Independent Submission
Internet-Draft
Intended status: Informational
Expires: 10 November 2026

CD. Langton
Vulnetix

9 May 2026

# Cloud Resource Identifier Templates (CRIT)
draft-vulnetix-crit-02

## Abstract

This document specifies the Cloud Resource Identifier Templates (CRIT) format. A CRIT record provides a machine-readable, parameterised template for locating cloud-native resources affected by a known vulnerability. CRITs do not define cloud resource identifier schemas; those are defined normatively by each cloud provider. CRITs define a variable system for expressing partially-known or consumer-resolved values within those provider-defined schemas, together with temporal, remediation, and detection metadata sufficient to determine exposure status and drive remediation workflows.

Each CRIT record is bound to exactly one vulnerability identifier. Cross-provider and multi-resource-type coverage of a single vulnerability is expressed as a set of CRIT records sharing the same vulnerability identifier, each independently specifying the provider-specific fix details, propagation mechanism, and detection strategy applicable to that resource type.

## 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 10 November 2026.

Table of Contents

## 1. Introduction

This document specifies the Cloud Resource Identifier Templates (CRIT) format, a machine-readable schema for describing cloud infrastructure resources affected by known vulnerabilities. CRIT provides parameterised templates over provider-native identifier schemas, together with fix propagation semantics, exposure window computation rules, and detection metadata sufficient to drive automated remediation workflows.

### 1.1. Overview

CPE [CPE23] and PURL [PURL] model the vulnerable entity as a build-from-source artifact — something with a static name, a version string, and a build-time identity that persists across deployment. Cloud infrastructure resources do not have these properties. An RDS instance, an EKS cluster, and a Cloudflare Worker are each identified by provider-native runtime identifiers whose components include consumer-specific variables (account identifiers, region codes, resource IDs) that do not exist until the resource is deployed. No

package name, version string, or source repository URL applies.

CRIT defines a parameterised template system over these provider-native identifier schemas, together with fix propagation semantics, exposure window computation rules, and detection metadata. It integrates with CVEListv5 ([CVEListv5]) ADP containers and OSV schema ([OSV-Schema]) using their existing extension mechanisms. Risk-based prioritisation signals such as EPSS ([EPSS]) remain complementary inputs to consumer tooling.

## 1.2. The Identifier Gap

CPE and PURL both assume the vulnerable entity is produced by a build process — compiled from source, packaged into a distributable artifact, and deployed by installing that artifact. This assumption holds for operating systems, libraries, and application binaries. It does not hold for cloud infrastructure resources.

A cloud resource is instantiated by a provider API call, not by installing a package. It is identified by a provider-native runtime identifier — an ARN, an Azure Resource ID, a GCP Resource Name, an OCID, or a Cloudflare Locator — that is assigned at creation time and contains components specific to the consumer’s account, region, and deployment. These identifiers have no analogue in any package registry. There is no source repository, no version string, and no build manifest.

Representing a cloud resource as a pkg:generic/ PURL, a synthesised CPE string, or a custom PURL type does not resolve this gap. The PURL specification [PURL] defines no registered type for cloud infrastructure resources. The pkg:cloud/ convention observed in the OSV ecosystem (see Section 9.3.1) is not a registered PURL type. Regardless of identifier scheme, the resulting string carries none of the information required to determine whether a specific deployed resource is affected: the deployment date relative to the fix, the propagation mechanism, whether the consumer has taken the required action, or whether a configuration change has since been reverted.

## 1.3. Cloud Resource Exposure Model

For package vulnerabilities, affected status is determined by a version comparison: if the installed version falls within the affected range, the package is vulnerable. Cloud resources have no equivalent comparison. Affected status is a function of four factors that must be evaluated simultaneously:

* When the resource was deployed relative to when the provider fix became available.

* The fix propagation type — whether the fix applies automatically, requires a version update, requires a configuration change, or requires the resource to be destroyed and recreated.

* Whether the consumer has taken the required action, if any.

* Whether a previously applied remediation has been reverted by subsequent configuration drift.

No static identifier carries these factors. A CPE or PURL string identifies what the resource is; it does not encode how the fix reaches the resource or whether a specific instance has been remediated. Each consumer tool that evaluates cloud resource exposure must independently model these semantics.

Discovery additionally requires interpolation. The identifier for a specific resource instance contains consumer-specific variables — account, region, resource ID — that must be substituted at resolution time. A single CRIT template represents all instances of a resource type; resolution produces the concrete identifier for a specific instance.

### 1.4. CRIT Approach

CRIT addresses the identifier gap by defining a parameterisation layer over provider-native identifier schemas. A CRIT record does not invent a new identifier format. It parameterises the identifier format the provider already defines, expressing consumer-specific and context-dependent values as variable slots within the provider’s own schema.

Each CRIT record carries the fix propagation type, shared responsibility model, temporal metadata, remediation actions, and detection queries required for a consumer to evaluate exposure and drive remediation for a specific vulnerability on a specific cloud resource type. The record is bound to exactly one vulnerability identifier. Cross-provider and multi-resource-type coverage of a single vulnerability is expressed as a set of records sharing the same vulnerability identifier, each independently specifying the provider-specific semantics.

CRIT does not replace CVE, CPE, or PURL. It complements them by providing the cloud resource scope, fix propagation semantics, and exposure window computation that those schemes do not address.

### 1.5. What CRIT Is and What It Is Not

If the problem could be described in one word, that word is *affected*. For packages, "affected" is a version comparison. For cloud, "affected" is a function of four factors evaluated simultaneously: when the resource was deployed relative to the fix, the fix propagation type, whether the consumer has acted, and whether a previously applied remediation has been reverted by configuration drift. No static identifier carries these factors. CRIT encodes all four.

#### 1.5.1. CRIT Is

* *A template engine for cloud-native resources.* Discovery requires interpolation of consumer-specific variables (account, region, resource ID) at resolution time. No static identifier can express this.

* *A solution to the "affected" problem.* It encodes deployment timing, fix propagation type, consumer action state, and configuration drift status — everything required to determine whether a specific deployed resource is impacted.

* *A parameterisation layer over provider-native identifier schemas.* CRITs do not invent identifier formats. AWS ARNs, Azure Resource IDs, GCP Resource Names, Cloudflare Locators, and Oracle OCIDs are adopted as-is. CRIT parameterises them with variable slots.

* *An extension to existing vulnerability data formats.* CRIT integrates with CVEListv5 ADP containers and OSV schema using their existing extension mechanisms. It does not require changes to those specifications.

* *A machine-readable encoding of fix propagation, remediation actions, detection queries, and exposure window computation* — the metadata that turns a vulnerability advisory into an actionable remediation workflow for cloud resources.

#### 1.5.2. CRIT Is Not

* *Not an identifier.* Cloud resources already have identifiers. CRITs reference them; they do not define new ones. The CRIT vectorString is a natural composite key (replacing the UUID), not a resource identifier.

* *Not a replacement for CPE, PURL, CycloneDX, or SPDX.* Those standards solve identification, inventory, and risk prioritisation for build-from-source artifacts. CRIT complements them by addressing cloud resource scope where they do not apply.

* *Not a single string that can encode the full record.* The CRIT vectorString is a lossy compact encoding of 12 enumerable fields from a 30+ field record. Descriptive values, detection queries, remediation action descriptions, provider-native templates, and consumer-specific variables cannot be represented in any static string. Any attempt to reduce CRIT to a single-string identifier discards the metadata that solves the problem.

* *Not a risk scoring or prioritisation system.* CRIT does not assign severity, CVSS scores, or risk rankings. Risk-based prioritisation signals (EPSS, CVSS, SSVC) remain complementary inputs to consumer tooling.

* *Not a replacement for cloud provider security advisories.* CRIT references provider advisories; it does not replace them. Provider advisory URLs and identifiers are carried as metadata within the record.

* *Not a software inventory or bill-of-materials format.* CRIT records describe how vulnerabilities affect cloud resources. They are not an inventory of deployed resources or a software composition.

#### 1.5.3. Why Not PURL?

PURL [PURL] succeeds because package identity is static: pkg:npm/@angular/core@12.3.1 is the same string regardless of where the package is installed. Cloud resource identity is not static. An RDS instance's ARN contains an account ID, region, and resource ID that do not exist until the resource is deployed. Even if a PURL were constructed at that granularity, it would carry none of the information required to determine affected status: the deployment date relative to the fix, the propagation mechanism, whether the consumer has acted, or whether a configuration change has been reverted.

The pkg:cloud/ convention observed in the OSV ecosystem (see Section 9.3.1) is not a registered PURL type. Regardless of the type scheme, a static string cannot express the interpolation that discovery requires or the temporal and propagation logic that affected-status determination demands.

## 1.6. Conventions and Terminology

### 1.6.1. 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.

### 1.6.2. Terminology

The following terms are used throughout this document.

CRIT Record:
: A JSON object conforming to this specification that describes the scope, remediation, and detection characteristics of a single vulnerability as it applies to a single cloud resource type.

CRIT Template:
: A parameterised string in which variable slots represent consumer-specific or context-dependent values. After all slots are resolved, the result is a valid provider-native resource identifier.

CRIT Dictionary:
: A machine-readable catalogue enumerating valid (provider, service, resource_type) tuples with their corresponding templates and metadata. See Section 12.

CRIT Vector String:
: A compact, deterministic, human-readable encoding of a CRIT record’s classification and identity fields, modelled on CVSS vector strings. See Section 4.1.2.

Natural Key:
: The tuple (vuln_id, provider, service, resource_type) that uniquely identifies a CRIT record within a conformant corpus.

Producer:
: An entity that creates and emits CRIT records.

Consumer:
: An entity that processes CRIT records to evaluate cloud resource exposure, deploy detections, or drive remediation workflows.

Conformant Corpus:
A collection of CRIT records in which no two records share the same natural key.

Provider:
A cloud service vendor whose resources are covered by this specification. Supported values are: aws, azure, gcp, cloudflare, oracle, salesforce, sap, and servicenow. Each provider maintains its own resource identification scheme; CRIT encodes those schemes as templates without altering or extending them.

Service:
A distinct offering within a provider's portfolio, identified by a lowercase underscore-delimited key (e.g., ec2, kubernetes_engine, waf).

Resource Type:
A specific kind of resource within a service, identified by a key matching the provider's API conventions (e.g., instance, cluster, waf_ruleset).

Template Format:
The provider-native identifier schema used by a template: aws_arn, azure_resource_id, gcp_resource_name, cloudflare_locator, or oracle_ocid.

Variable Slot:
A delimited placeholder within a CRIT template, enclosed in braces, that represents a value to be supplied or fixed.

Named Variable:
A slot whose value the consumer must supply at resolution time. Syntax: {field-name}.

Wildcard:
A slot representing any value, used for inventory enumeration. Syntax: {field-name=*}.

Hardcoded:
A slot with a fixed value determined by the provider schema. Syntax: {field-name=value}.

Empty:
A slot for a field that is structurally present but semantically inapplicable to the resource type. Syntax: {field-name=}.

Slot Resolution:
: The process of substituting concrete values into variable slots to produce a live provider identifier.

Resource Lifecycle:
: A classification of a resource type's operational behaviour with respect to data durability and replacement. See Section 4.4.1.

Shared Responsibility:
: The remediation responsibility model describing whether the provider, the consumer, or both must act to remediate a vulnerability. See Section 4.4.2.

VEX Status:
: The Vulnerability Exploitability eXchange status of a record: affected, fixed, not_affected, or under_investigation. Aligns with the OpenVEX ([OpenVEX]) and CSAF VEX ([CSAF-VEX]) vocabularies. See Section 4.8.

Fix Propagation:
: The mechanism by which a provider-released fix reaches existing deployed resources. See Section 4.4.3.

Exposure Window:
: The time interval [W_start, W_end] during which a specific resource instance is vulnerable. See Section 7.

Existing Deployments Remain Vulnerable:
: A boolean indicating whether resources deployed before the provider fix date remain in the exposure window absent explicit consumer action.

Remediation Action:
: An ordered step a consumer takes to remediate or mitigate a vulnerability on a specific resource type. See Section 4.4.4.

Compensating Control:
: A remediation action that reduces exploitability but does not fully remediate the vulnerability.

Detection Entry:
: A log query, metric filter, or alerting rule for identifying vulnerable configurations, active exploitation, or configuration drift. See Section 4.6.

Detection Phase:
: The lifecycle stage of a detection: pre_fix, exploitation, post_fix, or misconfiguration. See Section 4.6.3.

Pending Reason:
An enumerated value indicating why a detection entry is a placeholder without a functional query. See Section 4.6.4.

Vulnerability Identifier:
A string uniquely identifying the vulnerability a CRIT record relates to (e.g., CVE-2024-6387).

Provider Fix Date:
The date a provider made a fix generally available.

Vulnerability Published Date:
The date the vulnerability was publicly disclosed.

Service Available Date:
The date a provider's service became generally available, bounding the earliest possible deployment of affected resources.

## 2. Scope and Relationship to Provider Schemas

CRITs operate as a parameterisation layer over externally-defined resource identifier schemas. The authoritative definition of each identifier format is owned by its respective provider:

| Provider | Identifier Type | Normative Reference |
|---|---|---|
| AWS | Amazon Resource Name (ARN) | [AWS-ARN] |
| Azure | Azure Resource ID | [Azure-ResourceID] |
| GCP | GCP Resource Name | [GCP-ResourceName] |
| Cloudflare | Cloudflare API Locator | [CF-API] |
| Oracle | Oracle Cloud ID (OCID) | [OCI-OCID] |
| Salesforce | REST API Resource URL | [SF-REST-API] |
| SAP BTP | BTP / OData / SuccessFactors URL | [SAP-BTP] |
| ServiceNow | Table API URL | [SN-TABLE-API] |

Table 1

This specification does not alter, extend, or redefine any provider identifier schema. A conformant CRIT template MUST produce a string that, after variable resolution, is a valid identifier according to the applicable provider schema.

A CRIT template MUST NOT use pkg:generic/ or any PURL type that implies a build-from-source artifact to represent a cloud infrastructure resource. Such usage introduces ambiguous semantics in tooling designed around the build-artifact assumption and is explicitly out of scope for this specification.

This specification does not cover:

* How CRIT records are extracted from vulnerability data sources.

* Internal database schemas or persistence mechanisms.

* Vulnerability ingestion pipelines or NLP-based service detection.

* Source corpus processing.

## 3. The CRIT Variable System

### 3.1. Overview

A CRIT record is a template engine for cloud-native resources: discovery requires interpolation of consumer-specific variables at resolution time, which no static identifier can express.

A CRIT template string is a provider identifier format with zero or more variable slots. Each slot expresses one of four states. The choice of state is normative: it is determined by the semantics of the field for the given resource type, not by what the consumer happens to know.

### 3.2. Slot Syntax

Variable slots are delimited with { and }. The content within the delimiters is a slot descriptor with the following ABNF ([RFC5234]) grammar:

```
slot            = "{" slot-descriptor "}"
slot-descriptor = named-var / wildcard / empty-marker / hardcoded
named-var       = field-name
wildcard        = field-name "=" "*"
empty-marker    = field-name "="
hardcoded       = field-name "=" literal-value
field-name      = 1*( ALPHA / DIGIT / "-" / "_" )
literal-value   = 1*( ALPHA / DIGIT / "-" / "_" / "." / ":" )
```

Figure 1: Slot ABNF Grammar

The characters { and } are reserved as slot delimiters and MUST NOT appear in literal-value or as literal characters within a template string outside of slot expressions.

### 3.3. Design Relationship to RFC 6570

CRIT slot syntax uses curly-brace delimiters that resemble [RFC6570] URI Templates. This resemblance is intentional but superficial. CRIT slots are *not* URI Templates and implementations MUST NOT process them with an RFC 6570 expansion engine.

The CRIT variable system defines exactly four slot states. The number four is a design invariant: each additional state multiplies the number of producer-consumer interactions that require specification and testing. Because cloud provider identifier schemas are not standardised, and the number of providers is unbounded, the slot state vocabulary must remain fixed to prevent combinatorial growth in conformance surface area. The four states are the minimum set that satisfies all operational requirements defined in this specification:

1. *Named variable* ({field}): Consumer-supplied value. This is the only state that corresponds to [RFC6570] Level 1 simple string expansion.

2. *Wildcard* ({field=*}): Population enumeration. No [RFC6570] equivalent exists. The semantic is an inventory operation, not a URI construction.

3. *Empty* ({field=}): Structurally present but inapplicable. The resolved value is the empty string. [RFC6570] does not distinguish between an undefined variable and a field that is normatively absent from the provider schema; CRIT requires this distinction to prevent consumers from treating a structurally absent field as an unknown value requiring resolution.

4. *Hardcoded* ({field=literal}): Provider-fixed value. The value is determined by the provider schema, not by the consumer. This state is required because some provider identifier schemas contain positional fields whose value is invariant for a given resource type (e.g., a region field that is always us-east-1 for a global service). Without a distinct hardcoded state, a consumer cannot distinguish a provider-fixed value from a consumer-specific value that coincidentally matches.

[RFC6570] operators (+, #, ., /, ;, ?, &) are not supported. CRIT templates use simple string replacement only. A fully-resolved CRIT template — one in which every named-variable slot has been substituted with a concrete value, every empty slot resolved to the empty string, and every hardcoded slot resolved to its literal — produces a valid provider-native resource identifier. The result is a plain string, not a URI Template expansion.

## 3.4. The Four Slot States

### 3.4.1. Named Variable

Syntax: {field-name}

The slot represents a value the consumer MUST supply at resolution time. A consumer MUST NOT treat a named variable as implying any default value. A consumer MUST substitute a concrete value before using the template as a live identifier.

Examples: {region}, {account}, {resource-id}.

### 3.4.2. Wildcard

Syntax: {field-name=*}

The slot represents "any value" and is used for inventory matching across a population of resources. A wildcard MUST NOT be used as a live identifier against a provider API; it is a query pattern only. A consumer MAY expand a wildcard by enumerating known values from their inventory. A consumer MUST record when a wildcard remains unexpanded, as an unexpanded wildcard indicates incomplete inventory coverage.

Examples: {region=*} matches all regions; {account=*} matches all accounts.

#### 3.4.3. Empty

Syntax: {field-name=}

The slot represents a field that is structurally present in the provider schema but not applicable for this resource type. The resolved value is the empty string. This MUST NOT be confused with an unknown value (use named variable) or a match-all (use wildcard). It is a precise semantic statement that the field does not apply to this resource type.

Example: GCP global resources carry no zone; the zone slot is expressed as {zone=}.

#### 3.4.4. Hardcoded

Syntax: {field-name=literal-value}

The slot represents a fixed value determined by the provider schema for this resource type. A CRIT producer MUST use hardcoded state only for values normatively fixed by the provider schema. A consumer MUST use the hardcoded value as-is and MUST NOT substitute an alternative value.

Example: {region=us-east-1} for AWS IAM resources, which the AWS ARN schema requires to always be us-east-1.

### 3.5. Slot State Selection Rules

A CRIT producer MUST select the slot state according to the following precedence:

1. If the provider schema normatively fixes this field to a specific value for the resource type: *hardcoded*.

2. If the provider schema specifies the field is structurally absent or inapplicable for this resource type: *empty*.

3. If the field represents a cross-resource population query rather than a specific resource: *wildcard*.

4. Otherwise: *named variable*.

A CRIT producer MUST NOT use wildcard as a fallback when the correct state is unknown. An unknown consumer-specific value is always a named variable; wildcard is a deliberate semantic choice meaning "enumerate all".

## 4. CRIT Record Schema

All field names are lowercase snake_case. The schema is expressed in JSON. Unless stated otherwise, absent optional fields are interpreted as null. All date values MUST be expressed in ISO 8601 [ISO8601] full-date format (YYYY-MM-DD) in UTC. Time-of-day components SHOULD be omitted unless a provider advisory specifies intraday precision is meaningful.

### 4.1. Envelope and Identity

```
{
  "vectorString": "<crit-vector>",
  "vuln_id": "<string>",
  "provider": "<enum>",
  "service": "<string>",
  "resource_type": "<string>",
  "resource_lifecycle": "<enum>",
  "shared_responsibility": "<enum>",
  "vex_status": "<enum>"
}
```

Figure 2: Envelope Fields

| Field | Required | Type | Description |
|---|---|---|---|
| vectorString | REQUIRED | string | Canonical CRIT vector string computed from record fields. See Section 4.1.2. |
| vuln_id | REQUIRED | string | The vulnerability this record relates to. MUST match exactly one vulnerability per record. |
| provider | REQUIRED | enum | One of: aws, azure, gcp, cloudflare, oracle, salesforce, sap, servicenow. |
| service | REQUIRED | string | Provider service key (e.g., lambda, aks, cloud_sql). |
| resource_type | REQUIRED | string | Specific resource type within the service (e.g., function, cluster, instance). |
| resource_lifecycle | REQUIRED | enum | See Section 4.4.1. |
| shared_responsibility | REQUIRED | enum | See Section 4.4.2. |
| vex_status | REQUIRED | enum | See Section 4.8. |

Table 2

#### 4.1.1. Natural Key and Uniqueness

The tuple (vuln_id, provider, service, resource_type) constitutes the natural key of a CRIT record. Within a conformant corpus, no two records MAY share the same natural key. A producer MUST enforce this uniqueness constraint before emitting records.

When a single vulnerability affects multiple resource types within the same service, or the same resource type across multiple providers, the correct representation is multiple CRIT records each with a distinct natural key and independently specified fix version, propagation mechanism, and detection entries. The vuln_id field is the join key allowing a consumer to retrieve the complete set of records for a given vulnerability.

Example: a Kubernetes vulnerability affecting EKS, AKS, and GKE yields three records:

* (CVE-2024-XXXX, aws, eks, cluster)

* (CVE-2024-XXXX, azure, aks, cluster)

* (CVE-2024-XXXX, gcp, gke, cluster)

The natural key components are embedded in the CRIT vector string: provider as the CP metric and vuln_id, service, resource_type as the three positional qualifiers. The vectorString is therefore a canonical single-string encoding of the record's natural key combined with its classification state.

#### 4.1.2. CRIT Vector String

The CRIT vector string is a compact, deterministic, human-readable encoding of a record's classification and identity fields. Its format is modelled on CVSS vector strings: a versioned prefix followed by slash-delimited metric-value pairs and a qualifier section.

The ABNF ([RFC5234]) grammar is:

```
crit-vector     = prefix "/" metrics "#" qualifiers
prefix          = "CRITv" semver
semver          = 1*DIGIT "." 1*DIGIT "." 1*DIGIT
                  [ "-" 1*(ALPHA / DIGIT / ".") ]
metrics         = metric *("/" metric)
metric          = metric-key ":" metric-value
metric-key      = 2ALPHA
metric-value    = 1*(ALPHA / DIGIT)
qualifiers      = qual-value ":" qual-value ":" qual-value
qual-value      = 1*(ALPHA / DIGIT / "-" / "_" / ".")
```

Example:

```
CRITv0.3.0/CP:AW/VS:FX/FP:RR/SR:CA/RL:SC/EV:T/PP:1719792000/SA:1514764800#CVE-2024-638
7:ec2:instance
```

#### 4.1.2.1. Registered Metrics

A conformant CRIT vector string MUST include all registered metrics listed below. Registered metrics MUST appear in the canonical order defined by this section. A producer MAY append additional metrics after the registered set and before the # delimiter; a consumer MUST ignore unknown metric keys without error.

*Table 1: Cloud Provider (CP)*

| Value | Code | Description |
|---|---|---|
| aws | AW | Amazon Web Services |
| azure | MA | Microsoft Azure |
| gcp | GC | Google Cloud Platform |
| cloudflare | CF | Cloudflare |
| oracle | OC | Oracle Cloud Infrastructure |
| salesforce | SF | Salesforce |
| sap | SP | SAP |
| servicenow | SN | ServiceNow |
| ibm | IB | IBM Cloud |
| vmware | VM | VMware (Broadcom) |
| adobe | AD | Adobe |
| akamai | AK | Akamai |
| alibaba | AL | Alibaba Cloud |
| atlassian | AT | Atlassian |
| digitalocean | DO | DigitalOcean |
| elastic | EL | Elastic |
| fastly | FA | Fastly |
| gitlab | GL | GitLab |

| | | |
|---|---|---|
| hashicorp | HC | HashiCorp |
| hetzner | HE | Hetzner Cloud |
| linode | LI | Linode (Akamai Connected Cloud) |
| mongodb | MO | MongoDB Atlas |
| ovh | OV | OVHcloud |
| snowflake | SO | Snowflake |
| tailscale | TS | Tailscale |
| tencent | TC | Tencent Cloud |
| twilio | TW | Twilio |
| vercel | VC | Vercel |
| vultr | VL | Vultr |
| zoom | ZM | Zoom |

Table 3

Note: the codes CF and GC appear in both Table 1 (Cloud Provider) and Table 5 (Resource Lifecycle). The grammar disambiguates by metric key prefix: CP:CF identifies Cloudflare while RL:CF identifies a configuration-only resource lifecycle. Producers and consumers MUST NOT rely on raw two-letter codes outside of their metric-key context.

*Table 2: VEX Status (VS)*

| Value | Code | Description |
|---|---|---|
| affected | AF | Resource type is affected; no fix available or not applied. |
| fixed | FX | Provider fix is available; provider_fix_date is set. |
| not_affected | NA | Resource type is not affected or vulnerability is not reachable. |
| under_investigation | UI | Provider has acknowledged but not confirmed status. |

Table 4

*Table 3: Fix Propagation (FP)*

| Value | Code | Description |
|---|---|---|
| automatic | AU | Provider applies fix transparently. |
| config_change | CC | Configuration change on existing resource. |
| opt_in | OI | Fix available but applies to non-default option. |
| version_update | VU | Consumer must update pinned version or runtime. |
| redeploy | RD | Consumer must redeploy using existing configuration. |
| rebuild_and_redeploy | RR | Consumer must rebuild artifact with updated base. |
| destroy_recreate | DC | Resource must be destroyed and recreated. |
| rolling_replace | RL | Fleet replacement with coexistence during transition. |
| no_fix_available | NF | No vendor fix has been released. |

Table 5

*Table 4: Shared Responsibility (SR)*

| Value | Code | Description |
|---|---|---|
| provider_only | PO | Provider is solely responsible for remediation. |
| customer_action_required | CA | Provider fix exists but customer action is needed. |
| customer_only | CO | Customer is solely responsible. |
| shared | SH | Remediation requires coordinated provider and customer action. |

Table 6

*Table 5: Resource Lifecycle (RL)*

| Value | Code | Description |
|---|---|---|
| ephemeral | EP | Short-lived; replaced rather than patched. |
| stateful_managed | SM | Long-lived; provider manages OS and runtime. |
| stateful_customer | SC | Long-lived; customer manages OS and runtime. |
| config_only | CF | No runtime; configuration-only resource. |
| global_control_plane | GC | Shared control-plane infrastructure. |

Table 7

*Table 6: Existing Deployments Remain Vulnerable (EV)*

| Value | Code | Description |
|---|---|---|
| true | T | Resources deployed before the fix remain vulnerable. |
| false | F | Provider fix applies retroactively to existing resources. |

Table 8

*Table 7: Vulnerability Published Date (PP)*

Unix epoch timestamp (integer seconds). REQUIRED. The date the vulnerability was publicly disclosed. Corresponds to temporal.vuln_published_date converted to epoch seconds.

*Table 8: Service Available Date (SA)*

Unix epoch timestamp (integer seconds). REQUIRED. The date the cloud service became generally available. Corresponds to temporal.service_available_date converted to epoch seconds.

#### 4.1.2.2. Qualifiers

Qualifiers appear after the # delimiter as positional colon-separated values with no metric keys. All three qualifiers are REQUIRED and MUST appear in the following fixed order:

| Position | Field | Description |
|---|---|---|
| 1 | vuln_id | Vulnerability identifier (e.g., CVE-2024-6387). |
| 2 | service | Provider service key. |
| 3 | resource_type | Provider resource type key. |

Table 9

#### 4.1.2.3. Computation and Validation

A conformant CRIT producer:

* MUST compute vectorString from the record’s own fields.

* MUST include all registered metrics in canonical order (CP, VS, FP, SR, RL, EV, PP, SA).

* MUST use the CRIT specification version the record conforms to as the semver prefix.

* MUST use only registered abbreviation codes from the tables in Section 4.1.2.1.

* MUST ensure qualifier values match the corresponding record field values exactly.

* MAY append additional metrics after the registered set and before the # delimiter.

A conformant CRIT consumer:

* MUST validate all known metric keys and their values.

* MUST ignore unknown metric keys without error.

* MUST reject a vectorString missing any registered metric.

* MUST reject a vectorString where registered metrics appear out of canonical order.

* SHOULD emit a warning when encountering unknown metric keys.

#### 4.1.2.4. Information Scope

The CRIT vector string is a lossy encoding. It carries 12 fields from the full CRIT record; the remaining fields are not representable in the vector and are discarded during conversion.

Fields carried in the vector string:

* CRIT specification version (prefix).

* Six enumerated classification fields: provider (CP), vex_status (VS), fix_propagation (FP), shared_responsibility (SR), resource_lifecycle (RL), existing_deployments_remain_vulnerable (EV).

* Two required temporal dates as epoch timestamps: vuln_published_date (PP), service_available_date (SA).

* Three identity qualifiers: vuln_id, service, resource_type.

Fields not carried in the vector string:

* template and template_format — recoverable via dictionary lookup from the (provider, service, resource_type) tuple embedded in the vector qualifiers and CP metric.

* Optional temporal dates: vulnerability_introduced_date, provider_acknowledged_date, provider_fix_date, customer_deadline_date, and related fields. A producer MAY include these as additional metrics appended after the registered set.

* Fix version details: version_type, comparison, version, build_date, auto_upgrade, note.

* Remediation actions: the complete remediation_actions array including step-by-step instructions, downtime estimates, and compensating control flags.

* Detection entries: the complete detections array including detection queries, query languages, detection phases, and pending reasons.

* Advisory metadata: advisory_id, advisory_url.

* Any producer-appended additional metrics beyond the registered set are also not preserved when converting from a full JSON record to a vector string, unless the converter explicitly retains them.

A consumer MUST NOT treat a vectorString as a complete record representation. A consumer MUST use the full JSON record for operational decisions that require fields not carried in the vector, including but not limited to: deploying detection queries, executing remediation actions, evaluating fix version comparisons, and computing exposure windows.

## 4.2. Resource Template

| Field | Required | Type | Description |
|---|---|---|---|
| template | REQUIRED | string | Parameterised identifier string. After all named variables are substituted, the result MUST be a valid provider identifier for |

| | | | |
|---|---|---|---|
| | | | the declared template_format. |
| template_format | REQUIRED | enum | One of: aws_arn, azure_resource_id, gcp_resource_name, cloudflare_locator, oracle_ocid, salesforce_url, sap_btp_url, sap_odata_url, sap_sf_url, servicenow_table_url. |

Table 10

## 4.3. Temporal Fields and Exposure Window

These fields collectively define the bounds of exposure. No single field closes the exposure window for a given consumer resource; see Section 7 for the formal computation.

```
{
  "temporal": {
    "service_available_date": "<date>",
    "vulnerability_introduced_date": "<date>",
    "vulnerability_introduced_date_estimated": "<boolean>",
    "vuln_published_date": "<date>",
    "provider_acknowledged_date": "<date>",
    "provider_fix_date": "<date>",
    "customer_deadline_date": "<date>",
    "customer_deadline_source": "<enum>"
  }
}
```

Figure 3: Temporal Object

service_available_date (OPTIONAL):
: When the provider first made this service or feature generally available. Bounds the earliest any resource could have been deployed into a vulnerable configuration.

vulnerability_introduced_date (OPTIONAL):
: When the vulnerability was first present. MAY predate vuln_published_date by months or years. When present, MUST be used as W_start of the exposure window.

vulnerability_introduced_date_estimated (OPTIONAL):
When true, vulnerability_introduced_date is an estimate. Consumers SHOULD surface this flag in exposure window reporting.

vuln_published_date (REQUIRED):
Date the vulnerability record was first published. MUST match the vulnerability record's datePublished field.

provider_acknowledged_date (OPTIONAL):
When the provider first confirmed the vulnerability.

provider_fix_date (OPTIONAL):
When the provider made a fix generally available. MUST NOT be interpreted as the date a consumer resource is remediated. Absent when no fix has been released.

customer_deadline_date (OPTIONAL):
A normative remediation deadline. Conformant consumer tools SHOULD use this for SLA computations.

customer_deadline_source (OPTIONAL):
One of: cisa_kev, pci_dss, hipaa, sox, internal_policy, other. REQUIRED when customer_deadline_date is present.

## 4.4. Fix Propagation and Remediation Actions

### 4.4.1. Resource Lifecycle

The resource_lifecycle field characterises the operational behaviour of the resource type with respect to data durability and replacement. This is a property of the resource type, not of any specific consumer deployment.

| Value | Meaning |
|---|---|
| ephemeral | No durable state; can be replaced without data concern. Examples: Lambda functions, containers, serverless workers. |
| stateful_managed | Provider manages data durability but replacement is disruptive. Examples: RDS, ElastiCache, Cosmos DB, Cloud SQL. |
| stateful_customer | Customer owns data migration entirely. Examples: EBS-backed EC2, self-managed databases on compute. |
| config_only | Pure configuration with no application data. Examples: IAM roles, security groups, WAF rules, DNS records. |
| global_control_plane | Globally scoped; changes propagate with eventual consistency. Examples: CloudFront, Route53, GCP global forwarding rules. |

Table 11

#### 4.4.2. Shared Responsibility

| Value | Meaning |
|---|---|
| provider_only | Provider remediates transparently. The exposure window closes automatically at provider_fix_date for all resources. |
| customer_action_required | A fix is available but the consumer MUST take explicit action. provider_fix_date does not close the window for existing resources. |
| customer_only | Misconfiguration or insecure default. No provider fix involved. |

| | | |
|---|---|---|
| | No provider_fix_date. | |
| shared | Both provider and consumer action are required. | |

Table 12

#### 4.4.3. Fix Propagation

For package vulnerabilities, remediation status is largely derivable from a version comparison: if the installed version is at or above the fixed version, the package is remediated. Cloud resources have no equivalent. There is no installed version to query. A fix becoming available at the provider level does not mean any running resource is remediated. Whether a specific resource is exposed depends on when it was deployed, what action the consumer has taken since, and how the fix propagates to existing resources.

Some fixes apply automatically to all existing resources regardless of deployment date. Most do not. A resource deployed before the fix date under a rebuild_and_redeploy propagation type is still fully exposed the day after provider_fix_date. A resource of the same type deployed the day after is clean. The two resources are indistinguishable by version string -- because neither has one.

existing_deployments_remain_vulnerable makes this distinction normative and machine-readable. It cannot be derived from a version comparison.

fix_propagation (REQUIRED):
: The mechanism by which the fix reaches existing deployed resources. See values below.

existing_deployments_remain_vulnerable (REQUIRED):
: When true, resources deployed before provider_fix_date remain in the exposure window unless an explicit consumer action has been taken. MUST be false only when fix_propagation is automatic AND shared_responsibility is provider_only.

Fix propagation enum values:

| Value | Meaning | Typical Consumer Action |
|---|---|---|
| automatic | Provider applies the fix transparently to | Verify fix is active; no |

| | | |
|---|---|---|
| | all existing resources. | operational change required. |
| config_change | A configuration change on the existing resource is sufficient. | Apply the change via API, console, or IaC. |
| opt_in | A fix exists but applies to a non-default option. | Enable the option; update IaC defaults. |
| version_update | Update a pinned version, runtime, or dependency reference. | Update version reference; trigger redeployment if required. |
| redeploy | Redeploy using the existing configuration. | Trigger redeployment. |
| rebuild_and_redeploy | Rebuild the artifact with updated base or patched dependencies, then redeploy. | Update base image, rebuild, push, redeploy. |
| destroy_recreate | The resource MUST be destroyed and recreated. In-place upgrade not supported. | Back up state if applicable, destroy, recreate at fixed version. |
| rolling_replace | Fleet or cluster replacement; old and new instances coexist during transition. | Trigger rolling update; monitor fleet until 100% replacement. |
| no_fix_available | Provider has not released a fix. provider_fix_date MUST be absent. | Apply compensating controls; monitor advisory. |

Table 13

#### 4.4.4. Remediation Actions

remediation_actions is an ordered array. The first entry is the primary recommended path. A consumer tool SHOULD present actions in declared sequence order.

| Field | Required | Description |
|---|---|---|
| sequence | REQUIRED | 1-based ordering index. MUST be unique and contiguous within the array starting at 1. |
| type | REQUIRED | One of the fix_propagation enum values. |
| title | REQUIRED | Short imperative description suitable for a task or ticket title. |
| description | REQUIRED | Step-by-step instructions sufficient for an engineer to execute without additional research. SHOULD include CLI invocations or IaC equivalents where applicable. |
| provider_guidance_url | OPTIONAL | Direct link to the provider’s advisory or remediation documentation. |
| auto_remediable | REQUIRED | Whether a conformant consumer tool MAY automate this action without human approval. |
| requires_downtime | REQUIRED | Whether this action causes a service interruption. |
| stateful_impact | REQUIRED | One of: none, backup_recommended, |

| | | | |
|---|---|---|---|
| | | | backup_restore_required, data_migration_required. |
| estimated_downtime_range_seconds | OPTIONAL | Object with min and max integer bounds. REQUIRED when requires_downtime is true. Informative only. |
| compensating_control | REQUIRED | When true, this action reduces exploitability but does not fully remediate. A record with only compensating actions MUST have vex_status of affected, not fixed. |

Table 14

## 4.5. Provider Fix Version

Cloud resources do not use package-style versioning. There is no semver string to compare against a fixed bound, no registry entry to look up, and no universal version format that applies across providers or even across services within a single provider. "Version" for a cloud resource might mean an engine release string, a runtime build date, a Kubernetes minor version within a release channel, a container image digest, or a platform image creation date -- depending on the service. In some cases, such as Cloudflare Workers, there is no consumer-visible version at all; only a platform build date.

The provider_fix_version field is a discriminated object whose structure is determined by the version_type discriminator. Each version_type value defines a specific set of fields and a comparison operator that together give a consumer everything needed to evaluate whether a deployed resource meets the fix threshold.

### 4.5.1. Envelope

| Field | Required | Description |
|---|---|---|
| version_type | REQUIRED | Discriminator. Determines which additional fields are present. See Sections Section 4.5.3 through |

| | | |
|---|---|---|
| | | Section 4.5.7. |
| comparison | REQUIRED | How a consumer evaluates whether a deployed resource meets the fix threshold. See Section 4.5.2. |
| auto_upgrade | OPTIONAL | When false, the provider does not automatically apply this version update. When false, existing_deployments_remain_vulnerable MUST be true. |
| note | OPTIONAL | Human-readable clarification. REQUIRED when a fix arrives at different dates across release channels. |

Table 15

### 4.5.2. Comparison Values

| Value | Meaning |
|---|---|
| gte | Deployed version MUST be greater than or equal to the specified value per the service's versioning scheme. |
| exact | Deployed version MUST exactly match. Used for content-addressed identifiers (image digests, AMI IDs, OCIDs). |
| date_gte | Resource's runtime build date or deployment date MUST be on or after the specified build_date. |
| channel_and_gte | Resource MUST be subscribed to a qualifying release channel AND be at or above the specified version within that channel. |

Table 16

#### 4.5.3. AWS Version Types

Defined version_type values for AWS services: runtime (Lambda and runtime-based services), engine_version (RDS, ElastiCache, Redshift), ami (EC2 and AMI-backed services), agent_version (SSM Agent, CodeDeploy Agent, ECS Agent), kubernetes_version (EKS), container_image (ECS tasks), managed_policy_version (AWS-managed IAM policies).

For engine_version, the auto_upgrade field indicates whether RDS auto minor version upgrade is sufficient. When auto_upgrade is false, consumers must explicitly trigger the upgrade and existing_deployments_remain_vulnerable MUST be true.

For container_image, image_digest (SHA256) is RECOMMENDED over image_tag. When image_digest is present with comparison: exact, consumers MUST verify digest, not tag. Tags are mutable and MUST NOT be used as the sole verification method.

#### 4.5.4. Azure Version Types

Defined version_type values for Azure services: api_version (ARM API operations), kubernetes_version (AKS clusters and node pools, with optional node_image_version), extension_version (VM Extensions), os_image_version (VM Scale Sets), runtime_version (App Service and Azure Functions).

#### 4.5.5. GCP Version Types

Defined version_type values for GCP services: kubernetes_version (GKE, with release_channel field and channel_and_gte comparison for channel-gated fixes), database_version (Cloud SQL), runtime_version (Cloud Functions and Cloud Run, using date_gte comparison), image_family (Compute Engine public image families).

For GKE, fix availability differs by release channel (RAPID, REGULAR, STABLE). The note field MUST enumerate channel-specific availability dates.

#### 4.5.6. Cloudflare Version Types

Cloudflare Workers does not expose a semantic version. Defined version_type values: runtime_build_date (Workers runtime, using date_gte comparison against build_date), deployment_id (Pages or Workers deployments where the fix requires consumer-controlled redeployment).

#### 4.5.7. Oracle Version Types

Defined version_type values for Oracle Cloud services: database_version (Autonomous Database, Base Database Service), kubernetes_version (OKE, with optional node_pool_image), image_ocid (Compute platform images, using date_gte comparison against build_date; OCID is region-specific so build_date is the normative threshold).

### 4.6. Detection Fields

Detection fields enable consumers to deploy log queries, metric filters, and alerting rules that identify vulnerable configurations, active exploitation, or configuration drift. A record with vex_status of affected or fixed SHOULD include at least one detection entry.

| Field | Required | Description |
|---|---|---|
| provider | REQUIRED | Cloud provider for this detection. |
| service | REQUIRED | Log, event, or security service for which the query is written. See Section 4.6.1. |
| query_language | REQUIRED | Query language of the query string. See Section 4.6.2. |
| query | REQUIRED | Detection query string. MUST be syntactically valid for the declared query_language. Variable slots MAY appear where consumer-specific values must be substituted before deployment. |
| detection_phase | REQUIRED | See Section 4.6.3. |
| description | REQUIRED | Explanation of what the query detects, why it is relevant, and any false positive caveats. |
| pending_reason | OPTIONAL | When present, indicates this detection entry is a placeholder without a functional query. The query field MUST be an empty string when pending_reason is set. See Section 4.6.4. |

Table 17

### 4.6.1. Detection Service Values

| Provider | Service values |
|---|---|
| aws | cloudwatch_logs_insights, cloudwatch_metric_filter, cloudtrail, security_hub, guardduty, config_rule |
| azure | monitor_kql, sentinel_analytics, defender_alert |
| gcp | cloud_logging, security_command_center, chronicle |

| cloudflare | logpush, firewall_events |
|---|---|
| oracle | oci_logging, cloud_guard |

Table 18

#### 4.6.2. Query Language Values

| Value | Language |
|---|---|
| cwli | CloudWatch Logs Insights |
| cloudwatch_filter | CloudWatch Metric Filter pattern syntax |
| kql | Kusto Query Language (Azure Monitor and Sentinel) |
| gcp_logging_filter | GCP Cloud Logging filter syntax |
| oci_logging_query | OCI Logging query syntax |
| lucene | Lucene query syntax (Cloudflare and SIEM integrations) |

Table 19

#### 4.6.3. Detection Phase

The detection_phase field is normative. A consumer tool MUST use this field to determine whether a detection is currently applicable and whether it should remain active after remediation.

| Value | Meaning | Retention Policy |
|---|---|---|
| pre_fix | Detects the vulnerable condition. MAY become misleading after remediation. | Deactivate or suppress after per-resource remediation is confirmed. |
| exploitation | Detects active exploitation attempts regardless of fix status. | MUST remain active permanently. |
| post_fix | Detects exploitation attempts that remain possible after apparent remediation. | Activate at provider_fix_date; retain permanently. |
| misconfiguration | Detects drift back to a vulnerable configuration after remediation. A confirmed match MUST be treated as a window-reopening event. | MUST remain active indefinitely after any opt_in or config_change remediation. |

Table 20

A record with fix_propagation of opt_in or config_change MUST include at least one misconfiguration detection entry.

If a functional detection query cannot be authored at publication time, the producer MUST include a placeholder entry with detection_phase of misconfiguration and a pending_reason value from Section 4.6.4.

#### 4.6.4. Pending Detection Reasons

When a producer cannot author a functional detection query at publication time, the producer MUST still include a detection entry with detection_phase set to the required phase and pending_reason set to one of the following values. The query field MUST be an empty string. The description field SHOULD provide additional human-readable context explaining the gap.

A producer SHOULD publish an updated record with a functional query replacing the placeholder once the constraint is resolved.

| Value | Meaning |
|---|---|
| query_in_development | The detection query is being authored or tested and will be published in a future record update. |
| awaiting_provider_telemetry | The cloud provider does not yet expose the log, event, or API data needed to detect this condition. Pending provider capability. |
| no_detection_surface | No provider service currently offers telemetry sufficient to detect this misconfiguration programmatically. This value indicates a permanent or long-term gap. |
| access_constraint | The record author lacks the provider environment access needed to develop and validate the query. |
| pending_review | A candidate query exists but is under review (security, accuracy, or false-positive assessment) before publication. |

Table 21

A consumer MUST NOT deploy a detection entry that has pending_reason set. A consumer SHOULD surface placeholder entries in operator-facing dashboards to indicate detection coverage gaps.

## 4.7. Provider Advisory

| Field | Required | Description |
|---|---|---|
| advisory_id | OPTIONAL | Provider’s own advisory identifier (e.g., ALAS2-2024-2456, MSRC-2024-0034, GCP-SA-2024-001). |

| | | |
|---|---|---|
| advisory_url | OPTIONAL | Direct URL to the provider's security advisory. |

Table 22

## 4.8. VEX Status

The vex_status field aligns CRIT records with the OpenVEX ([OpenVEX]) / CSAF VEX ([CSAF-VEX]) vocabulary for composability with VEX-aware tooling.

| Value | Meaning |
|---|---|
| affected | The resource type is affected. No fix is available, or fix has not been applied. |
| not_affected | The resource type is not affected, or the vulnerability is not reachable in this deployment context. |
| fixed | A provider fix is available and provider_fix_date is set. |
| under_investigation | Provider has acknowledged the vulnerability but has not yet confirmed affected status. |

Table 23

A consumer MUST treat vex_status as a record-level statement about provider fix availability, not as a per-resource remediation status. A record with vex_status = fixed and existing_deployments_remain_vulnerable = true represents the common real-world condition: a fix exists at the provider level, but existing deployed resources are not automatically remediated. Both facts are simultaneously true and MUST both be surfaced to operators.

# 5. Provider Template Reference

## 5.1. AWS ARN

Canonical formats:

arn:aws:{service-prefix}:{region}:{account}:{resource-type}/{id}
arn:aws:{service-prefix}:{region}:{account}:{resource-type}:{id}

Figure 4

The {service-prefix} slot is always hardcoded (e.g., iam, s3, ec2, lambda, eks, rds).

The {region} slot MUST be hardcoded to us-east-1 for globally-scoped services whose ARN schema requires the region field to carry a fixed value: iam, cloudfront, route53, waf, wafv2, shield, organizations, sts, globalaccelerator. For globally-scoped services whose ARN schema structurally omits the region field (the field is positionally present but the value is the empty string), the region slot MUST be empty ({region=}). S3 buckets and objects are the principal example: the ARN format is arn:aws:s3:::{resource}. For all other AWS services the region slot MUST be a named variable or wildcard and MUST NOT be empty.

The {account} slot is a named variable, except for resource types whose ARN schema structurally omits the account field (e.g., S3 buckets and objects), in which case it MUST be empty ({account=}). The {resource-type} slot is hardcoded or empty per the service schema. The {resource-id} slot is a named variable or wildcard.

```
arn:aws:iam:{region=us-east-1}:{account}:role/{resource-id}
arn:aws:s3:{region=}:{account=}:{resource-id}
arn:aws:ec2:{region}:{account}:instance/{resource-id}
arn:aws:lambda:{region}:{account}:function:{resource-id}
arn:aws:eks:{region}:{account}:cluster/{resource-id}
```

Figure 5: AWS ARN Examples

## 5.2. Azure Resource ID

Canonical format:

```
/subscriptions/{subscriptionId}/resourceGroups/{resourceGroup}
  /providers/{namespace}/{type}/{name}
```

Figure 6

{subscriptionId} and {name} are always named variables. {resourceGroup} is a named variable or wildcard. {namespace} and {type} are always hardcoded (e.g., Microsoft.Compute/virtualMachines, Microsoft.ContainerService/managedClusters).

## 5.3. GCP Resource Name

Canonical format:

```
//{api}.googleapis.com/{collection-path}
```

Figure 7

{api} is always hardcoded (e.g., compute, container, sqladmin). {project} is always a named variable. {zone} is a named variable for zonal resources and empty ({zone=}) for global or regional resources.

### 5.4. Cloudflare Locator

Canonical format:

```
com.cloudflare.api.account.{account_id}.{resource-type}.{id}
```

Figure 8

Cloudflare resources are globally scoped. There is no region component. A CRIT producer MUST NOT add a region slot to a Cloudflare template. {resource-type} is always hardcoded (e.g., worker, r2_bucket, zone, d1_database).

### 5.5. Oracle OCID

Canonical formats:

```
ocid1.{type}.{realm}.{region}..{unique-id}       (regional)
ocid1.{type}.{realm}...{unique-id}               (global)
```

Figure 9

{type} is always hardcoded. {realm} is hardcoded to oc1 for commercial regions. Separate CRIT records SHOULD be produced for government realms (oc2, oc3) when fix timelines differ. {region} is a named variable for regional resources and empty ({region=}) for global resources.

### 5.6. Parameter Naming Conventions

Slot field names vary between providers by design. AWS ARNs use a consistent positional structure so slot names are uniform across services (region, account, resource-id). GCP, Azure, Oracle, and Cloudflare use service-specific path components (project, location, subscriptionId, resourceGroup, realm, account_id) because their resource path structures are service-specific. This asymmetry is intentional: CRIT adopts provider-native naming conventions rather than imposing a normalised abstraction. Consumers SHOULD expect different slot vocabularies per provider and MUST NOT assume that all providers use the same field names.

# 6. Variable Resolution Rules

## 6.1. Resolution Order

A consumer resolving a CRIT template to a live identifier MUST apply substitutions in the following order:

1. Replace all hardcoded slots ({field=literal}) with their literal value.

2. Replace all empty slots ({field=}) with the empty string.

3. Replace all named variable slots ({field}) with consumer-supplied concrete values.

4. Wildcard slots ({field=*}) MUST NOT be resolved to a live identifier. For inventory enumeration, a consumer MAY enumerate known values to produce a set of resolved templates.

After step 3, the resulting string MUST be a valid provider identifier conforming to the declared template_format. A consumer MUST validate this and MUST reject a template that fails validation after full substitution.

## 6.2. Reserved Field Names

The following field names carry defined semantics across all providers. A CRIT producer MUST use these names where applicable and MUST NOT reuse them for different semantics.

| Field Name | Semantics |
|---|---|
| account | AWS account ID or equivalent top-level ownership identifier. |
| subscriptionId | Azure subscription ID. |
| project | GCP project ID. |
| account_id | Cloudflare account ID. |
| region | Provider geographic region identifier. |
| zone | Provider availability zone or GCP zone identifier. |

| | |
|---|---|
| location | GCP region or multi-region identifier as used in resource paths. |
| resource-id | Unique identifier of the specific resource instance. |
| name | Azure resource name. |
| id | Cloudflare or Oracle resource identifier. |
| unique-id | Oracle OCID unique identifier component. |
| realm | Oracle OCID realm component. |
| service-prefix | AWS service prefix as used in ARN construction. |
| resource-type | Resource type component within an ARN. |
| namespace | Azure resource provider namespace. |
| type | Azure resource type or Oracle OCID type component. |
| api | GCP API host prefix. |

Table 24

# 7. Exposure Window Computation

## 7.1. Definition

For package vulnerabilities, an exposure window can be approximated from version data alone: a package was exposed from the time the vulnerable version was released until the time the fixed version was installed. Cloud resources have no equivalent computation. There is no "installed version" to timestamp, no registry entry recording when a resource was last updated, and no version comparison that determines whether a specific running resource is currently in the affected range.

The CRIT exposure window is therefore defined over time and consumer action, not over version ranges. A resource enters the window when it is deployed into a vulnerable configuration. It exits the window

when a qualifying remediation event is recorded -- which may be long after provider_fix_date for resources where existing_deployments_remain_vulnerable is true, or never, for resources under no_fix_available propagation.

Formally, the exposure window is the interval [W_start, W_end] where:

* W_start = vulnerability_introduced_date when present; otherwise vuln_published_date. When vulnerability_introduced_date_estimated is true, consumers SHOULD indicate this in user-facing reporting.

* W_end is determined per Section 7.2.

## 7.2. Record-Level W_end

| Condition | W_end |
|---|---|
| shared_responsibility = provider_only AND provider_fix_date is present | W_end = provider_fix_date. Window closed for all resources automatically. existing_deployments_remain_vulnerable MUST be false. |
| shared_responsibility is customer_action_required or shared | W_end undefined at record level. provider_fix_date opens remediation possibility but does not close the window. Per-resource closure requires a confirmed consumer action. |
| shared_responsibility = customer_only | W_end undefined. No provider_fix_date. Per-resource closure requires confirmed consumer remediation. |
| fix_propagation = no_fix_available | W_end = null. Window open. provider_fix_date MUST be absent. |
| provider_fix_date absent for any other reason | W_end = null. Window open. |

Table 25

## 7.3. The Deployed-Before-Fix Problem

When existing_deployments_remain_vulnerable is true, the exposure window for a specific resource instance is NOT closed by provider_fix_date. A consumer MUST apply the following logic per resource:

```
if resource.deployed_date < provider_fix_date
  AND existing_deployments_remain_vulnerable == true
  AND no confirmed remediation action recorded for this resource:
    resource.exposure_window_end = null  // open
```

Figure 10

A consumer MUST record a per-resource remediation event to close the window for that resource. A consumer MUST NOT mark a resource as remediated solely because provider_fix_date has passed.

### 7.4. Opt-In and Config Change Drift

When fix_propagation is opt_in or config_change, a remediation may be reversed by a subsequent configuration change, reopening the window. When a misconfiguration-phase detection fires for a resource, a consumer MUST treat this as a window-reopening event. A consumer MUST keep misconfiguration-phase detections active indefinitely for any resource remediated via opt_in or config_change.

### 7.5. Rolling Replace Fleet Exposure

When fix_propagation is rolling_replace, the exposure window is partially open during the fleet transition. A consumer MUST NOT consider the window closed until fleet replacement is confirmed at 100%.

### 7.6. Channel-Gated Exposure

When provider_fix_version.comparison is channel_and_gte, the effective fix availability date differs by release channel. A consumer MUST use the channel-specific fix date derived from the note field when computing per-cluster exposure windows.

## 8. Conformance

### 8.1. Producer Conformance

A conformant CRIT producer MUST:

* Emit records that validate against the schema defined in Section 4.

* Enforce the natural key uniqueness constraint: no two records in a corpus MAY share (vuln_id, provider, service, resource_type).

* Apply slot state selection rules defined in Section 3.5.

* Apply AWS region hardcoding rules defined in Section 5.1.

* Set existing_deployments_remain_vulnerable = false only when fix_propagation = automatic AND shared_responsibility = provider_only.

* Set existing_deployments_remain_vulnerable = true when provider_fix_version.auto_upgrade is present and false.

* Set fix_propagation = no_fix_available and omit provider_fix_date when no fix exists.

* Include at least one remediation_actions entry for every record where vex_status is affected or fixed.

* Use ISO 8601 [ISO8601] full-date format for all date fields.

* Include at least one misconfiguration-phase detection entry for records where fix_propagation is opt_in or config_change. A placeholder entry with pending_reason (Section 4.6.4) satisfies this requirement.

* Compute vectorString as the canonical CRIT vector string from the record’s own fields per Section 4.1.2.

* Encode temporal.vuln_published_date as the PP metric value in Unix epoch seconds (UTC).

* Encode temporal.service_available_date as the SA metric value in Unix epoch seconds (UTC).

A conformant CRIT producer SHOULD:

* Include at least one detection entry for records where vex_status is affected or fixed.

* Populate provider_advisory when a provider security advisory exists.

* Populate vulnerability_introduced_date when determinable; set vulnerability_introduced_date_estimated = true when the date is an estimate.

## 8.2. Consumer Conformance

A conformant CRIT consumer MUST:

* Treat provider_fix_date as closing the exposure window only when existing_deployments_remain_vulnerable is false.

* Not substitute hardcoded slot values with alternative values.

* Not use wildcard templates as live provider API identifiers.

* Track per-resource remediation events separately from record-level vex_status.

* Treat a misconfiguration-phase detection match as a window-reopening event.

* Keep misconfiguration-phase detections active indefinitely once deployed.

* Use channel-specific fix dates for channel_and_gte version types when per-resource channel enrollment is known.

* Prefer image_digest over image_tag for container_image version comparison when both are present.

* Ignore unknown metric keys in a vectorString without error (forward compatibility per Section 4.1.2.3).

* Reject a vectorString missing any registered metric.

* Not treat a vectorString as a complete record representation. Use the full JSON record for operational decisions requiring fields not carried in the vector (see Section 4.1.2.4).

A conformant CRIT consumer SHOULD:

* Present remediation_actions in declared sequence order.

* Substitute consumer-specific named variable values into detection query slots before deploying queries.

* Apply customer_deadline_date when computing remediation SLAs.

* Surface vulnerability_introduced_date_estimated = true in operator-facing exposure window reporting.

## 9. Upstream Schema Integration

### 9.1. Integration Strategy and Phasing

CRIT data is published via two upstream vulnerability schema ecosystems: the CVE List v5 ADP container and the OSV schema. Each integration follows a two-phase strategy.

*Phase 1 -- Extension (current):* CRIT records are embedded as custom x_ properties within conformant records of the target schema. This is immediately deployable without requiring changes to either upstream schema. Phase 1 records are fully schema-valid because both CVEListv5 and OSV permit additional properties with the x_ prefix.

*Phase 2 -- Native integration (proposed):* CRIT fields are expressed using native objects defined by the upstream schema wherever a semantic mapping exists. Fields without a native mapping continue to use x_crit_* prefixed properties within the appropriate extension points. Phase 2 requires coordination with CVEProject and OpenSSF but does not require either upstream schema to define new first-class fields for all CRIT concepts.

A producer MUST NOT remove Phase 1 fields until: Phase 2 native fields have been published for at least one full release cycle of the target schema; the cloud:* ecosystem namespace has been formally registered with OSV schema maintainers or the ADP native field mapping accepted by the CVEProject schema working group; downstream consumers have confirmed migration to Phase 2; and a 90-day deprecation notice has been in the relevant records.

### 9.2. CVE List v5 ADP Container Integration

Vulnetix publishes CRIT data as an Authorized Data Publisher (ADP) in CVEListv5 records. The Vulnetix ADP container is identified by providerMetadata.shortName = "VVD" or by Vulnetix’s orgId in the containers.adp[] array.

#### 9.2.1. Phase 1 -- x_crit Extension (Current)

In Phase 1 a single top-level x_crit field in the Vulnetix ADP container carries an array of CRIT records. The x_crit array MUST contain one entry per natural key tuple applicable to the CVE. The vuln_id within each entry MUST match the cveMetadata.cveId of the enclosing CVEListv5 record.

#### 9.2.2. Phase 2 -- Native ADP Container (Proposed)

In Phase 2, each CRIT record contributes one entry to the ADP affected[] array. Provider-native CVEListv5 fields carry data wherever a mapping exists; fields without a native mapping use x_crit_* extension properties on the affected[] item or at the ADP container level.

#### 9.2.3. CVEListv5 Field Mapping

provider:
affected[].vendor -- Provider key.

service:
affected[].product -- Service key.

resource_type:
affected[].modules[] -- Array of resource type strings.

template:
affected[].platforms[] -- CRIT template strings as platform descriptors.

provider_fix_version (range bound):
affected[].versions[].lessThan and changes[].at -- Range [0, fix_version) expressed natively; full subschema in residual field x_crit_fix_version.

temporal.*_date fields:
ADP container timeline[] array -- Each date as a timeline entry with a descriptive value string.

provider_advisory CVSS fields:
ADP container metrics[] array -- cvssV3_1 or cvssV4_0 per the vector string prefix.

provider_advisory.advisory_url:
ADP container references[] array with tags: ["vendor-advisory"].

remediation_actions (non-compensating):
ADP container solutions[] array -- One entry per action.

remediation_actions (compensating_control: true):
ADP container workarounds[] array.

Residual fields (no native CVEListv5 mapping):
vex_status -> x_crit_vex_status; fix_propagation -> x_crit_fix_propagation; existing_deployments_remain_vulnerable ->

x_crit_existing_deployments_remain_vulnerable;
shared_responsibility -> x_crit_shared_responsibility;
resource_lifecycle -> x_crit_resource_lifecycle;
provider_fix_version (full subschema) -> x_crit_fix_version;
template with slot syntax -> x_crit_template +
x_crit_template_format; detections[] -> x_crit_detections.

### 9.3. OSV Schema Integration

Publishers may produce CRIT data in OSV schema format for consumption by OSV.dev and compatible tooling.

#### 9.3.1. Naming Conventions

Cloud provider ecosystems are expressed as cloud:<provider> (e.g., cloud:aws, cloud:azure, cloud:gcp). This namespace is proposed for registration with the OSV schema ecosystem list. Until registered, tooling that does not recognise a cloud:* ecosystem MUST NOT reject records using it.

Package names use the convention <service>:<resource_type> (e.g., rds:db, aks:cluster, lambda:function).

PURLs follow the form pkg:cloud/<provider>/<service>/<resource_type> (e.g., pkg:cloud/aws/rds/db). The cloud type is observed in the OSV ecosystem but is not a registered type in the PURL specification [PURL]. This specification acknowledges its use for OSV integration but does not define or govern the pkg:cloud/ type itself.

OSV record IDs follow the convention: OSV-<year>-<ID>.

#### 9.3.2. Phase 1 -- database_specific Extension (Current)

Each OSV record carries one affected[] entry per CRIT natural key tuple. The full CRIT record is embedded in affected[].database_specific.x_crit. Multiple CRIT records for the same vulnerability are published as separate OSV records, each with a distinct ID and a single affected[] entry. The aliases array on all records includes the shared vuln_id.

#### 9.3.3. OSV Field Mapping

provider:
affected[].package.ecosystem -- "cloud:<provider>" (e.g., "cloud:aws").

service + resource_type:
affected[].package.name -- "<service>:<resource_type>" (e.g., "rds:db").

provider + service + resource_type:
affected[].package.purl -- "pkg:cloud/<provider>/<service>/<resource_type>".

provider_fix_version (range bound):
affected[].ranges[].events -- introduced and fixed events.

temporal.vuln_published_date:
published -- RFC3339 format.

provider_fix_date:
modified -- Set to the most recent significant update date.

provider_advisory.provider_cvss_vector:
severity[] with type: "CVSS_V3" or "CVSS_V4".

provider_advisory.advisory_url:
references[] with type: "ADVISORY".

vuln_id:
aliases[].

Residual fields in ecosystem_specific:
fix_propagation -> x_crit_fix_propagation;
existing_deployments_remain_vulnerable -> x_crit_existing_deployments_remain_vulnerable; all temporal fields -> x_crit_temporal; detections[] -> x_crit_detections; remediation_actions[] -> x_crit_remediation_actions; vex_status -> x_crit_vex_status.

Residual fields in database_specific:
provider_advisory.advisory_id -> x_crit_provider_advisory_id.

## 10. IANA Considerations

This document has no IANA actions. Future revisions targeting standards track may request registration of the cloud PURL type with the PURL specification maintainers, and registration of the cloud:* ecosystem namespace with the OSV schema maintainers.

## 11. Security Considerations

### 11.1. Detection Query Sensitivity

Detection strings specify exact log filter patterns for identifying vulnerable configurations and exploitation. A corpus of CRIT detection entries reveals what a consumer is and is not monitoring for. CRIT records SHOULD be treated as security-sensitive and access-controlled in consumer systems.

### 11.2. Exposure Window Date Sensitivity

The combination of vulnerability_introduced_date, provider_fix_date, and existing_deployments_remain_vulnerable can allow an adversary to infer whether specific consumer resources remain vulnerable. Consumers SHOULD NOT expose exposure window details in public-facing interfaces.

### 11.3. Compensating Control Disclosure

Remediation actions with compensating_control = true reveal which mitigating controls are in place. Consumers SHOULD NOT expose active compensating control details in contexts where that information assists an adversary in targeting unmitigated surface.

### 11.4. Template Wildcard Enumeration

Wildcard templates reveal the structural scope of a consumer’s cloud footprint. A consumer MUST NOT expose unresolved wildcard templates in contexts where asset enumeration is harmful.

### 11.5. Provider Fix Version Trust

provider_fix_version values are advisory in nature. A consumer MUST independently verify that a deployed resource meets the version threshold. Container image tags are mutable; digest comparison MUST be preferred. A consumer MUST NOT assume remediation solely on the basis of a version string match.

### 11.6. Natural Key Collision

A producer accepting CRIT records from multiple upstream sources MUST enforce natural key uniqueness before serving records. Duplicate natural keys with conflicting field values can cause consumers to make incorrect remediation decisions. Producers SHOULD define and expose a conflict resolution policy.

# 12. CRIT Dictionary

## 12.1. Definition

A CRIT Dictionary is a machine-readable catalogue of entries that enumerate the valid combinations of provider, service, and resource_type values recognised by this specification, together with the provider-native identifier template and supporting metadata for each combination. A dictionary entry is the atomic unit of service coverage: it asserts that a given cloud provider service and resource type is within CRIT scope and provides the template and slot semantics required to locate instances of that resource type in a consumer's inventory.

A CRIT Dictionary is not a vulnerability database and does not contain vulnerability-specific data. It is a stable reference layer that producers and consumers use to validate and resolve CRIT records. A CRIT record's (provider, service, resource_type) tuple MUST resolve to an entry in a conformant dictionary before the record is considered valid.

Two categories of dictionary exist:

Spec Default Dictionary:
: The normative dictionary published alongside this specification in the specification repository. It covers the providers described in Section 12.6 and representative service and resource type combinations for each. Producers and consumers MUST support the Spec Default Dictionary as a baseline.

Extended Dictionary:
: A superset of the Spec Default Dictionary produced by a Vulnetix deployment or third party. Extended dictionaries MAY add entries for services or resource types not present in the Spec Default Dictionary, and MAY add entries for additional providers. Extended dictionaries MUST NOT remove or alter the semantics of entries present in the Spec Default Dictionary.

## 12.2. Dictionary Entry Schema

Each dictionary entry is a JSON object. All fields except notes are REQUIRED.

```
{
  "provider":        "<enum: aws | azure | gcp | cloudflare | oracle
                         | salesforce | sap | servicenow>",
  "service":         "<string: normalised service key>",
  "resource_type":   "<string: resource type within service>",
  "template":        "<CRIT template string>",
  "template_format": "<enum: aws_arn | azure_resource_id | gcp_resource_name
                          | cloudflare_locator | oracle_ocid
                          | salesforce_url | sap_btp_url | sap_odata_url
                          | sap_sf_url | servicenow_table_url>",
  "region_behavior": "<enum: regional | global-only>",
  "notes":           "<string: optional annotation>"
}
```

Figure 11: Dictionary Entry Structure

provider:
The canonical provider key as defined in Section 4.1.

service:
The normalised service key (lowercase, underscores). This is the value used in the CRIT record service field and the second component of the natural key tuple. Where a provider uses multiple common names for the same service, the dictionary carries the canonical key; synonyms are resolved to it by the producer prior to record emission.

resource_type:
The resource type within the service. This is the value used in the CRIT record resource_type field. For services with multiple resource types, each type has its own dictionary entry with a distinct (provider, service, resource_type) natural key.

template:
The CRIT template string for this entry, expressed using the slot syntax defined in Section 3. After variable resolution, the string MUST be a valid provider identifier of the declared template_format.

template_format:
One of: aws_arn, azure_resource_id, gcp_resource_name, cloudflare_locator, oracle_ocid, salesforce_url, sap_btp_url, sap_odata_url, sap_sf_url, servicenow_table_url. The applicable value for each provider is defined in Section 12.6.

region_behavior:
: One of: regional (the {region} slot is a named variable, consumer-supplied) or global-only (the region slot is hardcoded in the template; the resource type is not regional).

notes:
: Optional human-readable annotation. Used to document aliasing, deprecation, or provider-specific constraints not expressible in other fields.

### 12.3. Dictionary Conformance

A conformant CRIT producer MUST:

* Validate each record's (provider, service, resource_type) tuple against a conformant dictionary before emitting the record.

* Use the template and template_format values from the matching dictionary entry as the basis for the record's template fields.

* Support the Spec Default Dictionary as a minimum baseline. An extended dictionary MAY be used in addition but not in place of the Spec Default Dictionary.

A conformant CRIT consumer MUST:

* Reject records whose (provider, service, resource_type) tuple does not resolve to an entry in any dictionary the consumer supports, rather than silently ignoring them.

* Use the dictionary entry's region_behavior field when constructing inventory queries from wildcard templates, to avoid submitting region-qualified identifiers for global-only resource types.

### 12.4. Dictionary Versioning

The Spec Default Dictionary is versioned alongside the CRIT specification. The dictionary version is the same as the semver string carried in the vectorString prefix of CRIT records. Additions of new entries within a minor version are backwards compatible. Removal or semantic modification of existing entries requires a major version increment.

Producers SHOULD include a dictionary_version field in their extended dictionaries to allow consumers to detect stale dictionary coverage.

### 12.5. Dictionary Governance

The Spec Default Dictionary is maintained alongside this specification and represents the minimum baseline catalogue. It is not an exhaustive enumeration of all cloud resource types. Additions to the Spec Default Dictionary within a minor version are backwards compatible; removal or semantic modification of existing entries requires a major version increment.

Extended Dictionaries are independently maintained by implementers and are subject to the conformance requirements of Section 12.3.

A conformant consumer MUST validate CRIT records against a dictionary before accepting them. Validation requires a known, trusted dictionary as the source of truth for the (provider, service, resource_type) tuple space. A consumer that does not maintain a dictionary cannot distinguish a valid tuple from an arbitrary one; therefore, dictionary availability is a prerequisite for conformant consumption.

A conformant producer MAY emit CRIT records without consulting a dictionary, but the resulting records are non-conformant until validated by a consumer that holds a dictionary containing matching entries. Producers SHOULD validate records against a dictionary before emission to avoid producing records that no conformant consumer will accept.

Cloud providers are encouraged to contribute dictionary entries for their services. Community contributions are accepted via the specification repository. No IANA registry is proposed for dictionary entries. The dictionary is a template catalogue scoped by provider, not a shared identifier namespace; provider-scoped entries do not compete for a global name and therefore do not require a central allocation authority.

### 12.6. Provider Identification Systems

Each provider covered by this specification maintains a published, authoritative identification scheme for its resources. CRIT dictionary entries do not define these schemes; they encode them as CRIT templates so that producers can emit identifiers that conform to the provider's own published format. Implementers adding dictionary entries for services not listed here MUST derive templates from the provider's current documentation; those entries MAY be contributed to the Spec Default Dictionary via the specification repository.

#### 12.6.1. Amazon Web Services (aws)

AWS resources are identified by Amazon Resource Names (ARNs) [AWS-ARN], a hierarchical URN-like format with the structure arn:partition:service:region:account-id:resource-type/resource-id. The ARN format is defined and maintained by Amazon Web Services; this specification does not alter or extend it. The template_format value for all AWS entries is aws_arn.

```
arn:aws:ec2:{region}:{account}:instance/{resource-id}
arn:aws:iam:{region=us-east-1}:{account}:role/{resource-id}
```

Figure 12: AWS CRIT template examples

The ARN service prefix matches the AWS service namespace documented in the IAM User Guide. Globally-scoped services either hardcode the region component (e.g., us-east-1) or structurally omit it (e.g., S3 buckets), following the AWS ARN specification for that service.

#### 12.6.2. Microsoft Azure (azure)

Azure resources are identified by Azure Resource IDs [Azure-ResourceID], hierarchical paths within the Azure Resource Manager (ARM) namespace. The path structure follows the pattern /subscriptions/{id}/resourceGroups/{rg}/providers/{Namespace}/{type}/{name}. The ARM provider namespace and resource type hierarchy are defined and maintained by Microsoft; this specification does not alter or extend them. The template_format value for all Azure entries is azure_resource_id.

```
/subscriptions/{subscriptionId}/resourceGroups/{resourceGroup}/providers/Microsoft.Com
pute/virtualMachines/{name}
```

Figure 13: Azure CRIT template example

The namespace segment follows the Microsoft.{ServiceArea} convention documented in the ARM provider list. Resource path depth varies by resource type and is authoritative in the ARM REST API reference.

#### 12.6.3. Google Cloud (gcp)

GCP resources are identified by canonical resource names [GCP-ResourceName], the full-resource-name form used by Cloud Asset Inventory. The format prefixes the API service hostname with // followed by the resource path: //serviceName.googleapis.com/projects/{project}/.... The resource name structure is defined and maintained by Google; this specification does not alter or extend it. The template_format value for all GCP entries is gcp_resource_name.

```
//compute.googleapis.com/projects/{project}/zones/{location}/instances/{resource-id}
//storage.googleapis.com/projects/{project}/buckets/{resource-id}
```

Figure 14: GCP CRIT template examples

The API hostname prefix and resource path structure are published per-service in Google's API documentation. For inventory queries, {location} may be set to the wildcard value - to span all regions, as described in the Cloud Asset Inventory documentation.

#### 12.6.4. Cloudflare (cloudflare)

Cloudflare resources are identified using Cloudflare API locators [CF-API], a structured dotted-string form scoped to a Cloudflare account. All Cloudflare resources are globally scoped; the region_behavior for every Cloudflare entry is global-only. The template_format value for all Cloudflare entries is cloudflare_locator.

```
com.cloudflare.api.account.{account_id}.zone.{id}
```

Figure 15: Cloudflare CRIT template example

The resource type segment corresponds to the Cloudflare API object category as documented in the Cloudflare API reference. Zone-scoped resources share the same account-rooted prefix.

#### 12.6.5. Oracle Cloud Infrastructure (oracle)

OCI resources are identified by Oracle Cloud IDs (OCIDs) [OCI-OCID], structured opaque identifiers with realm and region components. Regional resources include the region segment; globally-scoped resources structurally omit both the region and availability domain segments, resulting in consecutive dots in the OCID. The OCID format is defined and maintained by Oracle; this specification does not alter or extend it. The template_format value for all OCI entries is oracle_ocid.

```
ocid1.instance.{realm=oc1}.{region}..{unique-id}
ocid1.compartment.{realm=oc1}...{unique-id}
```

Figure 16: OCI CRIT template examples

The realm component identifies the OCI infrastructure partition; oc1 is the commercial realm. The dot-separated positional structure is authoritative in the OCI Resource Identifiers documentation.

#### 12.6.6. Salesforce Platform (salesforce)

Salesforce resources are addressed by REST API resource URLs [SF-REST-API], the canonical programmatic address for every sObject record in a Salesforce org. Resources are org-scoped rather than region-scoped; the region_behavior for every Salesforce entry is global-only. The template_format value for all Salesforce entries is salesforce_url.

```
https://{instance}.salesforce.com/services/data/v{api-version}/sobjects/{object-type}/
{record-id}
```

Figure 17: Salesforce CRIT template example

The instance subdomain identifies the Salesforce org. The api-version slot follows the Salesforce seasonal release cadence (e.g., v61.0). The sObject name in the path identifies the resource type and corresponds to the resource_type field in the dictionary entry.

#### 12.6.7. SAP Business Technology Platform (sap)

SAP BTP resource URL conventions vary by product tier [SAP-BTP]: BTP core services use the BTP API endpoint form; ABAP Cloud systems use OData service paths; SAP SuccessFactors uses a dedicated API host pattern. The authoritative URL form for each service is published by SAP in its API Hub and product documentation. SAP does not publish a single unified resource identifier scheme; this specification encodes the per-tier URL patterns as CRIT templates without altering or extending the URL structures SAP defines.

```
https://api.{region}.hana.ondemand.com/resourcemanager/v1/instances/{resource-id}
https://{host}/sap/opu/odata/sap/{service-path}/{object-id}
https://api{api-server}.sapsf.com/odata/v2/{entity-set}('{resource-id}')
```

Figure 18: SAP CRIT template examples (one per product tier)

The product tier determines both the URL structure and the template_format value: sap_btp_url for BTP core services, sap_odata_url for ABAP Cloud and OData services, and sap_sf_url for SAP SuccessFactors. Implementers SHOULD consult the SAP API Hub entry for the specific SAP product to identify the canonical resource URL form before constructing dictionary entries.

#### 12.6.8. ServiceNow (servicenow)

ServiceNow resources are addressed by Table API URLs [SN-TABLE-API], the REST-addressable form of every Now Platform record. Resources are instance-scoped with no regional sub-division; the region_behavior for every ServiceNow entry is global-only. The template_format value for all ServiceNow entries is servicenow_table_url.

```
https://{instance}.service-now.com/api/now/table/{table-name}/{sys-id}
```

Figure 19: ServiceNow CRIT template example

The table-name slot is the ServiceNow database table name (e.g., incident, sys_script_include). The sys-id slot is the 32-character hex unique identifier assigned to every Now Platform record, as documented in the ServiceNow Table API reference.

## 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>.

[ISO8601] International Organization for Standardization, "Date and time -- Representations for information interchange", ISO 8601, 2019.

[AWS-ARN] Amazon Web Services, "Amazon Resource Names (ARNs)", 2024, <https://docs.aws.amazon.com/IAM/latest/UserGuide/reference-arns.html>.

[Azure-ResourceID] Microsoft, "Azure Resource Manager REST API Reference", 2024, <https://learn.microsoft.com/en-us/rest/api/resources/resources>.

[GCP-ResourceName] Google Cloud, "Resource Name Format -- Cloud Asset Inventory", 2024, <https://cloud.google.com/asset-inventory/docs/resource-name-format>.

[CF-API] Cloudflare, "Cloudflare API Reference", 2024, <https://developers.cloudflare.com/api/>.

[OCI-OCID] Oracle Cloud Infrastructure, "Resource Identifiers", 2024, <https://docs.oracle.com/en-us/iaas/Content/General/Concepts/identifiers.htm>.

[SF-REST-API] Salesforce, "Salesforce REST API Developer Guide", 2024, <https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/>.

[SAP-BTP] SAP SE, "SAP Business Technology Platform Documentation", 2024, <https://help.sap.com/docs/btp/sap-business-technology-platform/btp-api-overview>.

[SN-TABLE-API] ServiceNow, "ServiceNow Table API Reference", 2024, <https://developer.servicenow.com/dev.do#!/reference/api/latest/rest/c_TableAPI>.

## 13.2. Informative References

[CVEListv5] MITRE Corporation, "CVE List v5 -- CVE JSON 5.0 Schema", 2024, <https://github.com/CVEProject/cvelistV5>.

[OSV-Schema] OpenSSF Vulnerability Disclosures Working Group, "Open Source Vulnerability (OSV) Schema", 2024, <https://ossf.github.io/osv-schema/>.

[OpenVEX] OpenVEX, "OpenVEX Specification", 2023, <https://github.com/openvex/spec>.

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

[EPSS] Forum of Incident Response and Security Teams (FIRST), "Exploit Prediction Scoring System (EPSS)", 2024, <https://www.first.org/epss/>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012, <https://www.rfc-editor.org/rfc/rfc6570>.

[PURL] package-url, "Package URL (PURL) Specification", 2024, <https://github.com/package-url/purl-spec>.

[CPE23] Cheikes, B., Waltermire, D., and K. Scarfone, "Common Platform Enumeration: Naming Specification Version 2.3", NISTIR 7695, August 2011, <https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7695.pdf>.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, STD 68, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

## Appendix A. Complete CRIT Record Example -- AWS RDS MySQL (Informative)

The following is a complete CRIT record for a MySQL vulnerability affecting AWS RDS, illustrating the engine_version subschema with opt-in auto-upgrade and the deployed-before-fix problem.


```
{
  "vectorString": "CRITv0.3.0/CP:AW/VS:FX/FP:RR/SR:CA/RL:SC/EV:T/PP:1719792000/SA:1514
764800#CVE-2024-6387:ec2:instance",
  "vuln_id": "CVE-2024-20967",
  "provider": "aws",
  "service": "rds",
  "resource_type": "db",
  "resource_lifecycle": "stateful_managed",
  "shared_responsibility": "customer_action_required",
  "vex_status": "fixed",
  "template": "arn:aws:rds:{region}:{account}:db:{resource-id}",
  "template_format": "aws_arn",
  "temporal": {
    "service_available_date": "2013-09-18",
    "vulnerability_introduced_date": "2023-01-01",
    "vulnerability_introduced_date_estimated":
      true,
    "vuln_published_date": "2024-01-16",
    "provider_acknowledged_date": "2024-01-20",
    "provider_fix_date": "2024-02-15",
    "customer_deadline_date": "2024-04-15",
    "customer_deadline_source": "internal_policy"
  },
  "fix_propagation": "version_update",
  "existing_deployments_remain_vulnerable":
    true,
  "provider_fix_version": {
    "version_type": "engine_version",
    "comparison": "gte",
    "engine": "mysql",
```

```
      "version": "8.0.36",
      "auto_upgrade": false,
      "note": "auto_minor_version_upgrade must be enabled
        for automatic application."
    },
    "remediation_actions": [
      {
        "sequence": 1,
        "type": "version_update",
        "title": "Upgrade RDS MySQL engine to 8.0.36
          or later",
        "description": "aws rds modify-db-instance \
    --db-instance-identifier {resource-id} \
    --engine-version 8.0.36 --apply-immediately",
        "provider_guidance_url":
          "https://docs.aws.amazon.com/AmazonRDS/
          latest/UserGuide/USER_UpgradeDBInstance.MySQL.html",
        "auto_remediable": true,
        "requires_downtime": true,
        "stateful_impact": "backup_recommended",
        "estimated_downtime_range_seconds": { "min": 60, "max": 600 },
        "compensating_control": false
      }
    ],
    "detections": [
      {
        "provider": "aws",
        "service": "config_rule",
        "query_language": "cloudwatch_filter",
        "query": "{ ($.eventName = ModifyDBInstance) &&
          ($.requestParameters.engineVersion < \"8.0.36\") }",
        "detection_phase": "misconfiguration",
        "description": "Detects when an RDS instance is modified to
          a MySQL version below the fix threshold."
      }
    ],
    "provider_advisory": {
      "advisory_id": "ALAS2-2024-2489",
      "advisory_url": "https://alas.aws.amazon.com/AL2/ALAS-2024.html"
    }
  }
```

Figure 20: AWS RDS MySQL CRIT Record

## Appendix B. Open Issues (Informative)

The following issues require design decisions prior to a stable v1.0 release:

1. *vulnerability_introduced_date sourcing:* What is the authoritative source? NVD, provider advisory, Git commit history, or producer-derived analysis? A structured date_provenance field may be warranted.

2. *Detection query versioning:* Cloud provider query languages and log schemas evolve. A query_language_version field on detection entries would allow consumers to detect stale queries.

3. *rolling_replace fleet progress tracking:* A structured fleet_remediation_event object may be needed for cross-consumer interoperability.

4. *Oracle realm handling:* Government OCI realms (oc2, oc3) may have different fix availability timelines. Confirm whether realm_overrides array within a single record is preferable to separate records.

5. *CVSS version discrimination:* A cvss_version discriminator on provider_advisory would allow consumers to apply version-appropriate scoring logic.

## Acknowledgements

The author thanks the Wiz security research team for open-sourcing their cloud vulnerability database work, the Anchore team for open-sourcing their CVE enrichment work, the CVEProject and OpenSSF communities for the ADP container and OSV schema mechanisms that make upstream integration possible, and the OWASP community for providing a home for applied security standards work.

## Author’s Address

CD Langton
Vulnetix
Australia
Email: chris@vulnetix.com
URI: https://www.vulnetix.com