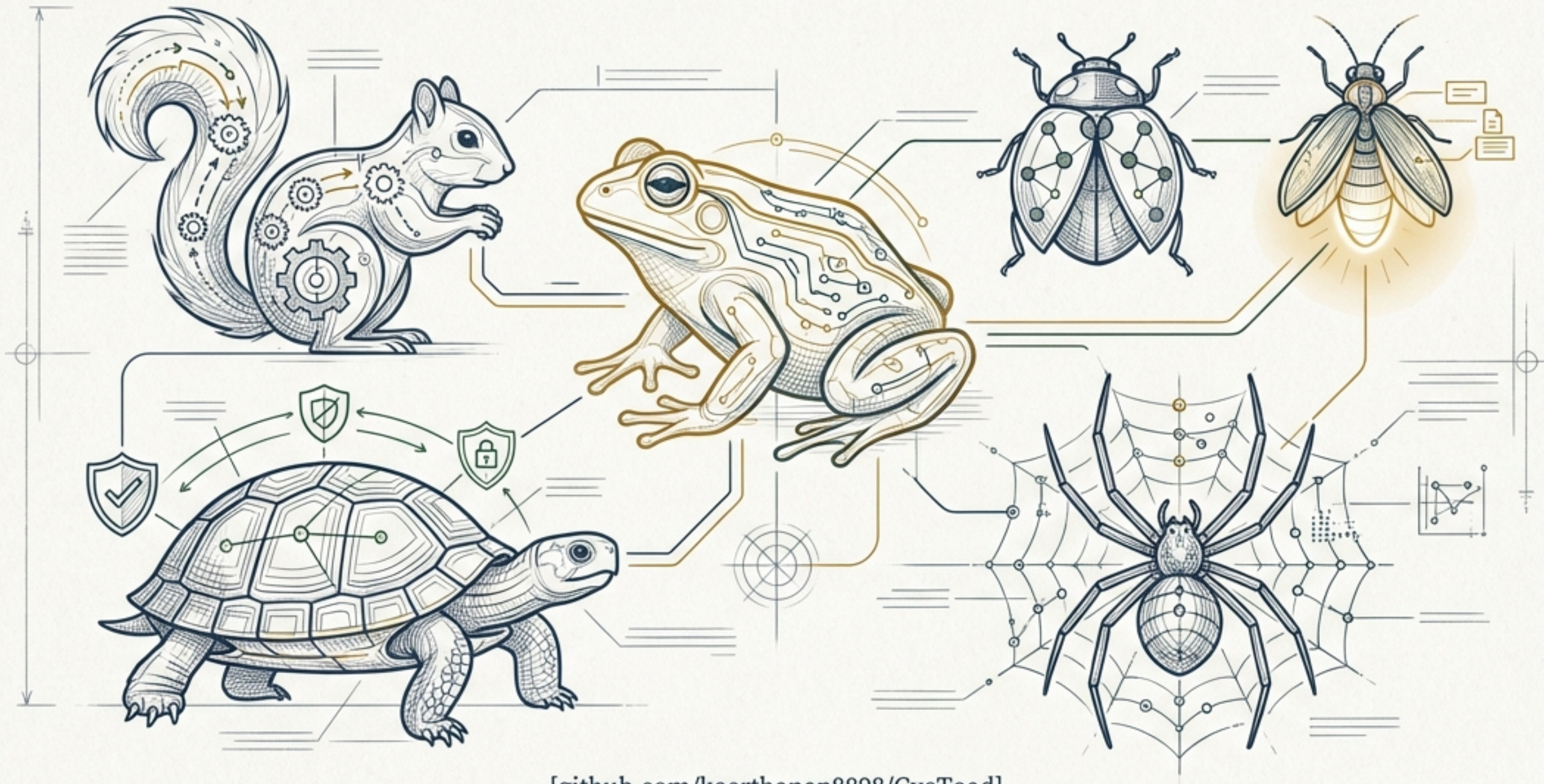


CVE Keeper

A New System for Automated, High-Fidelity Vulnerability Intelligence



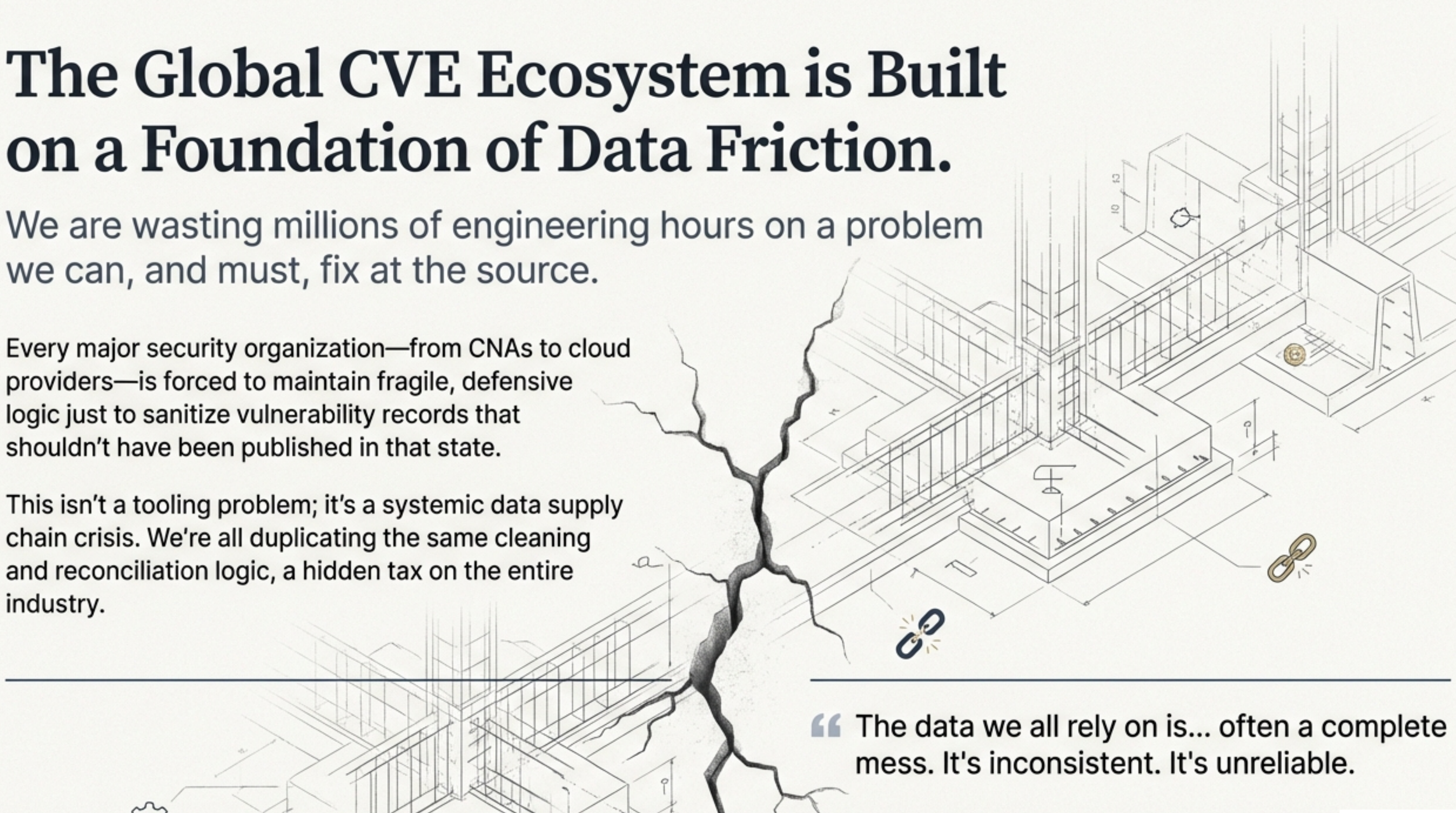


[github.com/keerthanap8898/CveToad]

The Global CVE Ecosystem is Built on a Foundation of Data Friction.

We are wasting millions of engineering hours on a problem we can, and must, fix at the source.

Every major security organization—from CNAs to cloud providers—is forced to maintain fragile, defensive logic just to sanitize vulnerability records that shouldn't have been published in that state.

This isn't a tooling problem; it's a systemic data supply chain crisis. We're all duplicating the same cleaning and reconciliation logic, a hidden tax on the entire industry.



“ The data we all rely on is... often a complete mess. It's inconsistent. It's unreliable.

The Cyber-Fauna Ecosystem: A Workflow for Automated Vulnerability Intelligence



CveSquirrel

A data-ingestion tool that collects all relevant CVE metadata, package metadata, and system data required for downstream analysis



CveLadybug

Spots anomalies or bugs in CVE templates across sources, compiling data to validate impact and provide context for automated CVSS estimation



Turtilla / TurtlePod

A minimal, non-CVE template app that authenticates & orchestrates containerized, sandboxed Gen-AI chatbot shells for diverse workflows



CveToad

Hops onto an OS to detect & validate active bugs/CVEs, coupling automation with manual verification to minimize threat modeling complexity.

Depends on inputs from Turtilla, CveLadybug, and CveSquirrel.



CveFirefly

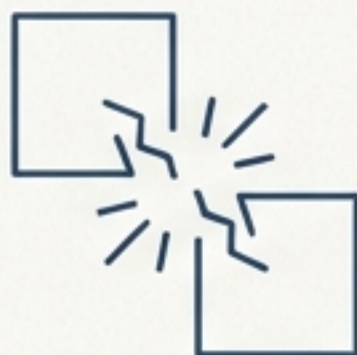
Automates and runs open-source AI penetration testing tools to perform active security assessments on the curated and verified CVE data from CveToad.



CveArachnid

A minimalist, non-CVE telemetry and dashboarding template app that enables correctness and validation steps as a foundational observability layer.

The Anatomy of Chaos: Recurring Nightmares in the Data Supply Chain



Structural & Semantic Errors

Non-standard JSON structures and malformed fields break automation.

Descriptions embedding entire commit logs or raw HTML tags, forcing manual parsing.

A fix for CVE-2023-0001 was applied in commit `<[^>]+>` to address XSS vulnerability.

Inconsistent package naming that requires fragile, fuzzy matching instead of canonical identifiers.



Conflicting Metrics & Scores

Divergent CVSS scores for the same CVE force constant manual arbitration and erode trust in risk models.

Cross-source **baseScore** deviations signal significant interpretation differences between a CNA and an ADP.

$$\Delta \geq 1.5$$



Missing & Ambiguous Classifications

Vague or missing data creates tracking confusion and undermines automated analysis.

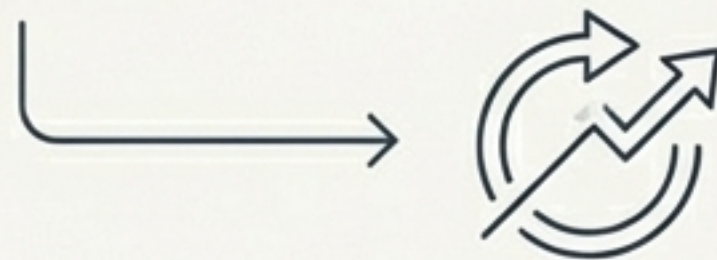
Persistent 'NVD-CWE-noinfo' or 'CWE-Other' entries that lack the specificity needed for effective mitigation.

A Framework for Systemic Improvement: Value for Every Stakeholder

MITRE | **CVE.org**

For CNAs & Standards Bodies

Enables high-quality, actionable feedback to resolve ambiguities like "CWE-Other" at the source. The system generates upstream data quality recommendations and CNA quality rankings, fostering a self-reinforcing cycle of improvement.

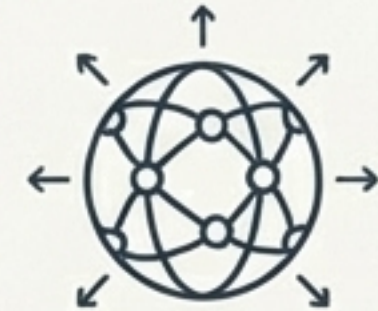
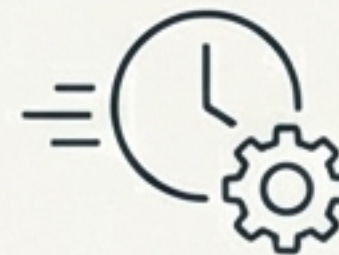


Google  Red Hat 

For Platforms & Enterprises

Drastically reduces the "hidden tax" of duplicated data cleaning efforts. Shrinks diagnosis time from weeks to hours by providing system-specific, dependency-aware impact analysis.

Optimizes patching and reboot frequency, reducing operational disruption.



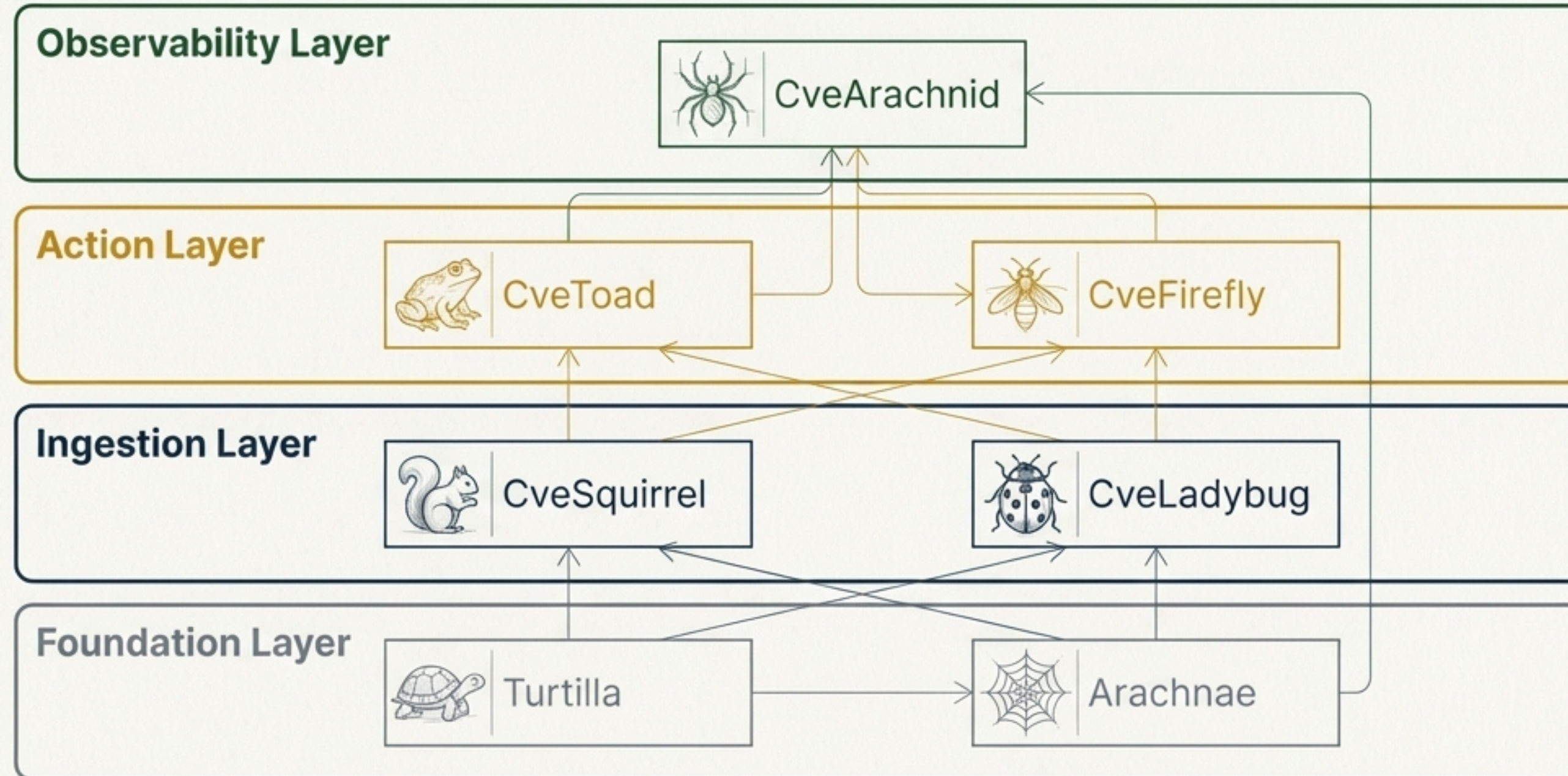
For the Global Community & Open Source

Makes high-quality, context-aware vulnerability management accessible beyond large enterprises.

Fosters community intelligence by creating a framework for shared, validated impact assessments.

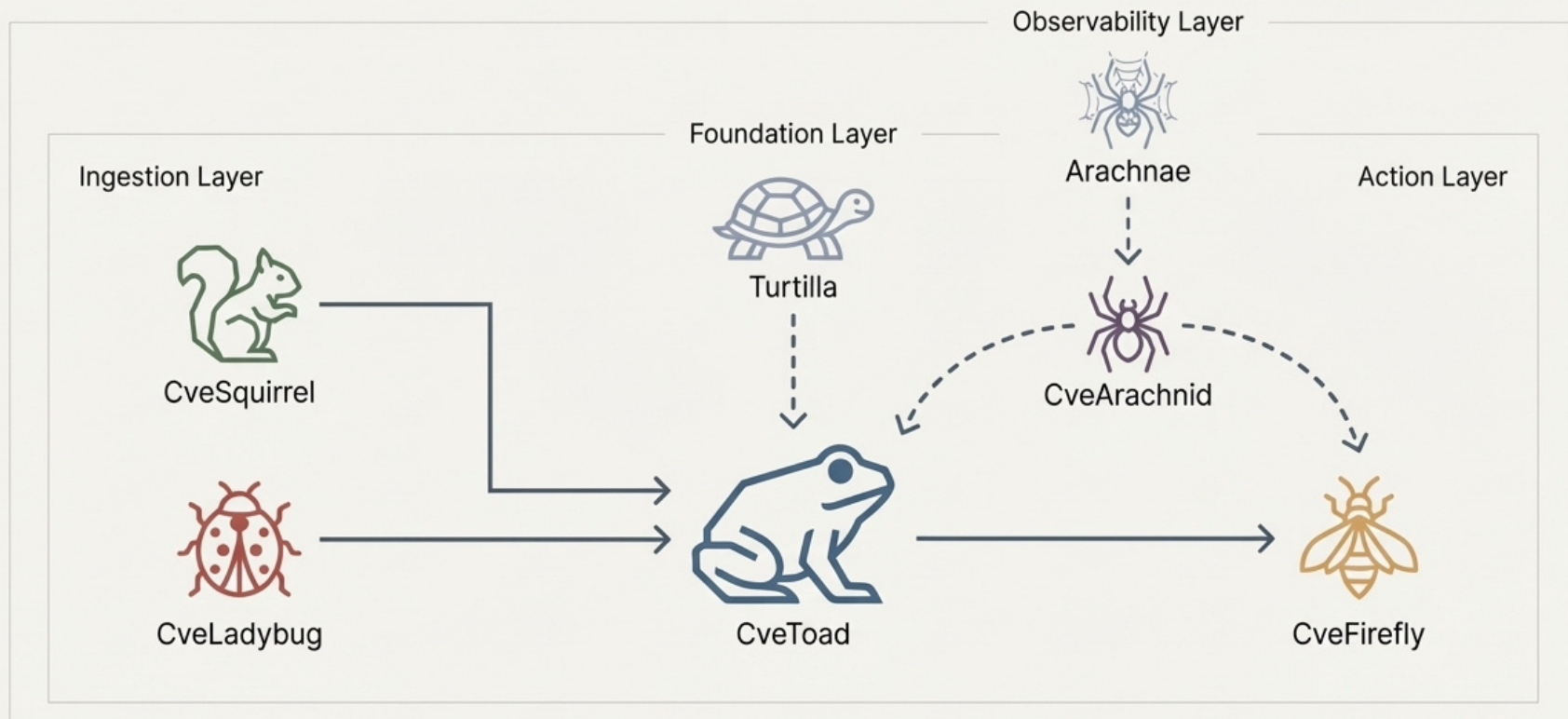


Our Solution: An Ecosystem for Clarity and Trust



A suite of specialised, interoperable tools designed to ingest, validate, analyse, and monitor CVE data through its entire lifecycle. Each component addresses a specific failure in the traditional data supply chain.

Our Solution: An Ecosystem for Clarity and Trust



A suite of specialized, interoperable tools designed to ingest, validate, analyze, and monitor CVE data through its entire lifecycle. Each component addresses a specific failure in the traditional data supply chain.

The Foundational Templates: Enabling Modular Design



Turtilla / TurtlePod: *The Secure Shell Foundation*

A minimal template app that authenticates & orchestrates containerized, full-stack multi-LLM access shells. It provides the sandboxed Gen-AI chatbot environment that **CveToad** depends on, prioritizing memory safety & low latency.

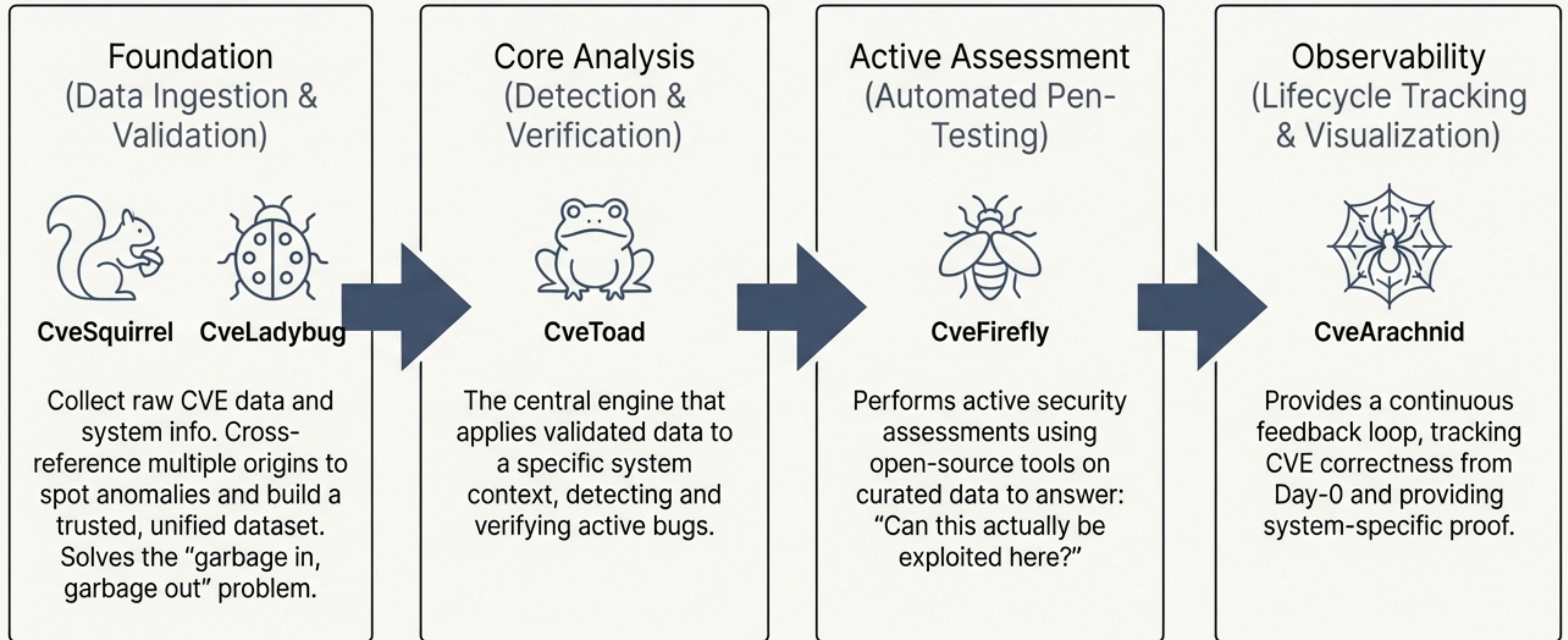


Arachnae: *The Observability Foundation*

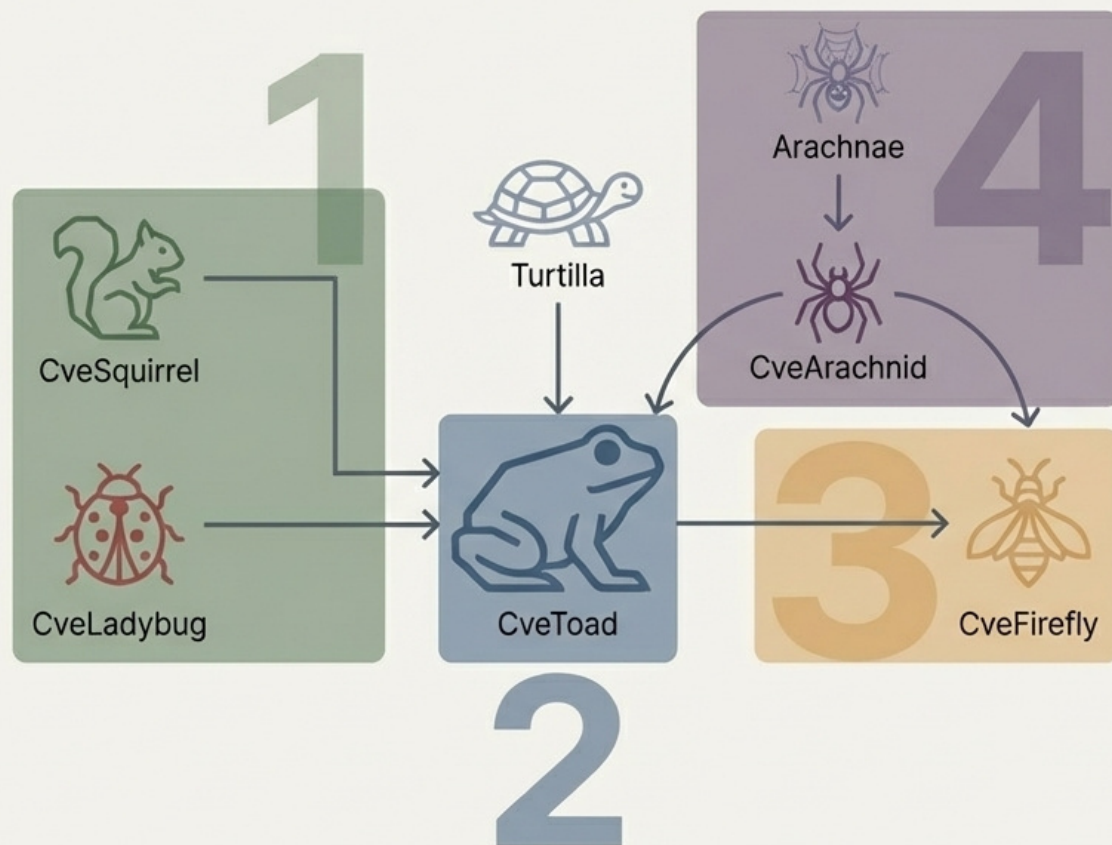
A minimalist telemetry & dashboarding template app. It provides the foundational observability layer—enabling correctness & validation steps—that **CveArachnid** is built upon.

Design Principle: Building specialized tools on hardened, reusable templates accelerates development and ensures consistency.

The Lifecycle of a Vulnerability: From Ingestion to Proof



The Lifecycle of a Vulnerability: From Ingestion to Proof



Phase 1: Foundation (Data Ingestion & Validation)

Function: Collect raw CVE data, package metadata, and system info. Cross-reference and validate data from multiple origins to spot anomalies and build a trusted, unified dataset.

Phase 2: Core Analysis (Detection & Verification)

Function: The central engine that detects and validates active bugs on a target OS, coupling automation with key manual verification points.

Phase 3: Active Assessment (Automated Pen-Testing)

Function: Performs active security assessments using OSS tools on the curated and verified CVE data from CveToad.

Phase 4: Observability (Lifecycle Tracking & Visualization)

Function: Provides a continuous feedback loop, running real-time tests to track CVE correctness and provide visualization and proof.

Phase 1: Building a Foundation of Truth



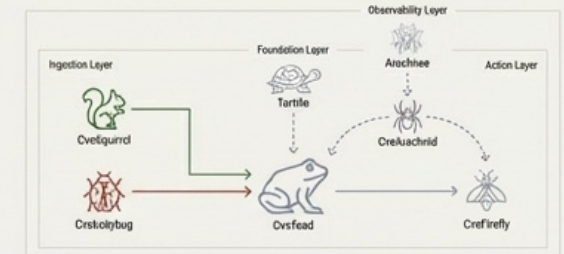
CveSquirrel: *The Collector*

A dedicated data-ingestion tool that collects all relevant CVE metadata, package metadata, and system data required for downstream analysis. It ensures a complete data picture from the start.



CveLadybug: *The Validator*

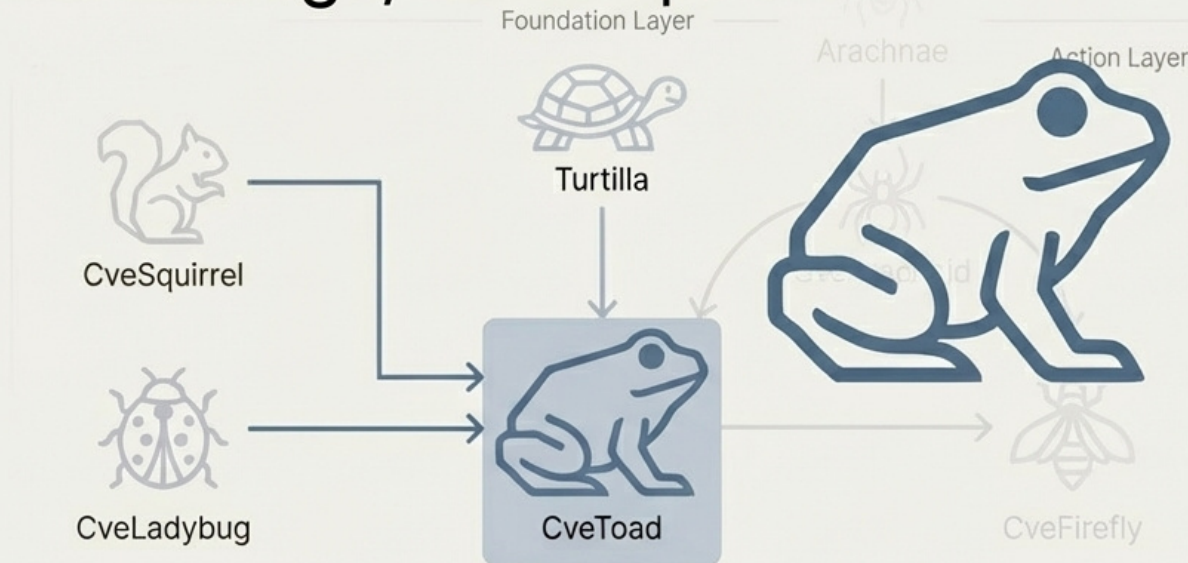
Spots anomalies or bugs in CVE templates across heterogeneous sources. Compiles a CVE's data from multiple origins to cross-reference and validate impact, providing context for automated CVSS vector estimation.



Benefit: Solves the 'garbage in, garbage out' problem by ensuring data is **complete** and **cross-validated** before analysis begins.

Phase 2: The Core Engine — CveToad

“Hops on, helps you catch some bugs, then hops off.”



What it is:

Temporarily hops onto an OS to detect and validate active bugs/CVEs.

How it works:

Implements a **CVE lifecycle workflow** where **automation** is coupled with key **manual verification** to minimize and abstract downstream threat modeling complexity.

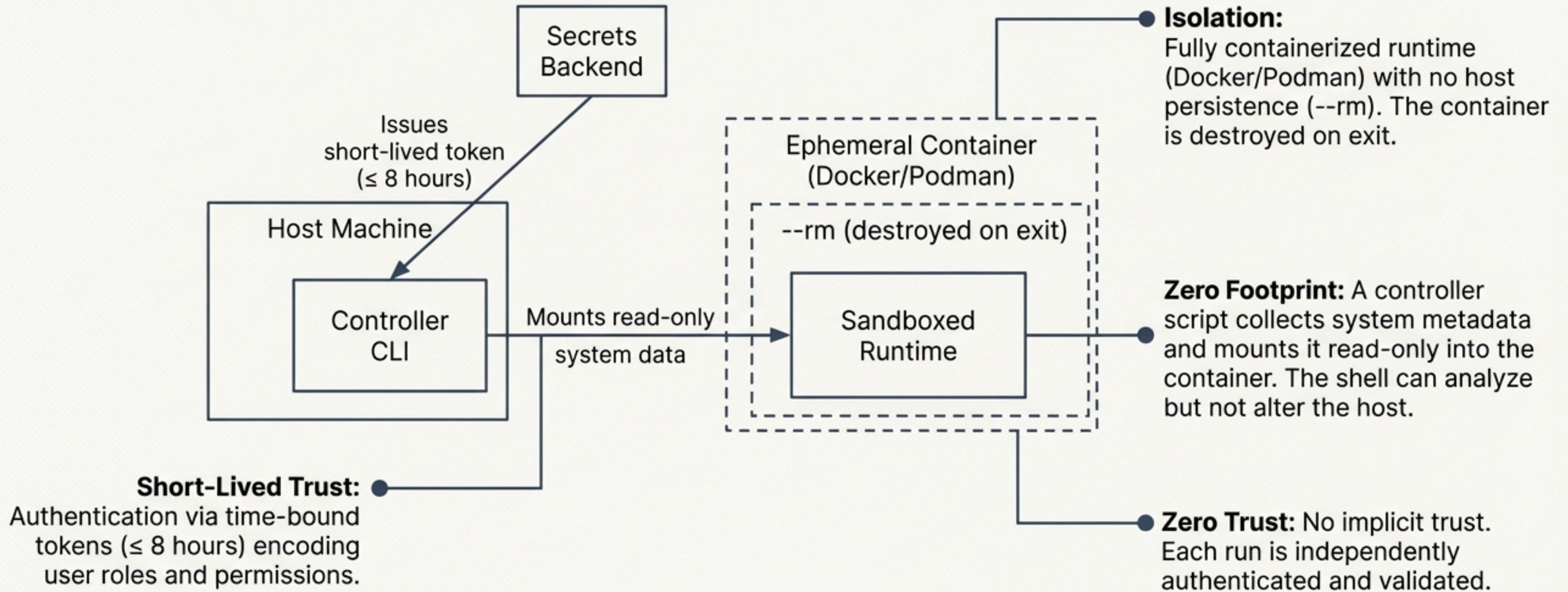
Key Function:

Serves as the central point where validated data from CveSquirrel and CveLadybug is applied to a specific system context. It relies on the Turtilla shell for its sandboxed execution environment.

“CveToad... incrementally feed[s] it the right context to optimise how accurately it predicts individual CVSS vector elements.”

The Core Innovation: A Secure, Ephemeral AI Shell

Zero footprint. Short-lived trust. Auditable sessions.

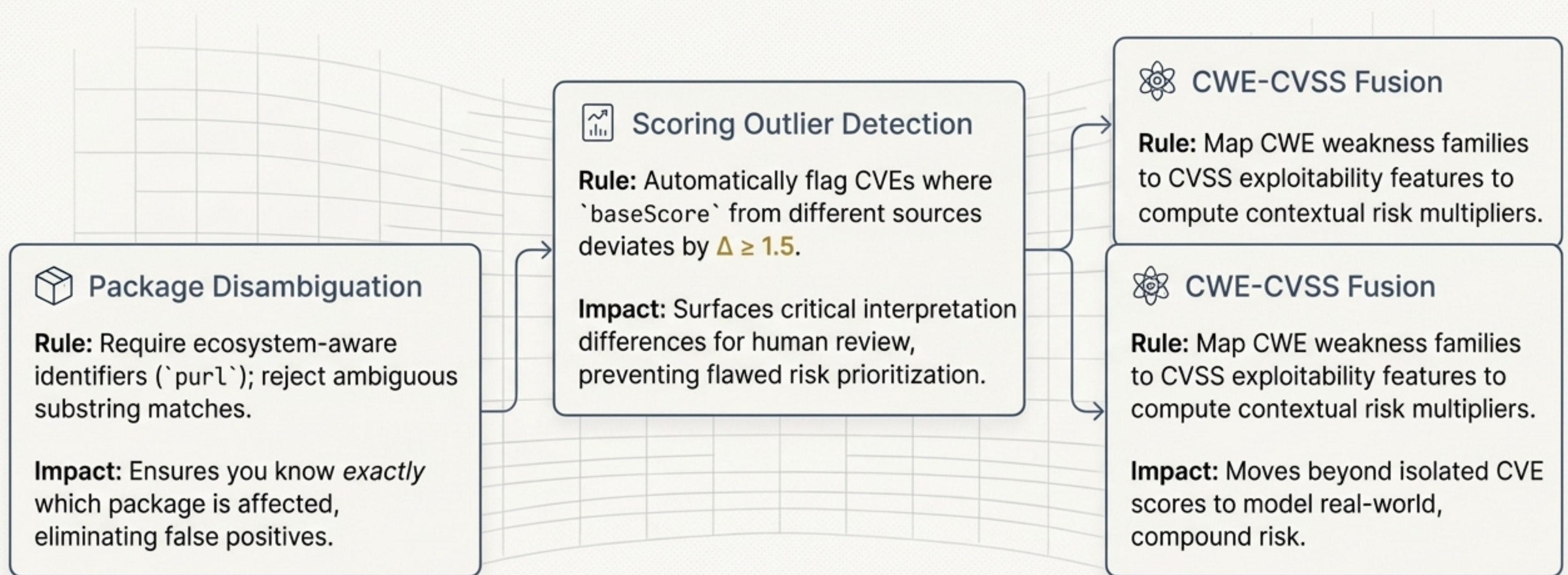


"Hops on, helps you catch some bugs, then hops off."

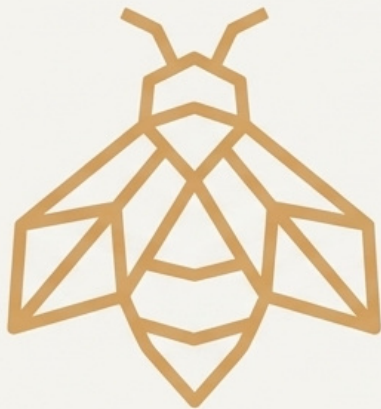
github.com/keerthanap8898/CveToad

The Engine of Trust: Enforcing Quality at the Source

Our system is built on a rigorous CVE Metadata Framework that programmatically validates, normalizes, and enriches vulnerability data before it enters your workflow.



Phase 3: From Validation to Active Assessment with CveFirefly




CveFirefly: *The Assessor*


Automates and runs open-source AI penetration testing tools to perform active security assessments on the **curated and verified CVE data** produced by CveToad.

This step moves beyond theoretical analysis to practical, active testing, answering the question: "Can this vulnerability actually be exploited in this environment?"



Example OSS Tools:

 osv-scalibr

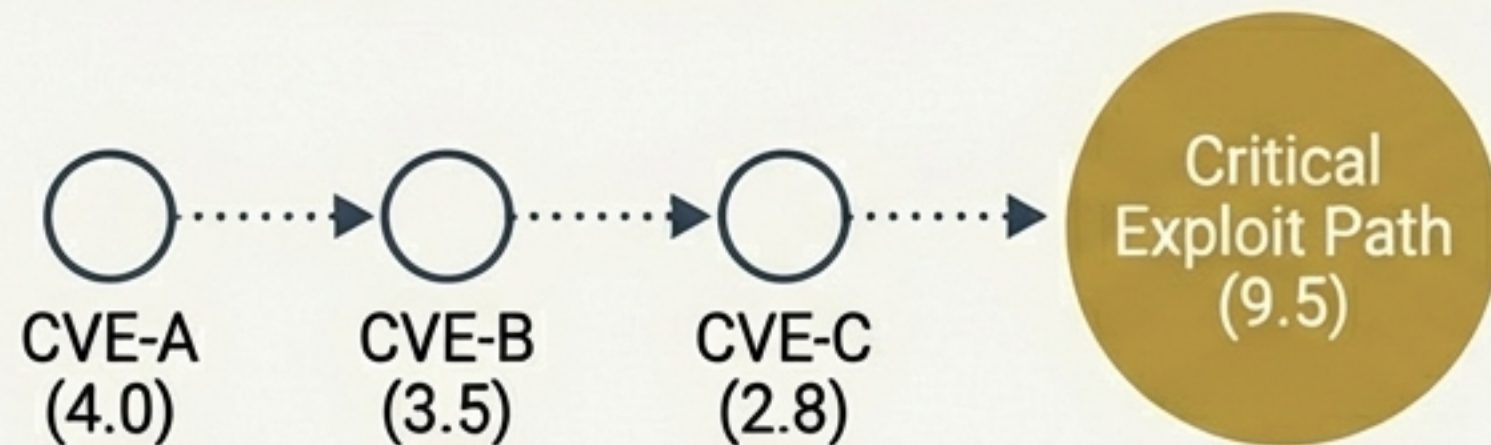
 raptor

Ensures that security efforts are focused on verified, actively testable vulnerabilities, not on noise.

Addressing the Hard Problems: Compound Risk and the AI Frontier

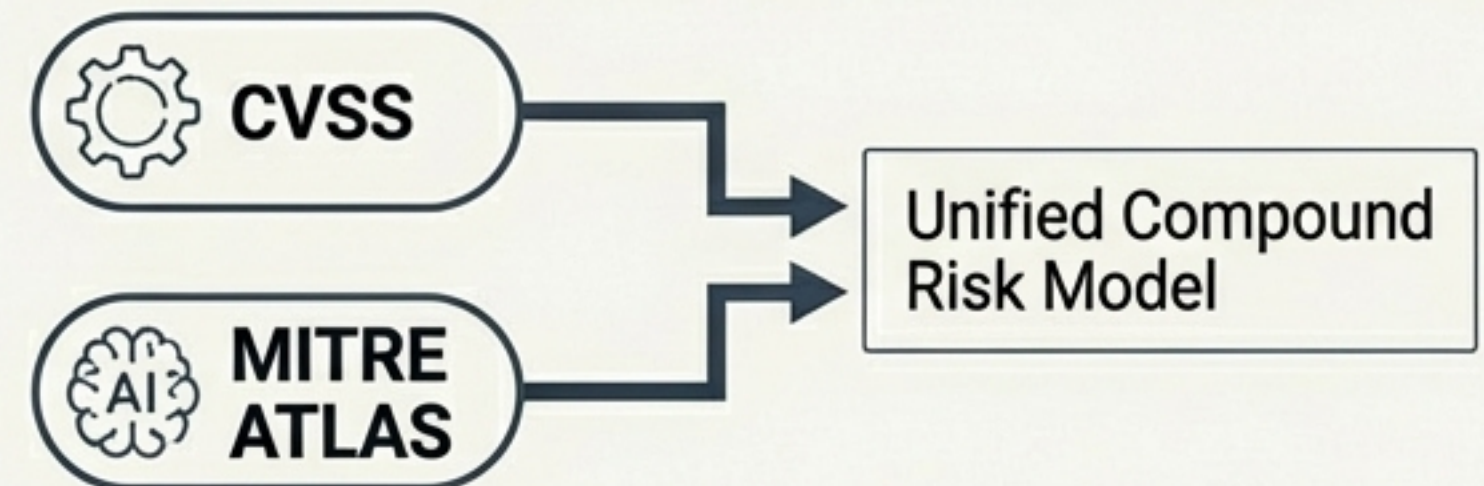
Current scoring models fail because they assess vulnerabilities in isolation. In the real world, multiple low-severity issues can be chained together to create a critical, exploitable path. Our framework is designed to model this reality.

Key Concept 1: Assessing the Impact Boundary



Our system evaluates risk across the entire dependency graph (direct and transitive), recognizing that multiple moderate issues can combine into high-impact failures.

Key Concept 2: Readiness for the Next Challenge



The architecture is explicitly designed to correlate traditional CVSS metrics with emerging AI-specific vulnerability vectors, such as those in the **MITRE ATLAS** framework. This allows for the evaluation of compound impact across software, infrastructure, and AI components.

Phase 4: Closing the Loop with CveArachnid Observability



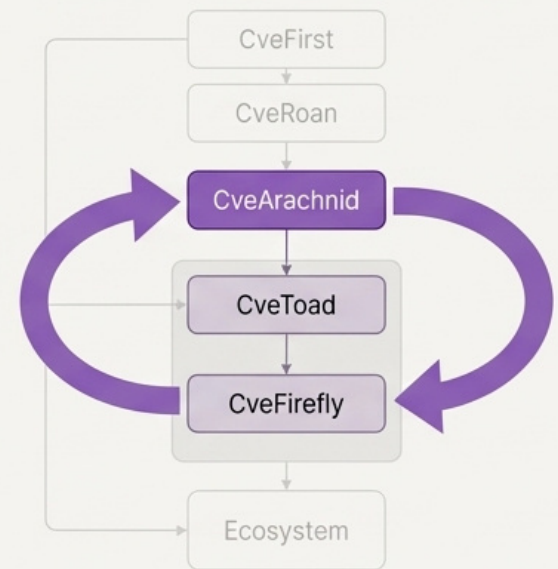
CveArachnid: *The Observer*

Optimized to run validation and real-time tests to track **CVE correctness through their lifecycle from Day-0**.

Monitors the outputs of **CveToad** and **CveFirefly**, offering visualization, contextualization, and proof tied to system-specific traits.

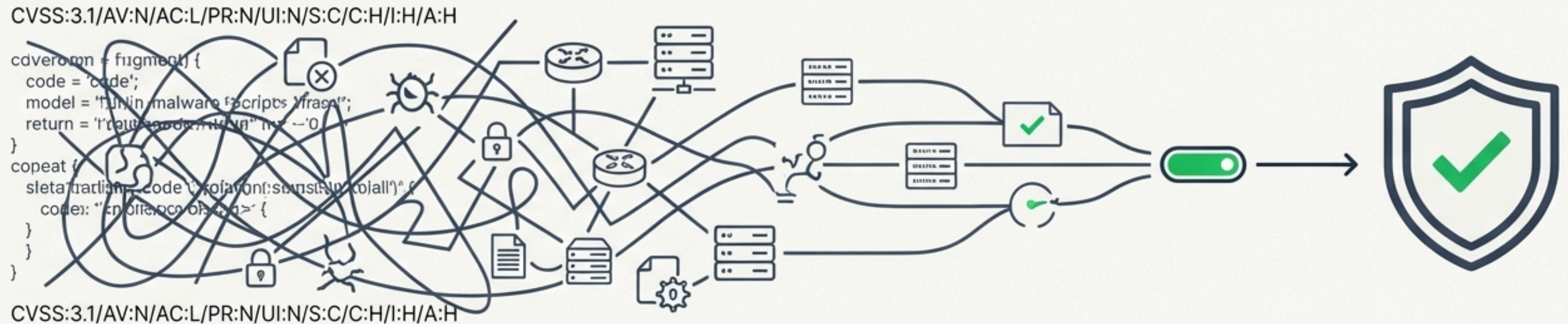
Normalizes results across diverse configurations, ensuring that findings are consistent and comparable.

Acts as the foundational observability layer, enabling correctness and validation steps and ensuring the entire ecosystem remains trustworthy over time. Depends on the **Arachnae** template app for its telemetry and dashboarding foundation.



Security Literacy for All

Security impacts everybody, but not everybody understands security or can afford enterprise-level support. The ultimate goal is to **abstract** the complex theory of CVSS and MITRE ATT&CK and communicate real, use-case-specific risk to a common civilian.

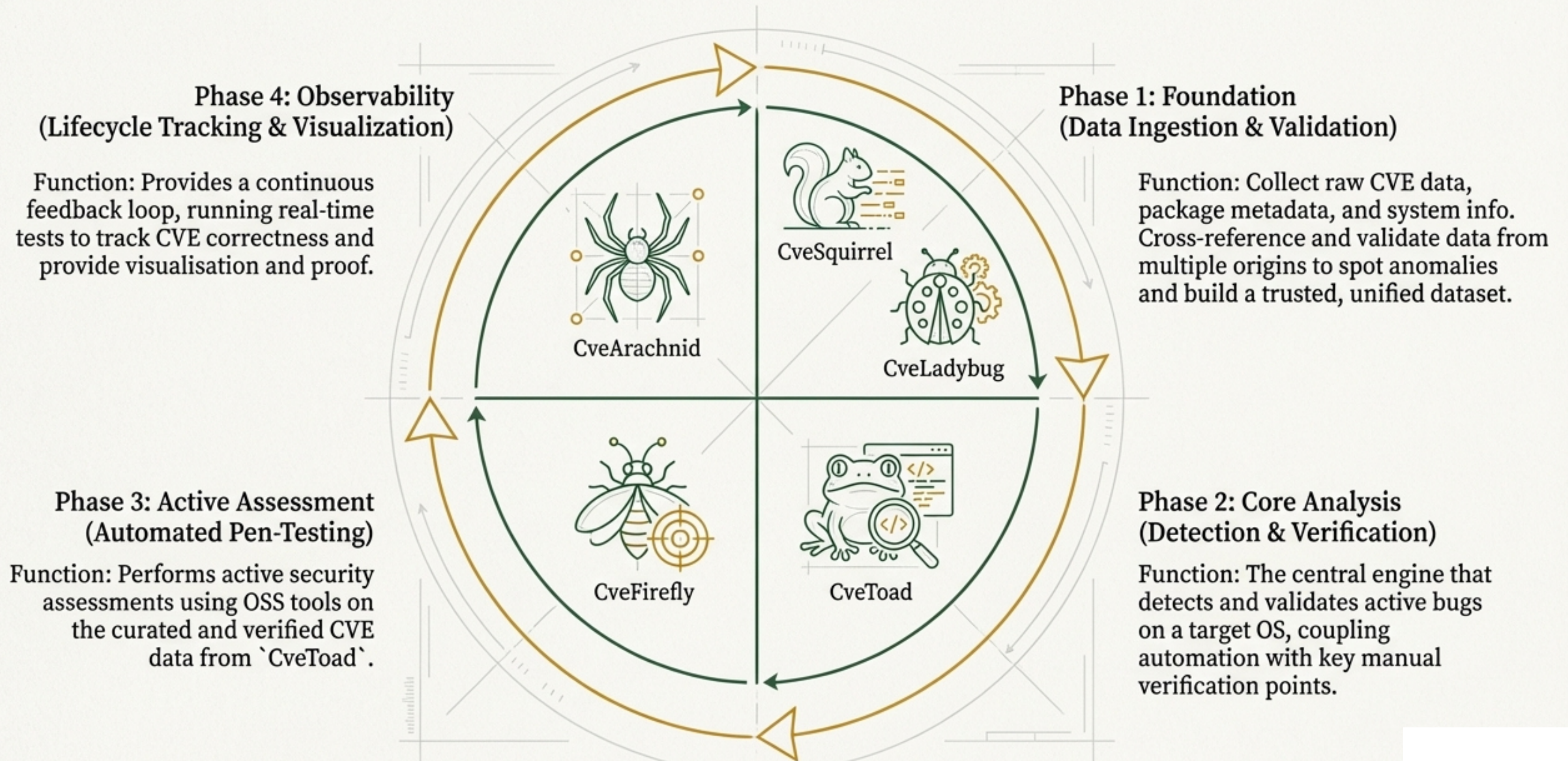


“We should aim to be able to simplify & **communicate** CVSS/MITRE exploitability etc metrics to a common civilian by abstracting the theory and focusing on their use-case.”



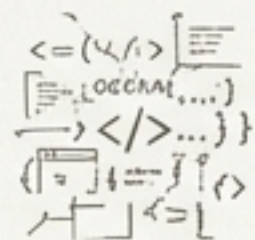
“The goal is to give everyone not just access but also literacy of security.”

The Lifecycle of a Vulnerability: From Ingestion to Proof



From Reactive Chaos to Proactive Intelligence

The Problem (Before)



- Inconsistent, Malformed Data



- Conflicting CVSS Scores



- Duplicated Engineering Effort



- Low Confidence in Automation



- Manual, Time-Consuming Analysis



The Ecosystem Solution (After)



- Standardised, Cross-Validated Data (**CveLadybug**)



- Contextual, Automated Vector Estimation (**CveLadybug**)



- Centralised, Reusable Logic (Ecosystem Design)



- Verified Data for Trusted Automation (**CveToad**)



- Secure, Ephemeral AI-Assisted Assessment (**CveToad**, **CveFirefly**)

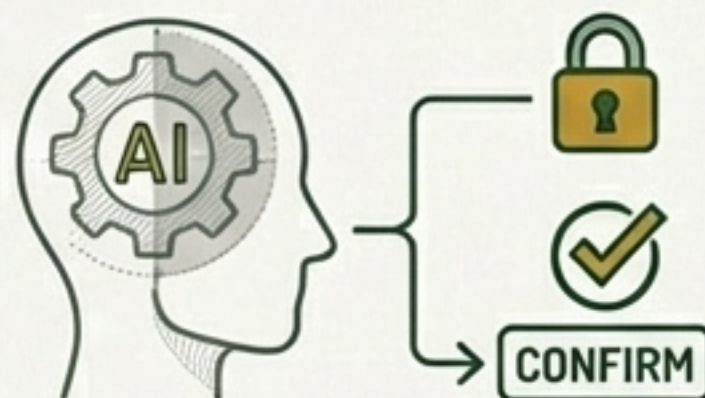
A Framework for Systemic Improvement

The Cyber-Fauna ecosystem is designed to not only consume CVE data but also to improve it.



Enabling Better Upstream Feedback

The detailed validation from **CveLadybug** and **CveArachnid** can generate high-quality feedback for CNAs, helping to resolve 'CWE-Other' and other ambiguities at the source.



Justifiable AI with Human Oversight

The workflow enforces UI-interactive confirmation at key milestones. It's about aiding human experts, not replacing them.

"I don't trust AI that much & I will find ways to enforce UI-interactive-confirmation."



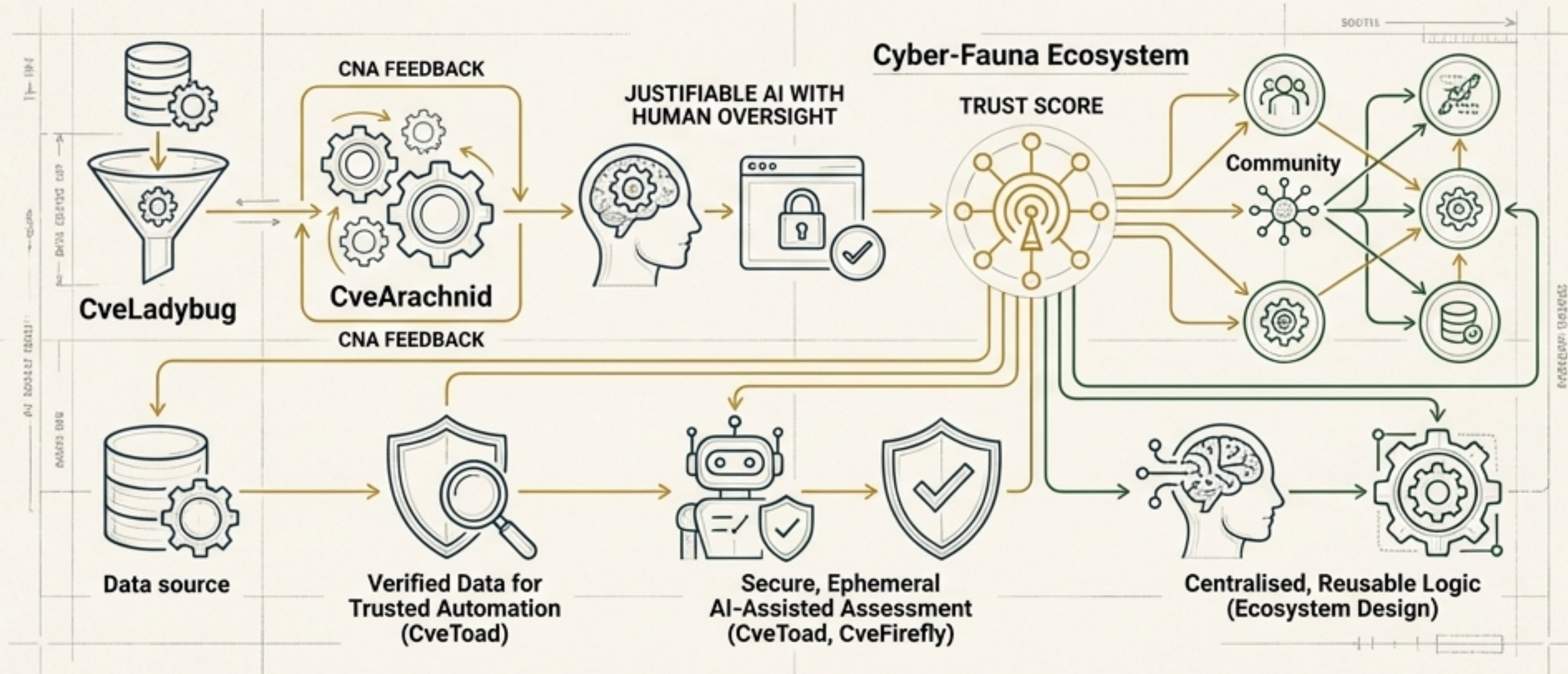
Fostering Community Intelligence

The system's design supports a future where community-sourced data can be integrated to create a **trust score** for CVE correctness, democratising impact assessment.

The goal is to create a self-reinforcing cycle of improvement, where **better data leads to better** models, and ultimately, a more secure digital ecosystem.

Transforming the CVE Data Supply Chain

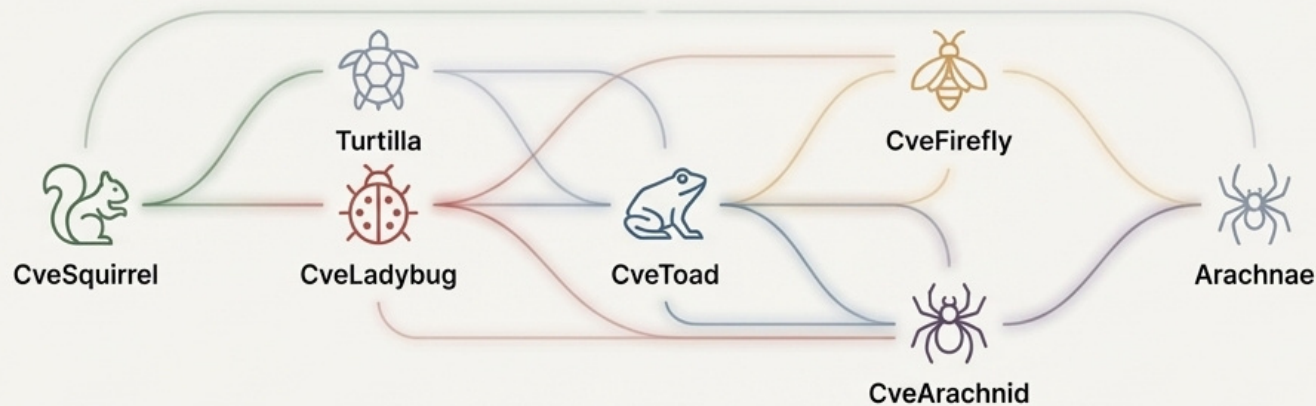
From a Fragmented Ingestion Practice into a Unified, Trustworthy Data Pipeline.



“By addressing these foundational quality issues within the global CVE ecosystem, the security community can power reliable automation, accelerate remediation, and ultimately strengthen cyber resilience across the world’s most critical infrastructures.”.

Transforming the CVE Data Supply Chain

From a Fragmented Ingestion Practice into a Unified, Trustworthy Data Pipeline



“By addressing these foundational quality issues within the **global CVE ecosystem**, the security community can power reliable automation, accelerate remediation, and ultimately strengthen cyber resilience across the world’s most critical infrastructures.”

The Google Chrome of Vulnerability Management

We have a formal system design that generalizes the challenges seen across the entire CVE lifecycle. By building a unified, open-source platform that prioritizes data quality, secure automation, and human accountability, we can fundamentally change how the world manages vulnerability intelligence.

*Security should be at $x+1$ to
whatever x momentum AI is at.*

By addressing these foundational quality issues within the global CVE ecosystem, the security community can power reliable automation, accelerate remediation, and ultimately **strengthen** cyber resilience across the world's most critical infrastructures.

From CVE Chaos to Clarity: The Cyber-Fauna Intelligence Ecosystem

The Problem: A Broken CVE Data Supply Chain



Structural & Semantic Errors
Records contain malformed data, such as embedded HTML or entire commit logs in text fields.




Conflicting Metrics & Scores
Different sources report conflicting severity scores for the same CVE, forcing manual arbitration.




Ambiguous Classifications
Many vulnerabilities are vaguely classified (e.g., "CWE-Other"), hindering automated analysis and remediation.

The Solution: An Automated Vulnerability Intelligence Lifecycle


Phase 1: Ingestion & Validation
CveSquirrel collects raw CVE data, and CveLadybug spots anomalies and validates it across sources.




Phase 2: Core Analysis & Verification
CveToad "hops on" a system in a secure, ephemeral shell to validate active bugs.

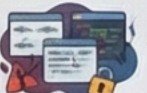







Phase 3: Active Assessment
CveFirefly uses the verified data to run automated penetration testing tools.



Phase 4: Observability & Feedback
CveArachnid provides continuous lifecycle tracking, visualization, and proof of CVE status.







CHAOS vs. CLARITY	
The Problem (Before)	The Ecosystem Solution (After)
 Inconsistent, Malformed Data	Standardized, Cross-Validated Data (CveLadybug) 
 Subjective Score Arbitration	Contextual, Automated Score Estimation (CveToad) 
 Manual, Time-Consuming Analysis	Secure, AI-Assisted Assessment (CveToad & CveFirefly) 

#	Category	Field / Element	JSONPath / JSON Pointer	Purpose / Description	Normalization / Validation Rules	Derived or Analytical Metrics
1	Core Metadata	fields: id, state, published, lastModified, providerMetadata	\$.cveMetadata.*, \$.containers.cna.providerMetadata	CVE identifiers, status, and CNA info.	Ensure consistent ID format (CVE-\d{4}-\d+); validate CNA attribution.	Record lineage; publication latency.
2	Description(s)	\$.containers.cna.descriptions[*].value (+ .lang)	Textual narrative of vulnerability, with language tags (BCP-47).	Enforce language codes; trim HTML tags; flag extreme length or non-ASCII noise.	char_len, token_len, html_tag_count, text anomaly scores.	
3	Affected Software / Platforms	\$.containers.cna.affected[*]	Lists impacted vendor, product, versions, platforms.	Require structured arrays; normalize vendor/product names; reject substring matches.	Ecosystem alignment via purl; product-level vulnerability graph.	
4	Source / CNA / Credits	\$.containers.cna.providerMetadata, \$.containers.cna.credits[*]	CVE issuer and contributors.	Validate CNA names against registry; ensure timestamps and contact fields exist.	Contributor diversity metrics; provenance reliability.	
5	Severity & Metrics (CVSS)	\$.containers.cna.metrics[*], \$.containers.adp[*].metrics[*]	CVSS scoring objects (v2/v3/v3.1/v4).	Check prefix-version match (CVSS:3.1/  object); reject invalid tokens; $\Delta \geq 1.5$ between sources \rightarrow flag.	base_score, base_severity, temporal_score, env_score, Exploitability Index, Cross-Vector Drift.	
6	Vector String	fields: ...vectorString	Encoded attack/impact vector.	Regex validate full CVSS structure; order and enums must match version spec.	Parsed metric factors (AV, AC, PR, UI, S, C, I, A).	
7	Temporal / Environmental Scores	fields: ...temporalScore, ...environmentalScore	Temporal and environmental context modifiers.	Verify enums (E, RL, RC); check consistency with base vector.	Adjusted severity differentials; confidence weighting.	
8	CWE / Problem Type	\$.containers.cna.problemTypes[*].descriptions[*].cweId, .description	CWE identifier and weakness description.	Match ^CWE-\d+; allow "CWE-Other" / "NoInfo" with expiry; compare text similarity ≥ 0.85 to official CWE.	CWE Depth, Specificity Index, CWE Completeness Score.	
9	References	\$.containers.cna.references[*]	URLs for advisories, patches, etc.	Validate URL format and type; deduplicate.	Reference type coverage; vendor vs. third-party ratio.	
10	Languages	\$.containers.cna.descriptions[*].lang	Declared description languages.	Must be valid BCP-47 code.	Multilingual coverage stats.	
11	Text / Semantic Outliers	—	Description or metric fields.	IQR thresholds for char_len, non-ASCII%, or HTML tags.	Text anomaly score; linguistic quality metrics.	
12	Scoring Outliers	—	CVSS scores across CNAs.	$\Delta \geq 1.5$ between base scores across sources \rightarrow anomaly.	Cross-source variance index.	
13	Identity Anomalies	—	Product/package names.	Require ecosystem-aware purl; no substring matching.	Product identity consistency rate.	
14	CWE-CVSS Fusion	CWE categories \times CVSS vectors	Derived model.	Map CWE family to exploitability features; compute contextual multipliers.	Risk aggregation index; predictive inference model inputs.	
15	CVE-Core Tables	fields: cve_core, description, affected, metric, problemtype, ref	Canonical relational design.	Ensure referential integrity; unique id.	Enables cross-CNA joins and longitudinal analysis.	
16	Upstream Quality Feedback	—	—	Identify systematic schema or data issues.	Enforce HTML/diff filters; metric provenance; SLA for "Other/NoInfo".	Upstream feedback metrics; CNA quality ranking.
17	Validation Regex	patterns: <[>]+>, `(?m)^(?:diff --git	index [0-9a-f]{7,}	@@[^@]+@@`; CVSS v3.1 coarse pattern	Regex-based validators.	Match/strip patterns before ingestion.
18	Package Disambiguation Rules	—	Affected product fields.	Tokenize by [-_.]; lowercase; drop non-semantic suffixes; restrict by ecosystem.	Package-match precision; false-positive rate.	
19	Statistical / Outlier Framework	—	Entire dataset (/cves/YYYY/**)	Quantitative quality analysis.	Compute P50/P75/P90 annually; flag $\geq Q3 + 1.5 \times \text{IQR}$.	Annual anomaly index; data-quality heatmap.
20	Upstream Data-Quality Recommendations	—	—	Guidelines for CNAs / MITRE improvements.	HTML filter; provenance fields; SLA for CWE refinement; purl inclusion.	Reduction in downstream remediation workload.
21	Extraction Selectors	As defined in section 8	Deterministic selectors for JSON fields.	Maintain reproducible extraction paths and validation rules.	Pipeline reproducibility score.	

Join the Initiative

This is an open-source initiative to fix the foundation of global vulnerability management. We are seeking partners, contributors, and early adopters.

- 1.  Explore the Research & Code**
Dive into the detailed documentation and components on GitHub.
github.com/keerthanap8898/CveToad
- 2.  Join the Automation Working Group**
Contribute to the standards and schema discussions.
cve.org/ProgramOrganization/WorkingGroups
- 3.  Partner on a Pilot Program**
Work with us to deploy and validate the Cyber-Fauna Ecosystem in your environment. Contact: keep.consult@proton.me
- 4.  Take the CVE Consumer Survey**
Provide feedback on your current pain points and needs.
forms.gle/QNCciH4vF7gtMaMF6
github.com/keerthanap8898/CveToad