ESIP Free Tier

Consumer Guide

Version: 1.8 · Status: Current · ZenzizenSec · May 2026

Applies to: public.exposuresignal.io · github.com/jonathanristo/ESIP-public
Schema version: 1.0
Classification: ZenzizenSec / External

---

1. What the Free Tier Is

ESIP's free tier gives you direct visibility into real-world exploitation evidence for CVEs, without requiring access to your environment. It exposes raw attacker activity signals. It does not provide interpretation or prioritization.

The free tier tells you what is happening. The commercial tier tells you what matters and what to do about it.

Free tier = evidence. Commercial tier = decision support.

Why the Free Tier Exists

The free tier exists to let you work with ESIP data before committing to the full platform. It allows you to:

- Validate ESIP data against your existing vulnerability intelligence
- Build integrations using a stable, versioned public dataset
- Understand how exploitation evidence emerges and changes over time
- Evaluate how ESIP data fits into your existing workflows before adopting the commercial tier

The free tier provides the observable evidence layer. When you need prioritization, lifecycle tracking, and decision support, the commercial tier provides the derived intelligence layer.

ESIP does not know your environment. All prioritization decisions require your asset context. ESIP supplies the global threat half of that equation.

Who This Guide Is For

This guide is intended for:

- Security engineers evaluating ESIP data
- Developers building integrations
- Researchers analysing vulnerability trends
- Organisations validating ESIP before adopting the commercial tier

This guide assumes familiarity with the ESIP Overview. If you have not read it, start there to understand how exposure signals differ from traditional vulnerability data.

> **Free Tier Facts-Only Guarantee**: The free tier surfaces observable evidence only. The following fields will never appear in any free-tier API response or GitHub snapshot under schema version 1.0:
>
> `severity_score` · `risk_score` · `signal_state` · `lifecycle_stage` · `exploitation_maturity`
>
> `remediation_urgency` · `confidence` · `weight` · `trust_tier` · `source_reliability` · `decay_factor`
>
> Also absent: EPSS percentile, observation score impact, source trust tier, internal scoring components.
>
> This is a contractual boundary enforced at the API level. It will not change without a schema version update. If you need any of these fields, the commercial API at `api.exposuresignal.io` is the correct tier. See Section 12.

What This Means in Practice

The free tier shows you that exploitation evidence exists. It does not tell you:

- How urgent this CVE is relative to others in your environment
- Whether the risk is increasing, confirmed, or declining
- How confident the system is in the evidence
- How to prioritise it within your remediation workflow

You can build this logic yourself using the free-tier data. The commercial tier eliminates that burden and provides it directly.

Free Tier vs Commercial Tier: A Concrete Example

Free tier response:

```
{
  "observation_events": [
    { "observation_type": "kev_inclusion" },
    { "observation_type": "exploit_reference" }
  ]
}
```

This tells you exploitation evidence exists, but not how to act on it.

Commercial tier response for the same CVE:

```
{
  "verdict": {
    "severity_band": "Critical",
    "lifecycle_stage": "Confirmed",
    "confidence": "High",
    "severity_score": 94
  },
  "evidence": {
    "what_changed": "Added to KEV with active exploit tooling confirmed in the
wild"
  }
}
```

This tells you this CVE is one of your highest priorities right now.

This is the difference between knowing exploitation exists and knowing it is your highest priority.

---

2. Authentication

The public API requires an `X-API-Key` header on every request. The GitHub snapshot requires no authentication. It is an anonymous public repository.

Getting a Key

Register at `www.zenzizensec.com/esip`. The website handles registration and issues your key. Once issued, the key is shown once. Store it immediately as it cannot be retrieved again.

Using Your Key

Pass your key in the `X-API-Key` header. The query string is explicitly rejected to prevent key leakage into URL access logs, browser history, Referer headers, and CDN logs.

```
# Correct: key in header
curl https://public.exposuresignal.io/v1/cves/CVE-2021-44228 \
  -H "X-API-Key: your_key_here"

# Wrong: query string rejected with 400
# curl ".../v1/cves/CVE-2021-44228?api_key=..."  ← never do this
```

401 Responses

The API returns the same 401 message for all authentication failures. It does not reveal whether a key exists. Revoked keys are indistinguishable from non-existent keys.

Situation	Response	What to do
No X-API-Key header	401	Add the header
Invalid or malformed key	401	Check the key value. Must be passed verbatim as issued
Key has been revoked	401 (identical message)	Register for a new key at www.zenzizensec.com/esip
Key not found	401 (identical message)	Verify you are using the key from your registration confirmation

Do not retry 401 responses. Authentication failures will not resolve without correcting the API key. Retrying the same request will return 401 indefinitely.

Rate Limits

Rate limits are per key, applied across all instances:

Limit	Value
Requests per minute	300
Requests per second	10

Every response, successful or rate-limited, includes rate limit headers:

```
X-RateLimit-Limit: 300
X-RateLimit-Remaining: 247
X-RateLimit-Reset: 1746432000
Retry-After: 12          # only on 429
```

Check `X-RateLimit-Remaining` on every response to track your budget. When you receive a 429, wait the number of seconds in `Retry-After` before retrying. `X-RateLimit-Reset` is a Unix timestamp marking when the window resets.

Python Sample with Rate Limit Handling

```
import requests, time

API_KEY = "your_key_here"
BASE    = "https://public.exposuresignal.io"
HEADERS = {"X-API-Key": API_KEY}

def get_cve(cve_id):
    resp = requests.get(f"{BASE}/v1/cves/{cve_id}", headers=HEADERS)
    if resp.status_code == 404:
        return None
    if resp.status_code == 429:
        wait = int(resp.headers.get("Retry-After", 60))
        print(f"Rate limited, waiting {wait}s")
        time.sleep(wait)
        return get_cve(cve_id)  # retry once
    if resp.status_code == 401:
        raise RuntimeError("Auth failed. Check key or re-register")
    resp.raise_for_status()
    remaining = resp.headers.get("X-RateLimit-Remaining")
    print(f"Rate budget remaining: {remaining}")
    return resp.json()["data"]
```

---

3. Two Access Paths

The same data is available through two surfaces. The per-CVE record shape is identical on both. Switching between them requires no code changes on your record-parsing logic. The outer envelope differs; see Sections 4 and 5.

	Public API	GitHub Snapshot
URL	`public.exposuresignal.io`	`github.com/jonathanristo/ESIP-public`
Best for	Live lookups, single CVE queries, real-time integration	Batch processing, offline analysis, reproducible research, immutable archive trail
Authentication	X-API-Key header required. Get key at www.zenzizensec.com/esip	None. Public GitHub repo, anonymous read
Freshness	Current published snapshot	Updated once per UTC day
Format	Paginated JSON via HTTP	Flat JSON file, no pagination
Provenance	content_hash in meta envelope	SHA-256 the file bytes and compare against API meta

Use the API when you need to look up a specific CVE right now, or want to integrate ESIP signals into a live workflow.

Use the GitHub snapshot when you are processing the full dataset, building offline analysis, need a reproducible point-in-time view, or want an immutable archive trail without depending on API availability.

---

4. Public API

`https://public.exposuresignal.io`

Requires `X-API-Key` header on every request. Get a key at `www.zenzizensec.com/esip`. Page size fixed at 100. No `?page_size=` parameter.

Endpoints

Method	Path	Purpose
GET	`/v1/cves/{cve_id}`	Single CVE record, full field set
GET	`/v1/cves?page=N`	Paginated list: cve_id and exposure_class_id only

Single CVE: GET /v1/cves/{cve_id}

```
curl https://public.exposuresignal.io/v1/cves/CVE-2021-44228 \
  -H "X-API-Key: your_key_here"

{
  "data": {
    "cve_id": "CVE-2021-44228",
    "exposure_class_id": "remote-code-execution-jvm",
    "attack_techniques": [
      {
        "technique_id": "T1190",
        "technique_name": "Exploit Public-Facing Application",
        "mapping_confidence": "high",
        "mapping_source": "esip"
      }
    ],
    "observation_events": [
      {
        "observation_type": "kev_inclusion",
        "source_name": "CISA KEV",
        "observed_at": "2021-12-10T00:00:00Z"
      }
    ],
    "detection_capability": {
      "public_detection_available": true,
      "detection_source_count": 1,
      "detection_sources": [
        {
          "source_name": "nuclei",
          "artifact_type": "template",
          "artifact_id": "http/cves/2021/CVE-2021-44228.yaml",
          "first_seen_at": "2021-12-10T00:00:00Z"
        }
      ]
    }
  },
  "meta": {
```

```
    "schema_version": "1.0",
    "snapshot_generated_at": "2026-05-05T08:00:00Z",
    "publication_identity": {
      "content_hash": "sha256-abc123...",
      "snapshot_generated_at": "2026-05-05T08:00:00Z",
      "git_commit_sha": "abc123...",
      "schema_version": "1.0"
    },
    "request_id": "uuid-per-request",
    "attribution": "Data provided by ESIP / ZenzizenSec — www.zenzizensec.com",
    "license_url":
"https://github.com/jonathanristo/ESIP/blob/main/LICENSE_DATA.md"
  }
}
```

Note: `detection_capability` may be `null` for most CVEs. Always check before accessing nested fields. See Section 5.

Paginated List: GET /v1/cves?page=N

Returns `cve_id` and `exposure_class_id` only. Use to enumerate the published population, then fetch individual records as needed.

```
{
  "data": [
    { "cve_id": "CVE-1999-0001", "exposure_class_id": "..." },
    { "cve_id": "CVE-1999-0002", "exposure_class_id": "..." }
  ],
  "pagination": {
    "page": 1,
    "page_size": 100,
    "total_pages": 285,
    "total_cves": 28507,
    "next_page": 2,
    "prev_page": null
  },
  "meta": { ... }
}
```

> **Ordering gotcha: read this before iterating.** List results are ordered lexicographically on `cve_id`: string sort, not numeric. `CVE-1999-12345` sorts before `CVE-1999-9999` because `"1" < "9"` in string comparison. Do not use position in the list to infer recency or numeric CVE sequence. Treat the order as stable but opaque.

Error Responses

Status	When	What to do
`400`	Invalid cve_id format (must match CVE-d{4}-d{4,}) or page < 1	Fix the request
`404`	CVE not in current snapshot	CVE has no observable signal. Expected, not a bug. See FAQ Section 10.
`503 + Retry-After: 3600`	Cache cold; publication cycle not yet complete	Wait one hour and retry. Use exponential backoff if polling at startup.

Sample Code

curl: single lookup

```
# Single CVE lookup
curl https://public.exposuresignal.io/v1/cves/CVE-2021-44228 \
  -H "X-API-Key: your_key_here"

# List page 1
curl "https://public.exposuresignal.io/v1/cves?page=1" \
  -H "X-API-Key: your_key_here"
```

Python: iterate all records with cold-start retry

```
import requests, time

BASE = "https://public.exposuresignal.io"

def fetch_with_retry(url, params=None, headers=None, retries=3):
    for attempt in range(retries):
        resp = requests.get(url, params=params, headers=headers or {})
        if resp.status_code == 503:
            wait = int(resp.headers.get("Retry-After", 3600))
            print(f"Cold start; retrying in {wait}s")
            time.sleep(wait)
            continue
        resp.raise_for_status()
        return resp.json()
    raise RuntimeError("Max retries exceeded")

def get_all_cves():
    page, results = 1, []
    while True:
        data = fetch_with_retry(f"{BASE}/v1/cves", {"page": page}, headers=HEADERS)
        results.extend(data["data"])
        if data["pagination"]["next_page"] is None:
            break
        page = data["pagination"]["next_page"]
    return results

def get_cve(cve_id):
    """Returns None on 404. Absence is accurate, not a bug."""
    resp = requests.get(f"{BASE}/v1/cves/{cve_id}")
    if resp.status_code == 404:
        return None
    resp.raise_for_status()
    record = resp.json()["data"]
    # detection_capability may be null; always guard
    detection = record.get("detection_capability")  # may be None
    has_detection = detection and detection["public_detection_available"]
    return record
```

JavaScript (fetch): single lookup with null guard

```
const API_KEY = "your_key_here";
const BASE    = "https://public.exposuresignal.io";
const HEADERS = { "X-API-Key": API_KEY };

async function getCve(cveId) {
  const resp = await fetch(`${BASE}/v1/cves/${cveId}`, { headers: HEADERS });
  if (resp.status === 404) return null;  // absent = no signal, not an error
  if (!resp.ok) throw new Error(`HTTP ${resp.status}`);
  const body = await resp.json();
  const record = body.data;
```

```
  // detection_capability may be null; guard before accessing
  const detection = record.detection_capability ?? null;
  const hasDetection = detection?.public_detection_available ?? false;
  return record;
}

async function getAllCves() {
  let page = 1;
  const results = [];
  while (true) {
    const resp = await fetch(`${BASE}/v1/cves?page=${page}`, { headers: HEADERS });
    if (resp.status === 503) {
      const wait = parseInt(resp.headers.get("Retry-After") ?? "3600", 10);
      await new Promise(r => setTimeout(r, wait * 1000));
      continue;
    }
    const body = await resp.json();
    results.push(...body.data);
    if (body.pagination.next_page === null) break;
    page = body.pagination.next_page;
  }
  return results;
}
```

---

5. GitHub Snapshot

`https://github.com/jonathanristo/ESIP-public`

No authentication required. The GitHub snapshot is an anonymous public repository. Anyone can pull it without a key. If you want to use ESIP data without registering, this is the path.

Files

Path	Behaviour
`public/snapshot_latest.json`	Overwritten on every successful daily publication cycle
`public/snapshots/snapshot_YYYYMMDD.json`	Immutable. One file per UTC day, never overwritten

Use `snapshot_latest.json` for the most recent data. Use dated archive files for reproducible research, auditing, or point-in-time comparison.

Snapshot Payload Shape

```
{
  "snapshot_generated_at": "2026-05-05T08:00:00Z",
  "schema_version": "1.0",
  "cve_count": 28507,
  "dataset_scope": "signal_set_only",
  "dataset_type": "public_snapshot",
  "record_definition": "CVE with at least one observable signal",
  "attribution": "Data provided by ESIP / ZenzizenSec — www.zenzizensec.com",
  "license_url": "https://github.com/jonathanristo/ESIP/blob/main/LICENSE_DATA.md",
  "data": [
    {
      "cve_id": "CVE-2021-44228",
      "exposure_class_id": "...",
```

```
      "attack_techniques": [ ... ],
      "observation_events": [ ... ],
      "detection_capability": { ... }  // or null
    }
  ]
}
```

The `data` array contains every published CVE in a single flat list, with no pagination. There is no per-record `meta` envelope.

Verifying Snapshot Integrity

The API's `meta.publication_identity.content_hash` is the SHA-256 hash of the GitHub snapshot file's bytes. This lets you prove the API and snapshot are from the same publication cycle, useful when comparing outputs from both surfaces or auditing a point-in-time view.

```
import hashlib, requests

snap_bytes = requests.get(
    "https://raw.githubusercontent.com/jonathanristo/"
    "ESIP-public/main/public/snapshot_latest.json"
).content

computed = hashlib.sha256(snap_bytes).hexdigest()

api = requests.get(
    "https://public.exposuresignal.io/v1/cves/CVE-2021-44228"
).json()
api_hash = api["meta"]["publication_identity"]["content_hash"]

assert computed == api_hash, "API and snapshot are from different publication
cycles"
```

Python: Parse the Snapshot

```
import requests

URL = ("https://raw.githubusercontent.com/jonathanristo/"
       "ESIP-public/main/public/snapshot_latest.json")

data = requests.get(URL).json()
print(f"Schema:    {data['schema_version']}")
print(f"Generated: {data['snapshot_generated_at']}")
print(f"CVEs:      {data['cve_count']}")

for record in data["data"]:
    has_kev = any(
        e["observation_type"] == "kev_inclusion"
        for e in record["observation_events"]
    )
    techniques = [t["technique_id"] for t in record["attack_techniques"]]
    # detection_capability may be None; guard before accessing
    detection = record.get("detection_capability")
    has_nuclei = bool(detection and detection["public_detection_available"])
```

---

6. Field Reference

Every field is present on both the public API and the GitHub snapshot under the same name. Fields marked optional may be `null` or absent.

cve_id

Standard NVD CVE identifier. Format: `CVE-YYYY-NNNNN`. Always present.

exposure_class_id

The CVE's mapped exposure class: a string identifier representing the vulnerability's category. Signals and ATT&CK mappings flow through this class. Multiple CVEs sharing the same `exposure_class_id` receive the same technique list.

observation_events

Observable evidence recorded for this CVE. Array; may be empty. Three `observation_type` values are surfaced on the free tier:

observation_type	Meaning	Example source_name
`kev_inclusion`	CISA has added this CVE to the Known Exploited Vulnerabilities catalogue	"CISA KEV"
`exploit_reference`	A weaponized exploit module, PoC, or tool exists publicly	"ExploitDB", "Metasploit", "GitHub PoC"
`patch_available`	A patch or remediation is available	NVD CPE-derived references

EPSS score and external prevalence metrics are not surfaced on the free tier.

attack_techniques

ATT&CK technique mappings for this CVE's exposure class. Array; may be empty. Mappings attach to the `exposure_class_id`. All CVEs sharing an exposure class receive the same technique list.

Field	Type	Description
`technique_id`	string	MITRE ATT&CK technique ID (e.g. T1190)
`technique_name`	string	Human-readable technique name
`mapping_confidence`	enum	`Low` · `Medium` · `High` · `Verified`
`mapping_source`	enum	`esip` (ESIP-curated) or `cve2capec` (reference-grade)

Note: `mapping_confidence` describes how well the CWE-to-technique mapping is evidenced. It is not a scoring contribution. The free tier does not expose how mappings contribute to signal weights.

detection_capability (optional)

Public detection tooling known for this CVE, drawn from publicly available tooling. Detection sources listed here do not represent a complete detection capability model. May be `null`. This is the expected state for most CVEs. Always null-check before accessing.

Field	Type	Description
`public_detection_available`	boolean	true if at least one detection source is active
`detection_source_count`	integer	Number of active detection sources
`detection_sources`	array	One entry per source

Each `detection_sources` entry:

Field	Description
`source_name`	Tool name (e.g. nuclei)
`artifact_type`	Type of detection artifact (e.g. template)
`artifact_id`	Identifier or path to the specific artifact
`first_seen_at`	ISO 8601 timestamp: when ESIP first observed this artifact

---

7. Fields That Will Never Appear

The following are guaranteed absent from all free-tier responses under schema version 1.0. This list is safe to include in procurement and compliance documentation.

Field	Available in paid tier?
`severity_score, risk_score`	Yes. Computed band and raw score
`lifecycle_stage, signal_state`	Yes. Emerging / Active / Confirmed / Declining / Dormant / Remediation_Available
`exploitation_maturity`	Yes
`remediation_urgency`	Yes
`confidence`	Yes. Low / Medium / High / Verified
`weight, decay_factor`	Yes. Evidence age weighting
`trust_tier, source_reliability`	Yes. Source quality indicators
EPSS percentile	Yes. Exploit probability score from FIRST.org
Observation score impact, rule IDs	Internal. Not exposed on any tier

---

8. Publication Cadence

The snapshot is published once per UTC day. Publication timing within the day is not guaranteed.

When the API returns `503` with `Retry-After: 3600`, the publication cycle has not yet completed for the day. Wait the indicated duration before retrying. The Python and JavaScript samples in Section 4 show the retry pattern.

The GitHub `snapshot_latest.json` always contains the most recent successful cycle. If the API returns 503, the GitHub snapshot is the correct fallback for batch consumers.

---

9. Schema Versioning

Current schema version: `"1.0"`. The version string appears in three places: `meta.schema_version`, `meta.publication_identity.schema_version`, and the GitHub snapshot's top-level `schema_version` field. All three always match.

Change type	Version impact	Notice
New optional field added	No bump	None required. Existing consumers unaffected
Field removed, type changed, semantic change	Major version bump	Announced via GitHub Releases on the ESIP-public repo before deployment

Where to watch for changes: GitHub Releases page at `github.com/jonathanristo/ESIP-public/releases`. Breaking changes will be published as a release note before the schema version is incremented. Watch the repo (GitHub Notifications → Releases) to receive advance notice automatically.

---

10. Frequently Asked Questions

Why isn't my CVE in the snapshot?

ESIP only publishes CVEs where at least one observable signal exists. The `record_definition` field on every snapshot confirms this: `"CVE with at least one observable signal"`.

A 404 from the API and absence from the snapshot both mean the same thing: ESIP has not observed globally measurable evidence (KEV inclusion, a weaponized exploit reference, or a patch reference) for that CVE. This is accurate, not a gap. The CVE exists. It simply has no signal. It may acquire one later.

Why does CVE-1999-12345 sort before CVE-1999-9999?

The list endpoint orders results lexicographically on `cve_id` as a string, not as a number. In string sort, `"1" < "9"`, so `CVE-1999-12345` comes before `CVE-1999-9999`. This ordering is stable across pages, but do not use it to infer recency, sequence, or numeric CVE ID order.

Why does detection_capability come back null?

A null value means no public detection tooling is currently known for this CVE. This is the expected state for most CVEs. Detection templates take time to author and publish after a CVE is disclosed. A null here is not an error; it means ESIP has no active detection source record for this CVE at the time of the last snapshot.

Can I redistribute the data or include it in my product?

Redistribution and commercial use are governed by the ESIP Data Licence at the `license_url` in every response. The required attribution string must be preserved in any downstream use or redistribution. If you are building a commercial product that resells or re-packages ESIP data as part of its value proposition, contact ZenzizenSec before publishing. Commercial embedding of the free-tier data has different terms from personal or research use.

Can I use the free tier in production?

Yes. The free tier has no authentication requirement and is available at the same uptime target as the commercial API. If your use case requires an SLA, priority support, or real-time signals ahead of the daily publication cadence, the commercial API is the correct tier.

---

11. Attribution and Licence

All data published by ESIP under this schema is subject to the ESIP Data Licence. The required attribution string and licence URL are present in every API response and snapshot payload.

```
Data provided by ESIP / ZenzizenSec — www.zenzizensec.com
Licence: https://github.com/jonathanristo/ESIP/blob/main/LICENSE_DATA.md
```

Attribution must be preserved in any downstream use, transformation, or redistribution of this data. Attribution survives transformation. If you convert the JSON to another format, integrate it into a database, or surface it in a UI, the attribution requirement still applies.

Redistribution rules: Personal use, research, and internal tooling are permitted under the free-tier licence. Redistribution in a product or service that makes the data available to third parties requires that the attribution and licence URL accompany the data at the point of access.

Commercial embedding: If ESIP data is a material component of a commercial product, for example a vulnerability management platform, MSSP service, or security tool that surfaces ESIP signals to paying customers: contact ZenzizenSec at `www.zenzizensec.com` before publishing. Commercial embedding has different terms from research or personal use.

---

12. When to Upgrade to the Commercial Tier

Move to the commercial tier when you need to make decisions, not just observe evidence. The triggers below are good indicators:

- The commercial tier adds severity scoring and band classification so you know which CVEs demand action first.
- Signals progress through stages (Emerging → Active → Confirmed → Declining → Dormant). The commercial tier surfaces this. The free tier does not.
- The `/v1/signals/changes` endpoint returns only what changed since your last poll. No custom diff logic required.
- The commercial tier tells you how well-evidenced each signal is and how evidence age affects the verdict.
- Free tier does not include EPSS percentile. The commercial tier adds exploit probability and tracks whether it is rising.
- The free tier is provided as-is. The commercial tier includes operational guarantees and priority support.

Contact ZenzizenSec at `www.zenzizensec.com` to request access. The full field contract for the commercial API is documented in the ESIP Output Contract, provided to commercial customers under their agreement.