CPAMS: A Cybersecurity Post-Market Analysis Monitoring System

Gerald Rigdon
Fellow, Software Engineering
Boston Scientific Corporation
St. Paul, USA
gerald.rigdon@bsci.com

***Abstract*—The Cybersecurity Post-Market Analysis Monitoring System (CPAMS) represents a composite approach to cybersecurity vulnerability monitoring and research, combining artificial intelligence and multi-source data integration to deliver threat intelligence. This paper presents a detailed analysis of the system's architecture, methodology, and capabilities, with particular emphasis on its application to medical device cybersecurity post-market surveillance. CPAMS integrates data from the National Vulnerability Database (NVD) and Tavily web-based search, generates an Azure index, and utilizes both Embedding Models and Large Language Models (LLMs) for vulnerability discovery and identification.**

***Index Terms*—Cybersecurity, Vulnerability Analysis, Medical Device Security, Post-Market Surveillance**

I. Introduction

In the prior research discussed in [1], a custom software application named CyberVulModelEval was designed to test the viability of LLMs for cybersecurity monitoring based primarily on Retrieval Augmented Generation (RAG) strategies. CyberVulModelEval was presented as a RAG-Blend Framework, leveraging concepts from ReAct [2], Reflection [3], Fusion [4], and Correction [5]. That research acknowledged the vast landscape of possible future research directions, either separate or derivative of CyberVulModelEval. In summary, that research recommended caution and careful deliberation when considering RAG pipelines as the primary mechanism for cybersecurity vulnerability discovery.

While CPAMS represents a technical break from CyberVulModelEval, it remains focused on Common Vulnerabilities and Exposures (CVEs) in post-market monitoring of software in medical devices. Given the same core essential mission of Identification and Monitoring of Sources per [6], integral to CPAMS are CVEs tracked in the NVD [7]. As a result CPAMS is confronted with the concerns drawn from the Exploit Prediction Scoring System (EPSS) Data and Performance Study [8] which emphasizes cybersecurity concerns and the growing number of CVEs.

In contrast to CyberVulModelEval [1], the CPAMS core input is a Software Bill of Materials (SBOM), and takes advantage of conventional technology tools such as web searches, database queries, Azure indexing and search technologies, and LLM semantic powers for asset correlation. Thus, the central research questions to be answered are: (1) How effective are LLM-based semantic methods at correlating CVE descriptions — mined from multiple sources — with SBOM component metadata, compared to traditional keyword matching approaches? (2) Can LLMs improve SBOM-to-CVE matching precision by (a) increasing true positive identification through deeper semantic understanding of component descriptions, and (b) reducing false positives caused by superficial textual similarity between unrelated components?

In concert with CyberVulModelEval [1], the goal of CPAMS is to present further research on the use of AI, particularly LLMs, to help assess the effectiveness and practical use of these technologies within a cybersecurity post-market vulnerability monitoring framework.

II. Literature Review

Although the comprehensive review of over 300 papers encompassing 25 different LLMs presented in [9] was a valuable resource, the composite CPAMS approach emerged from the knowledge gained during the CyberVulModelEval study [1] where the literature review was more detailed.

III. Research Methodology Approach

The CPAMS process is captured in a series of steps as presented in Fig. 1 and further described in phases.

Input Phase: Per [Step 1] as depicted in Fig. 1, this phase involves the input of SBOM information (saved in JSON format) and additional keyword terms to be used during the CVE Acquisition Phase as follows:

- Extracting the TargetName, TargetGroup, and TargetVersion information (software package identifiers) from the SBOM files.
- Allowing the user to add additional keywords to be employed in the search activity.
- Allowing the user to activate AI search term enhancement, which generates a supplemental list of semantically similar words to the search.

Hence, the Final Keyword List = SBOM keywords + user supplied keywords (optional) + AI enhanced keywords (optional).

Reference Index Generation Phase: Per [Step 2], the SBOM file is sent to a Vector Embedding Model and saved as an Azure Search Reference Index to be used during the

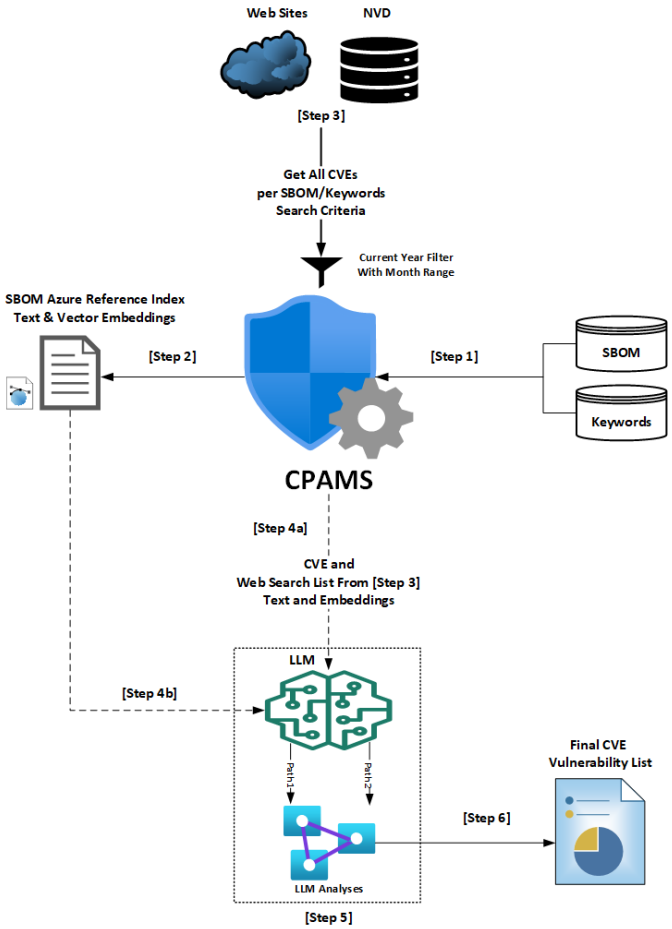

Fig. 1. CPAMS Process

LLM Analysis Phase.

CVE Acquisition Phase: This search and discovery phase per [Step 3] uses the Final Keyword List to get CVEs from the NVD and various websites that are user configurable. The raw search results can also be filtered (the finest resolution is a month within a calendar year) and then saved in memory. The NVD and Tavily [12] Web search activities use traditional keyword search techniques only.

Analysis Prep Phase: This preparatory phase per [Step 4a] involves converting CVE Descriptions found during the CVE Acquisition Phase into vector embeddings for LLM analysis.

LLM Analysis Phase: This phase per [Step 5] includes the use of the following:

- CVE descriptive text and vector embeddings from [Step 4a].
- SBOM Azure Reference Index from [Step 4b] which includes both SBOM Components Text and vector embeddings.

This phase consists of two independent analyses designated as Path 1 and Path 2 as follows:

Path 1: This is an analysis with CVE Descriptions and SBOM Components as text.

- It feeds CVE IDs and associated CVE Descriptions along with the SBOM Components directly to an LLM.
- It leverages an LLM's inherent knowledge of software vulnerabilities and component relationships to generate correlations.

CVE Description Text + SBOM Components Text → LLM → AI-Generated Path 1 Analysis.

Path 2: This is a purely computational analysis with no LLM reasoning, just semantic similarity scoring.

- It creates vector embeddings of CVE Descriptions.
- It creates vector embeddings of SBOM Components (TargetName, TargetGroup, TargetVersion).
- It uses mathematical similarity to find matches above a threshold.

CVE Description Embeddings + SBOM Components Embeddings → Semantic Similarity (Cosine) → AI-Generated Path 2 Analysis

Report Phase: This phase per [Step 6] synthesizes the AI analysis results into a final list of vulnerabilities.

IV. DATASET DESCRIPTION

For the research project, CPAMS was realized as a customized Python application implemented to interrogate the NVD and various websites using keywords and production SBOMs identifying software packages in fielded medical devices. The generated raw CVE list was then filtered based on a date range. The CPAMS feature set included connectivity to an LLM and a Vector Embedding Model that enabled the Dual-Path AI Analysis as explained. The validation approach was to mine the past 10 years of CVEs from NVD by executing the previously described phases and confirm that the final AI generated list from the Report Phase included all CVEs found since 2015 based on specific organizational data that was compiled using the existing corporate search process and which is identified as the ground truth or golden reference CVE set.

V. RESULTS AND DISCUSSIONS

Experiment 1 was performed using the following configuration:

Configuration Settings:

- GPT-5– Large Language Model.
- OAI EML – Open AI Embedding Model Large[10].
- AZ – Azure Vector Storage.
- SST – A RAG algorithm based on cosine similarity that returns only retrieved documents that exceed a specified Similarity Score Threshold [11].

The results are presented in Table I and Table II.

Analysis ID	Number of CVEs
Pre-AI Analysis Raw Search CVEs	10307
Post AI CVEs	706
Number Golden Matches	11 of 11
False Negatives	0
False Positives	695

TABLE I
SEMANTIC CORRELATION WITH [.35 SST] RESULTS

Analysis ID	Number of CVEs
Pre-AI Analysis Raw Search CVEs	10307
Post AI CVEs	107
Number Golden Matches	7 of 11
False Negatives	4
False Positives	100

TABLE II
SEMANTIC CORRELATION WITH [.40 SST] RESULTS

Experiment 2 was performed using the following configuration, where the only difference between Experiment 1 was the Embedding Model which could potentially have a direct impact on the SST.

Configuration Settings:

- GPT-5
- OAI EMS – Open AI Embedding Model Small[10].
- AZ
- SST

The results presented in Table III and Table IV reveal identical numbers. Hence, the large vs. small Embedding Model had no impact. This is not surprising or concerning given the difference is essentially doubling the high-dimensional vector space. There are some cases when this matters; yet as shown in [13] it amounts to increasing the Massive Text Embedding Benchmark (MTEB) scores a couple of points. For example, 64.6 vs. 62.3 respectively.

Analysis ID	Number of CVEs
Pre-AI Analysis Raw Search CVEs	10307
Post AI CVEs	706
Number Golden Matches	11 of 11
False Negatives	0
False Positives	695

TABLE III
SEMANTIC CORRELATION WITH [.35 SST] RESULTS

Analysis ID	Number of CVEs
Pre-AI Analysis Raw Search CVEs	10307
Post AI CVEs	107
Number Golden Matches	7 of 11
False Negatives	4
False Positives	100

TABLE IV
SEMANTIC CORRELATION WITH [.40 SST] RESULTS

These results highlight that the key point of discussion is the SST value and not the Embedding Model. With an SST value of .35 there was approximately a 15:1 reduction of CVEs (from over 10,000 to 706). Moreover, with this SST, all target or ground truth CVEs were a proper subset of the AI analysis results. With a higher SST of .40, the False Positive number was significantly reduced but there were 4 False Negatives. Therefore, the SST value of .35 was necessary for a successful experiment with 0 False Negatives. While the False Positives are a detractor, it should be emphasized that these experiments address the worst-case since 10 years of CVE data was extracted in order to conduct a validation of CPAMS. Thus, using CPAMS on a quarterly basis to identify a most recent 90-day window of CVEs, the actual CVE raw list and the AI analysis generated output is a much reduced list as shown in Table V. This table is not meant to capture the notion of True Positives or False Positives, rather an example of how CPAMS reduces the analysis list from 111 to 21 items. Thus, it is representative of the number of CVEs that would need further analysis and review every quarter.

Analysis ID	Number of CVEs
Pre-AI Analysis Raw Search CVEs	111
Post AI CVEs	21

TABLE V
90-DAY SEARCH RESULTS (JULY-SEPT 2025)

VI. CONCLUSION

CPAMS represents an advancement from prior cybersecurity vulnerability research and management, addressing limitations noted in the use of CyberVulModelEval [1] by integrating multi-source data with artificial intelligence. This composite architecture may enhance organizational competence in advanced threat awareness while reducing manual effort and improving accuracy. The performance analysis demonstrated substantial improvements in vulnerability discovery efficiency and False Positive reduction, establishing CPAMS as a valuable tool for modern cybersecurity vulnerability monitoring and management. The results of this case study demonstrate the practical effectiveness in a specific organizational context and provides a foundation for continued innovation and capability enhancement, ensuring long-term value and adaptability to evolving cybersecurity monitoring challenges.

REFERENCES

[1] Rigdon, G. [2025]. Evaluating LLM Use In Identifying OSS Packages Linked to Cybersecurity Vulnerabilities for HAVOSS.

[2] Yao, S., Zhao, J., Yu, D. Du, N. Shafran, I., Narasimhan, K., Cao, Y. (2023, March.) ReAct: Synergizing Reasoning And Acting In Language Models. arvix.org. https://arxiv.org/pdf/2210.03629

[3] Asai, A., Wu, Z., Wang, Y., Sil, A., Hajishirzi. H. (2023, October). Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. arvix.org. https://arxiv.org/abs/2310.11511

[4] Rackauckas, Z. (2024, February). RAG-Fusion: A New Take On Retrieval-Augmented Generation. arvix.org. https://arxiv.org/pdf/2402.03367

[5] Yan, S., Gu, J., Zhu, Y., Ling, Z. (2024, October). Corrective Retrieval Augmentation Generation. arvix.org. https://arxiv.org/pdf/2401.15884

[6] Bideh, P., Host, M., Hell, M. (2018, November). HAVOSS: A Maturity Model for Handling Vulnerabilities in Third Party OSS Components. researchgate.net. https://www.researchgate.net/publication/328689964

[7] NIST National Vulnerability Database (2025). CVEs and the NVD Process. nvd.nist.gov. https://nvd.nist.gov/general/cve-process

[8] Hughes, C. (2024, August). Vulnerability Exploitation in the Wild. resilientcyber.io. https://www.resilientcyber.io/p/vulnerability-exploitation-in-the

[9] Zhang, J., Bu, H., Wen, H., Liu, Y., Fei, H., Xi, R., Li, L., Yang, Y., Zhu, H., Meng, D. (2025, February). When LLMs meet cybersecurity: a systematic literature review. cybersecurity. springeropen.com. https://cybersecurity.springeropen.com/articles/10.1186/s42400-025-00361-w

[10] https://platform.openai.com/docs/guides/embeddings

[11] LangChain (2025). How to add scores to retriever results. python-langchain.com. https://python.langchain.com/docs/

[12] Tavily API Documentation. (2023). Web Search Intelligence Platform. Retrieved from https://tavily.com/

[13] Mauri, D. (2024, December). Embedding models and dimensions: optimizing the performance to resource-usage ratio. devblogs.com. https://devblogs.microsoft.com/azure-sql/embedding-models-and-dimensions-optimizing-the-performance-resource-usage-ratio/

[14] Xu, S. (2024, March). A Guide to Open-Source Embedding Models. bentoml.com. https://www.bentoml.com/blog/a-guide-to-open-source-embedding-models