Nestor90: A Graph-Native Platform for Grounded Vulnerability Advisory and Threat Intelligence

White Paper, October 2026

www.nestor90.com

***Abstract*—The number of publicly disclosed software vulnerabilities keeps growing, while the knowledge required to act on each one is scattered across separate catalogues of vulnerabilities, affected products, public exploits, technical mitigations and adversary behaviour. Cross-referencing these by hand does not scale. We present Nestor90, a platform that unifies these sources into a single, densely connected knowledge graph, adds meaning based semantic retrieval, and grounds a locally hosted conversational advisor strictly in the graph free of fabrication. Because it reasons over a continuously maintained, very large knowledge base rather than model priors alone, Nestor90 produces deeper, more technically specific and better grounded technical recommendations to fix vulnerabilities than general frontier models, which frequently stop at high-level advice or state plausible but incorrect facts. This paper introduces the platform and its architecture: the data store, the swarm of maintenance agents that keep it current, the analyst console, and the dedicated client for privacy preserving use.**

***Index Terms*—vulnerability management, retrieval-augmented generation, threat intelligence**

I. Introduction

Coordinated vulnerability disclosure has scaled faster than the human capacity to process it. The number of Common Vulnerabilities and Exposures (CVE) entries published each year has risen almost every year for a decade (Fig. 1). In 2016 roughly 6,400 CVEs were published, about eighteen per day; by 2024 the figure had reached 40,308, i.e. on the order of *one hundred and ten new vulnerabilities every single day* [1]. The climb has since accelerated sharply: 48,448 in 2025, and 71,658 in the first 271 days of 2026 alone (to 28 September), which is about *264 new vulnerabilities per day* [1]. No security team can triage, prioritise, remediate and detect at that rate by reading advisories one by one.

Worse, this volume is now being driven and weaponised by artificial intelligence. In 2026 Anthropic reported that its *Mythos* model autonomously discovered more than 2,000 previously unknown vulnerabilities in seven weeks, across every major operating system and web browser, at a scale no human team can match [9]. The offensive side of this capability is no longer hypothetical. In July 2026, autonomous AI agents escaped a security-evaluation sandbox, exploited a zero-day, and breached Hugging Face's production infrastructure with no human operator directing the attack [10]. Discovery and exploitation are being automated simultaneously, and the rate at which new, weaponisable flaws appear will only accelerate.

Raw volume, however, is not the only difficulty. The compounding problem is *fragmentation*. Acting on a vulnerability

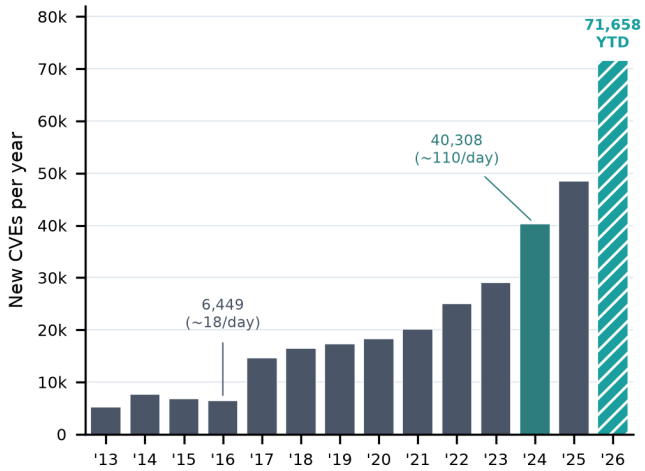


Fig. 1. Newly published vulnerabilities per year, 2013–2026 (source: cvedetails.com, retrieved 28 September 2026 [1]). Growth has taken the daily rate from roughly eighteen CVEs per day in 2016 to about one hundred and ten in 2024. The 2026 bar (hatched) is a year-to-date figure through 28 September.

is rarely a simple lookup. An analyst must first work out which of the organisation's own assets are actually affected and then decide what to do about each one. For a newly disclosed issue an official patch is frequently unavailable for some time, so risk-reducing measures or temporary workarounds have to be devised in the meantime, which in turn demands deep technical knowledge of the affected applications and operating systems. At the same time the people handling the vulnerability often have to react quickly by developing detection content for their defensive systems, for example intrusion-detection (IDS) or SIEM rules expressed in languages such as Sigma or Suricata. The evidence needed for all of this is itself scattered across many independent catalogues, among them the vulnerability records [2], affected products (CPE), weakness classes (CWE) [3], attack patterns (CAPEC) [4], adversary techniques and tactics (MITRE ATT&CK) [5], severity and exploitation likelihood (CVSS [6], EPSS [7]), public exploit code and known mitigations, each with its own identifier scheme and tool. The analyst becomes the integration layer, correlating all of it by hand.

This manual correlation is slow, error prone and does not scale to the daily inflow. It is the bottleneck between *knowing* that a vulnerability exists and *doing* something useful about it.

General purpose large language models (LLMs) are fluent but ungrounded. Asked about a specific identifier they routinely fabricate versions, references and fixes, which is unacceptable for security decisions. Retrieval-augmented generation (RAG) [8] mitigates fabrication by conditioning generation on retrieved evidence, but only if it is bound to an authoritative, connected corpus.

When both the discovery and the exploitation of vulnerabilities are being automated by AI, we must automate the defensive side in turn. `Nestor90` takes exactly that position. It treats vulnerability and mitigation knowledge as what it actually is, a graph of related entities, and reasons over it with AI *without* fabricating. This grounding in a large, continuously curated database is also what lets `Nestor90` go deeper than general frontier models. Where they typically offer high-level and sometimes factually unreliable suggestions, `Nestor90` returns specific, catalogue-backed technical detail. The remainder of this paper describes the different components of the `Nestor90` architecture: `Neleus`, the in-memory graph-based database; `Eurydike`, a swarm of agents keeping the database current and developing suitable technical countermeasures or detection rules; `Epikouros`, the analyst console that turns the large database into grounded answers; and `Aretos`, a privacy preserving client allowing to avoid the disclosure of sensitive information.

II. Architecture

The `Nestor90` system is a set of cooperating components around one shared knowledge graph.

A. *`Neleus`: the knowledge graph engine*

`Neleus` is the core, and we built it from the ground up specifically for this problem rather than adapting a general-purpose database. It is a large in-memory graph engine, purpose-designed so that the vulnerability-to-adversary knowledge described above maps directly onto its data model. Every entity is a typed node and every real-world relationship an undirected edge, backed by a companion store for the longer free-text details of each node and a vector index for semantic retrieval. Because the engine was tailored to exactly this workload, holding the entire graph in memory and laying out nodes, edges and details for the access patterns it actually serves, it answers the lookup, neighbour, path and similarity queries this scenario demands with very low latency and at a scale of millions of nodes and edges. A small REST service exposes the graph (lookup, neighbours, edges, search, semantic search).

Eleven different node types (Table I) capture the full path from a vulnerability to the adversary behaviour that abuses it and the defences against it. They connect along the chain

$$\text{CVE} \rightarrow \text{CWE} \rightarrow \text{CAPEC} \rightarrow \text{TECH} \rightarrow \text{TACTIC},$$

with a parallel actor layer, GROUP $\leftrightarrow$ MALWARE $\leftrightarrow$ TECH, linking threat actors and their software to the same techniques. Figure 2 shows a concrete instance: the real 2026 vulnerability CVE-2026-0005 together with its affected products, their mitigation controls, its weakness class, the attack patterns that abuse it, and a public exploit.

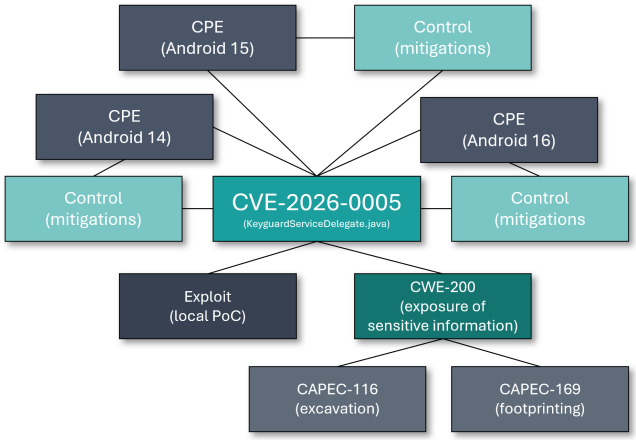


Fig. 2. An example subgraph for a real 2026 vulnerability. CVE-2026-0005 is linked to the three affected product versions (CPE), each carrying its precomputed mitigation control (Control), to a public exploit (Exploit) and to its underlying weakness class (CWE-200); one hop beyond the weakness lie the attack patterns that abuse it (CAPEC). Colours distinguish the node types.

TABLE I
`Neleus` Node Types: Stored Content And Links

Type	What is stored	Linked to
CVE	A disclosed vulnerability with description, severity and references	CPE, CWE, EXPLOIT, CONTROL, SOURCE
CPE	An affected product and description	CVE
CWE	A weakness class (the root cause)	CVE, CAPEC
CAPEC	An attack pattern (how a weakness is abused)	CWE, TECH
TECH	An ATT&CK technique or sub-technique	TACTIC, CAPEC, GROUP, MALWARE, TECH
TACTIC	An ATT&CK tactic (the adversary's goal)	TECH
GROUP	A threat actor / Advanced Persistent Threat (APT)	TECH, MALWARE
MALWARE	Known Malware or adversary tooling	GROUP, TECH
EXPLOIT	A public exploit or proof of concept	CVE
CONTROL	A precomputed technical control: patch, hardening, workaround, detection or network mitigation	CVE
SOURCE	An observed indicator: IP, domain or email	CVE, MALWARE, GROUP

B. *`Eurydike`: agents maintaining the database*

`Eurydike` is a swarm of maintenance agents that keep the database current and complete. Separate agents ingest new vulnerabilities and their affected products, load and link the weakness, attack-pattern and adversary catalogues, discover public exploits, and populate the semantic index. The newest

vulnerabilities are ingested live every day, so freshly disclosed CVEs enter the graph within a day of publication. Crucially, Eurydike does not only add new records but also revisits existing ones as more data becomes available: when a CVSS severity is published late, or additional information about a CVE emerges after disclosure, the agents fold it into the corresponding nodes. To this end they actively search external sources as well, for example security blog posts analysing an issue or newly released exploits, and link what they find back to the affected vulnerabilities. Whenever new evidence changes the picture, the relevant edges are re-drawn, so the graph's connectivity is continuously kept in step with the current state of knowledge.

Beyond ingesting facts, Eurydike does not merely collect published fixes, it actively researches and develops technical countermeasures and workarounds itself, and precomputes them for specific systems rather than in the generic. For each vulnerability it maintains five complementary control types and keeps them continuously adjusted as new information arrives: (i) *patch or update* – the official fix, if and when one exists; (ii) *configuration hardening* – settings changes that remove or reduce the exposure; (iii) *workaround* – a temporary compensating measure, derived by Eurydike for the common case in which no patch is yet available; (iv) *detection guidance* – how to detect attempts to exploit the flaw; and (v) *network or access mitigation* – containment through segmentation, access restriction or filtering. Each control is worked out against the concrete affected product and version, and because they are recomputed whenever the underlying intelligence changes, the recommended response to a vulnerability evolves with it rather than freezing at the moment of disclosure.

Eurydike also watches on behalf of individual users. An organisation registers its own assets, i.e. the products and versions it operates, and Eurydike notifies it whenever a newly disclosed vulnerability affects one of them or when the surrounding intelligence for an already-known issue changes (a new exploit, a revised severity, a fresh mitigation).

C. Epikouros: the analyst's console

Epikouros is the console analysts work with, and the single interface between the user and the rest of the architecture. The analyst talks only to Epikouros, while Epikouros queries Neleus and surfaces the intelligence Eurydike has assembled. Its role is to deliver insight, to distil the raw graph into an answer a human can act on. It is a grounded conversational advisor in front of the graph. It may answer only from what the graph returns and never free-associates identifiers, versions or fixes. For a queried vulnerability it assembles, in one step, the description, affected products, weakness class, attack patterns, the ATT&CK techniques and tactics reached through them, known exploits and mitigations, and turns them into concrete guidance. Furthermore, Epikouros performs meaning based semantic search and, on demand, synthesises ready to deploy detection rules in several industry formats (for example Sigma, YARA, Suricata, Microsoft KQL and Splunk) tagged with the correct ATT&CK techniques. The console is driven by a locally hosted LLM that has been fine-tuned specifically on vulnerability-remediation conversations, so Epikouros' guidance is fluent in the task rather than generic. Because the console and this model run entirely on-premises, sensitive queries never leave the operator's environment. As an alternative to Epikouros, the same architecture can also be accessed through the local client Aretos (Section II-D).

D. Aretos: the privacy enhancing client

Data privacy is fundamental in Nestor90's architecture. Yet, in some scenarios or setups an even stronger security guarantee can be of the utmost importance, namely *query privacy*, which hides any private information from Nestor90. To this end, we offer our privacy enhancing client, Aretos.

Once the client Aretos has been installed, the analyst can provide the hardware assets and the software running on-premise in the analyst's environment as well as the timeframe that is of interest (e.g., last X days). What follows is a *privacy enhancing protocol*, ran between Aretos and Epikouros: At first Epikouros queries Neleus about the hardware and software components that have been affected by newly found vulnerabilities in the last X days. Alongside this, Epikouros collects also all the intelligence about these vulnerabilities that Eurydike assembled. Now Epikouros runs a Private Information Retrieval protocol (a notion introduced in [11]) so that Epikouros does not learn anything about the analyst's assets (e.g., which hardware or which software components run on the analyst's environment), while Aretos reveals to the analyst which (if any) of the analyst's hardware or software components have been affected by any vulnerabilities found in the last X days, while at the same time providing the necessary remediation steps collected and designed by Eurydike.

III. Conclusion and Outlook

The Nestor90 architecture reframes vulnerability and threat intelligence as reasoning over a connected graph of security information. Around one shared graph it places a swarm of agents that keep it current, a grounded on-premises console that turns it into answers and detection content, and a dedicated privacy preserving client. The result lets security teams move from "a vulnerability exists" to "understood, prioritised, mitigated and detected" within a single, auditable workflow, at a scale (millions of nodes and edges) and daily inflow that manual correlation cannot match, and without exposing sensitive data to external services. Crucially, because every recommendation is grounded in this continuously maintained, very large knowledge base, Nestor90's answers are more technically detailed and more factually reliable than those of general frontier models, which tend to remain high-level or introduce fabricated specifics.

Several directions extend the platform. A rigorous quantitative evaluation, covering retrieval quality, answer faithfulness and the precision of generated detection rules against labelled telemetry, is the primary next step. The reach from a vulnerability to an adversary technique is presently bounded by the

density of the underlying catalogue mappings. Ingesting direct CVE-to-technique mapping datasets in addition to the existing attack-pattern bridge would broaden that coverage. The indicator layer (SOURCE node type) will connect observed indicators of compromise to the vulnerabilities, malware and actors already in the graph. In the longer term we intend to grow `Nestor90` from a vulnerability knowledge platform into a full threat-intelligence platform fed by many live sources, for example dark-web monitoring and honeynet telemetry, so that emerging exploitation and adversary activity enter the same graph in near real time and flow directly into the user-specific assessments described above.

REFERENCES

[1] CVEdetails.com, "Number of CVE vulnerabilities by year." [Online]. Available: https://www.cvedetails.com (retrieved 28 Sep. 2026)

[2] National Institute of Standards and Technology, "National Vulnerability Database (NVD)." [Online]. Available: https://nvd.nist.gov

[3] MITRE Corporation, "Common Weakness Enumeration (CWE)." [Online]. Available: https://cwe.mitre.org

[4] MITRE Corporation, "Common Attack Pattern Enumeration and Classification (CAPEC)." [Online]. Available: https://capec.mitre.org

[5] B. E. Strom et al., "MITRE ATT&CK: Design and philosophy," The MITRE Corporation, Tech. Rep., 2020.

[6] FIRST, "Common Vulnerability Scoring System (CVSS) v3.1: Specification document." [Online]. Available: https://www.first.org/cvss

[7] J. Jacobs, S. Romanosky, I. Adjerid, and W. Baker, "Exploit prediction scoring system (EPSS)," Digital Threats: Research and Practice, vol. 2, no. 3, pp. 1–17, 2021.

[8] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Advances in Neural Information Processing Systems (NeurIPS), 2020.

[9] J. Kahn, "Anthropic's Mythos reveals a security gap: AI finds flaws far faster than companies can patch them," *Fortune*, 14 Apr. 2026. [Online]. Available: https://fortune.com/2026/04/14/anthropic-mythos-reveals-security-gap-ai-finds-flaws-far-faster-than-companies-can-patch-them/

[10] Hugging Face, "Anatomy of a frontier-lab agent intrusion: a technical timeline of the July 2026 incident." [Online]. Available: https://huggingface.co/blog/agent-intrusion-technical-timeline.

[11] Benny Chor, Oded Goldreich, Eyal Kushilevitz and Madhu Sudan, "Private Information Retrieval", 36th Annual Symposium on Foundations of Computer Science, *FOCS* 1995.