed event time;

- preserve unsupported extension fragments when practical;
- clearly indicate corrected, superseded, or retracted sightings;
- avoid converting individual sightings directly into global KEV status without additional assessment.

1.22.3 Automation Tools

Automation tools SHOULD:

- use a dedicated author identity;
- identify the upstream source;
- avoid creating excessive duplicate sightings;
- retain the original publication or observation reference;
- distinguish collection time from observation time;
- document their mapping logic;
- avoid converting weak signals into strong sighting types without sufficient evidence.

For example, a social-network mention would normally produce `seen`, not `confirmed` or `exploited`.

1.23 Examples

1.23.1 Social-Network Mention

```
{
  "uuid": "095641dc-7260-4673-8291-02ecf328ea97",
  "vulnerability_lookup_origin": "1a89b78e-f703-45f3-bb86-59eb712668bd",
  "author": "9f56dd64-161d-43a6-b9c3-555944290a09",
  "vulnerability": "CVE-2026-3323",
  "type": "seen",
  "source": "https://social.example/@security/114839220001",
  "content": "A security advisory referencing CVE-2026-3323 was shared.",
  "creation_timestamp": "2026-07-30T06:20:00Z"
}
```

1.23.2 Published Proof of Concept

```
{
  "uuid": "08b320a0-52fe-4c16-b509-0690dadc633e",
  "vulnerability_lookup_origin": "1a89b78e-f703-45f3-bb86-59eb712668bd"
     ,
  "author": "9f56dd64-161d-43a6-b9c3-555944290a09",
  "vulnerability": "GCVE-1-2026-0042",
  "type": "published-proof-of-concept",
  "source": "https://code.example/security-research/example-poc",
  "content": "A public proof of concept reproducing the issue was
     published.",
  "creation_timestamp": "2026-07-30T06:30:00Z"
}
```

1.23.3 Observed Exploitation

```
{
  "uuid": "4cb3de64-d5a8-4db6-b471-09a4277fda6f",
  "vulnerability_lookup_origin": "1a89b78e-f703-45f3-bb86-59eb712668bd"
     ,
  "author": "9f56dd64-161d-43a6-b9c3-555944290a09",
  "vulnerability": "CVE-2026-3323",
  "type": "exploited",
  "source": "incident-response-case-2026-184",
  "content": "Exploitation was observed during incident response.",
  "creation_timestamp": "2026-07-30T06:40:00Z"
}
```

1.23.4 Scoped Negative Observation

```
{
  "uuid": "b70c2727-6243-4f22-96a7-26a3c3346088",
  "vulnerability_lookup_origin": "1a89b78e-f703-45f3-bb86-59eb712668bd"
     ,
  "author": "9f56dd64-161d-43a6-b9c3-555944290a09",
  "vulnerability": "CVE-2026-3323",
  "type": "not-exploited",
  "source": "internal-investigation-2026-415",
  "content": "No evidence of exploitation was found in the examined
     environment.",
  "creation_timestamp": "2026-07-30T07:10:00Z"
}
```

The scope, period, and investigation method SHOULD be provided through an extension fragment when they can be shared.

1.24 References

- GCVE-BCP-12 public discussion
- Vulnerability-Lookup Sightings User Manual
- Vulnerability-Lookup Sighting JSON Schema
- Vulnerability-Lookup Sighting Model
- Vulnerability-Lookup Sighting API
- GCVE-BCP-07 - Known Exploited Vulnerability Assertion Format
- MISP Vulnerability Taxonomy
- RFC 2119 - Key words for use in RFCs to Indicate Requirement Levels
- RFC 8174 - Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words
- RFC 3339 - Date and Time on the Internet: Timestamps
- ECMA-404 - The JSON Data Interchange Syntax

1.25 Acknowledgements

1.25.1 BCP-12 Coordinators

- Alexandre Dulaunoy, CIRCL
- Cédric Bonhomme, CIRCL

1.25.2 Contributions

The GCVE initiative gratefully acknowledges contributions and feedback submitted through the public BCP-12 review process.