Enhancing Vulnerability Management with Artificial Intelligence Algorithms

Gabriela TOD-RĂILEANU, Ana-Maria DINCĂ, Sabina-Daniela AXINTE, Ioan C. BACIVAROV
Faculty of Electronics, Telecommunications and Information Technology,
National University of Science and Technology POLITEHNICA Bucharest, Romania
gabriela.tod@stud.etti.upb.ro, ana_maria.dinca@stud.etti.upb.ro, axinte_sabina@yahoo.com, ioan.bacivarov@upb.ro

Abstract

The rising number of vulnerabilities, highlights the growing cybersecurity challenges and the need for robust vulnerability management. This paper examines the role of Artificial Intelligence in enhancing vulnerability detection and management, focusing on scalable and accurate solutions to address large-scale codebase analysis. AI-driven techniques bridge traditional static analysis and advanced detection, uncovering hidden vulnerabilities and improving efficiency. Future research should optimize these tools for diverse languages, Secure Software Development Life Cycle workflows, and predictive threat analysis. These advancements highlight AI's potential to strengthen software security in an increasingly complex threat landscape.

Index terms: vulnerability management, Artificial Intelligence, code scanning, Secure Software Development Lifecycle, vulnerability detection

1. Introduction

According to the National Vulnerability Database (NVD) maintained by NIST, 28961 new Common Vulnerabilities and Exposures (CVEs) were published in 2023 and there are already 29004 new CVEs published by November in 2024 [1]. Even though the year is not yet completed, it can be observed an increase of approximately 0.15% and this underscores the growing challenges in cybersecurity and the critical importance of robust vulnerability management practices. This paper analyzes the current implementation of Artificial Intelligence (AI) in vulnerability detection and management. The goal is to understand if there are any solutions, proprietary or open-source software, that can help in a world where only one critical vulnerability has affected over 2000 organizations [2] and had an estimated cost of over 15 million USD.

2. Vulnerability management

It is important to understand what a vulnerability in cybersecurity is: a weakness or flaw in a system, application, network, or process that can be exploited by attackers to compromise its security. Vulnerabilities can lead to unauthorized access, data theft, system disruptions, or other malicious activities.

Vulnerabilities arise from flaws in software development, such as unvalidated input, weak authentication, or insecure configurations [3], which can be exploited by attackers to execute malicious actions or gain unauthorized access. These weaknesses can compromise key security objectives: confidentiality (exposing sensitive data), integrity (altering or corrupting data), and availability (disrupting operations or causing denial-of-service). Effective mitigation is essential to maintain system security and resilience.

Vulnerability management constitutes a continuous process of identifying, assessing, prioritizing, and addressing vulnerabilities in an organization's systems, applications, and networks. The objective is to mitigate the risk of exploitation and enhance the overall security posture by proactively addressing weaknesses before potential exploitation by malicious actors. It is noteworthy that this process requires ongoing improvement due to the escalating number of threats, and scalability represents one of the most significant factors influencing modifications in the vulnerability management process.

The vulnerability management process encompasses several critical components, which can be compared to the framework established for Secure Software Development Lifecycle (SSDLC). However, organizations may opt to adapt this process to their specific requirements. The key components are:

- **Identification**: Utilize tools such as vulnerability scanners to detect weaknesses across systems, networks, and applications.
- **Assessment**: Evaluate the severity of detected vulnerabilities using frameworks such as the Common Vulnerability Scoring System (CVSS). Determine the probability of exploitation and the potential impact on the organization.
- **Prioritization**: Rank vulnerabilities based on risk factors including exploitability, asset criticality, and exposure (e.g., public-facing systems). Allocate resources to address high-risk vulnerabilities as a priority.
- **Remediation**: Implement fixes such as software patches, updates, or configuration changes to resolve vulnerabilities. If immediate remediation is not feasible, implement temporary mitigation measures to minimize risk.
- **Validation**: Verify that vulnerabilities have been resolved and no residual risks remain.
- **Reporting and Continuous Improvement**: Document findings, actions taken, and improvements made for compliance and future reference. Continuously refine processes based on lessons learned and emerging threats.

This analysis will focus on the contributions of Artificial Intelligence to the Identification and Assessment phases of vulnerability management. Specifically, it will examine how AI-driven tools enhance the detection of vulnerabilities by automating the scanning of complex systems, analyzing codebases, and identifying patterns of weaknesses more efficiently than traditional methods. In the Assessment phase, the analysis will explore how AI leverages predictive algorithms, machine learning models, and real-time data to prioritize vulnerabilities based on exploitability, potential impact, and business context. By addressing these areas, this study aims to highlight how AI not only accelerates the identification and assessment processes but also improves accuracy, scalability, and decision-making in vulnerability management.

3. Artificial Intelligence used in Vulnerability Detection

In this study, a structured approach for vulnerability detection is proposed, by categorizing it into two primary domains: infrastructure scanning and code scanning. The first domain, **infrastructure scanning**, encompasses the identification of vulnerabilities across network devices, servers, cloud environments, and other components that form the backbone of IT systems. This approach ensures a robust and resilient infrastructure against threats.

The second domain, **code scanning**, delves into analyzing the application layer, specifically the source code and software components, to identify security flaws during development. This category includes methods such as static and dynamic code analysis, which are critical for addressing vulnerabilities before deployment. By segmenting the topic into these two areas, it is aimed to provide a comprehensive understanding of vulnerability detection, addressing both the foundational infrastructure and the applications built upon it.

From an infrastructure scanning perspective, several industry leaders have established a strong reputation for their expertise, extensive research, and innovative solutions in this domain. Notable among these are: Tenable IO with "Tenable One: The world's only AI-powered exposure management platform" [4], Qualys with Qualys VMDR (Vulnerability Management, Detection, and Response) [5] and Rapid7 [6]. Complementing these proprietary offerings is an open-source tool with significant potential: Trivy [7]. One of the greatest advantages identified for the open-source tool mentioned before is that by utilizing Trivy as a security scanner, developers become more aware of security issues as they arise. This proactive approach enables them to address vulnerabilities promptly, reducing the likelihood of security problems being discovered late in the development process [8]. This feature positions Trivy as an invaluable tool for bridging the gap between development and security in modern DevOps practices.

From the code scanning perspective, there are some industry leaders that have already enhanced their tools with AI. Notable mentions are: Snyk Code that utilizes AI to provide real-time static application security testing (SAST), which offers immediate feedback on code vulnerabilities, enabling developers to identify and fix issues during the development process [9]. GitHub's CodeQL is a semantic code analysis engine that uses AI to detect vulnerabilities across codebases. It allows developers to write custom queries to find specific vulnerabilities and integrates seamlessly with GitHub workflows [10]. There are also new companies that have become popular, such as: Armur AI, that employ Large Language Models (LLMs) to perform static code analysis, identifying vulnerabilities and providing remediation suggestions [11]. The researchers have also contributed to the list with multiple tools such as Vulnhuntr [12], VulBERTa [13] and VulCNN. VulCNN is an image-inspired scalable vulnerability detection system that applies deep learning techniques to source code analysis. By converting code into image-like representations, VulCNN utilizes convolutional neural networks (CNNs) to identify potential vulnerabilities [14].

4. Exploring the Capabilities of Open-Source Scanning Tool

This section provides a comprehensive examination of VulCNN's underlying mechanisms, its key strengths, and its potential applications, emphasizing its contributions to the field of vulnerability detection.

4.1. Design of VulCNN

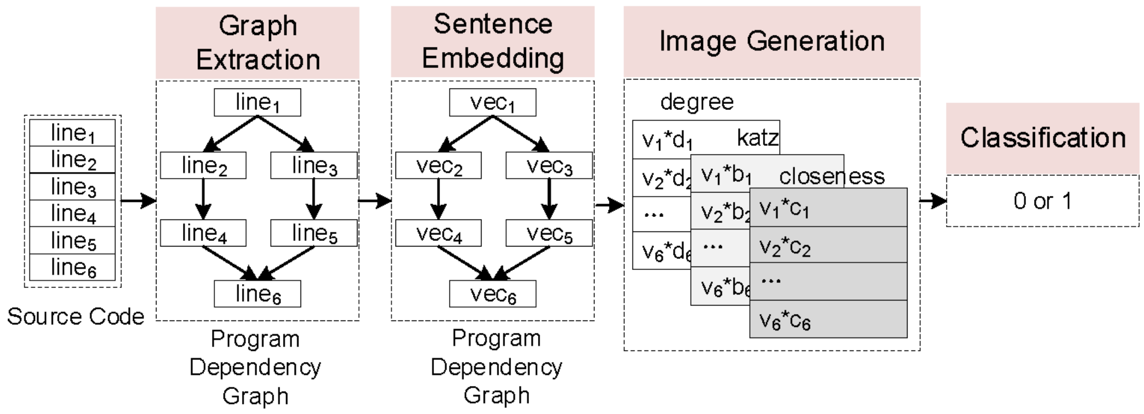


Fig. 1. Pipeline of VulCNN: From Source Code to Vulnerability Classification [14]

The tool operates through four key phases:

- Graph Extraction: Starting with the source code of a function, the process involves normalizing the code and performing static analysis to construct the program dependency graph for the function.
- Sentence Embedding: In this phase, each node in the program dependency graph, representing a line of code, is treated as a sentence. These lines are then embedded into corresponding vectors, capturing their semantic and structural information.
- Image Generation: Following the embedding phase, centrality analysis is applied to determine the relative importance of each line of code. These centrality scores are multiplied by the respective vectors, and the resulting data is formatted as an image.
- Classification: In the final phase, the generated images are used to train a convolutional neural network (CNN). Once trained, CNN classifies the images to detect whether vulnerabilities exist within the code.

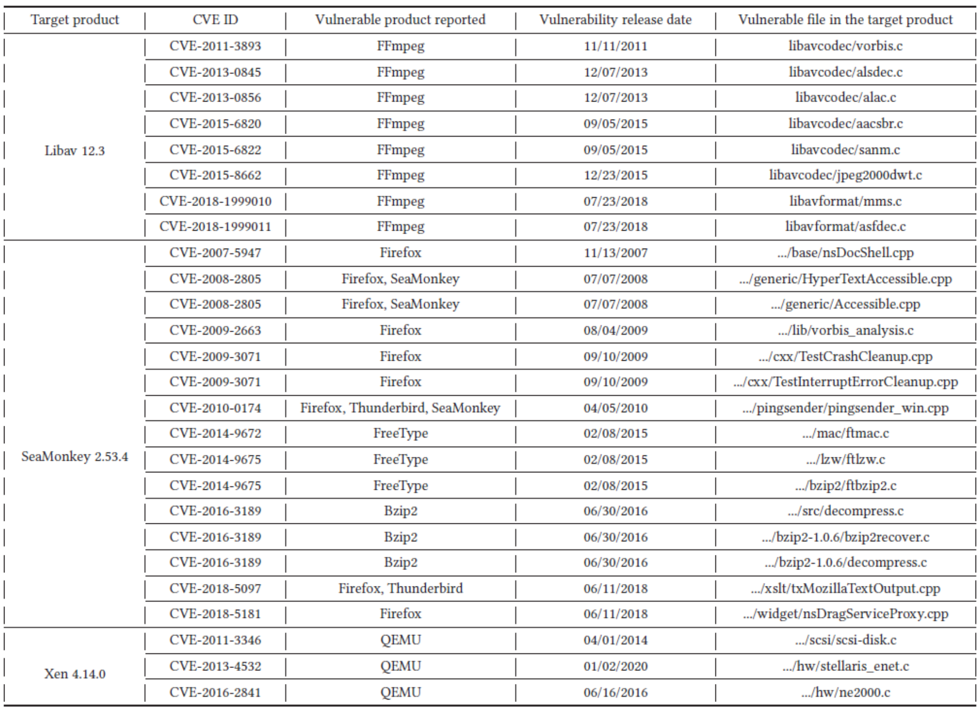
4.2. Dataset

The dataset used is sourced from the Software Assurance Reference Dataset (SARD) [15], a project maintained by the National Institute of Standards and Technology (NIST) [16]. SARD provides a comprehensive collection of production, synthetic, and academic security flaws (referred to as "bad functions") alongside a large set of "good functions." Since the focus is on detecting vulnerabilities in C/C++ code, there were selected only the C/C++ functions from SARD. The dataset includes 12,303 vulnerable functions and 21,057 non-vulnerable functions.

4.3. Results of the case study

There were selected three open-source applications for the test: Libav [17], Xen [18], and Seamonkey [19]. The analysis reveals that 73 warnings align with known vulnerability patterns listed in the NVD. Among these identified vulnerabilities, 17 have been "silently" patched by vendors in the latest versions of their respective products, the code related to four vulnerabilities has been removed, and the remaining 52 vulnerabilities persist in the products [14].

Table 1. A part of the vulnerabilities discovered by VulCNN from the latest versions of the selected products [14]



Target product	CVE ID	Vulnerable product reported	Vulnerability release date	Vulnerable file in the target product
Libav 12.3	CVE-2011-3893	FFmpeg	11/11/2011	libavcodec/vorbis.c
	CVE-2013-0845	FFmpeg	12/07/2013	libavcodec/alsdec.c
	CVE-2013-0856	FFmpeg	12/07/2013	libavcodec/alac.c
	CVE-2015-6820	FFmpeg	09/05/2015	libavcodec/aacsbr.c
	CVE-2015-6822	FFmpeg	09/05/2015	libavcodec/sanm.c
	CVE-2015-8662	FFmpeg	12/23/2015	libavcodec/jpeg2000dwt.c
	CVE-2018-1999010	FFmpeg	07/23/2018	libavformat/mms.c
	CVE-2018-1999011	FFmpeg	07/23/2018	libavformat/asfdec.c
SeaMonkey 2.53.4	CVE-2007-5947	Firefox	11/13/2007	.../base/nsDocShell.cpp
	CVE-2008-2805	Firefox, SeaMonkey	07/07/2008	.../generic/HyperTextAccessible.cpp
	CVE-2008-2805	Firefox, SeaMonkey	07/07/2008	.../generic/Accessible.cpp
	CVE-2009-2663	Firefox	08/04/2009	.../lib/vorbis_analysis.c
	CVE-2009-3071	Firefox	09/10/2009	.../cxx/TestCrashCleanup.cpp
	CVE-2009-3071	Firefox	09/10/2009	.../cxx/TestInterruptErrorCleanup.cpp
	CVE-2010-0174	Firefox, Thunderbird, SeaMonkey	04/05/2010	.../pingsender/pingsender_win.cpp
	CVE-2014-9672	FreeType	02/08/2015	.../mac/ftmac.c
	CVE-2014-9675	FreeType	02/08/2015	.../lzw/ftlzw.c
	CVE-2014-9675	FreeType	02/08/2015	.../bzip2/ftbzip2.c
	CVE-2016-3189	Bzip2	06/30/2016	.../src/decompress.c
	CVE-2016-3189	Bzip2	06/30/2016	.../bzip2-1.0.6/bzip2recover.c
	CVE-2016-3189	Bzip2	06/30/2016	.../bzip2-1.0.6/decompress.c
	CVE-2018-5097	Firefox, Thunderbird	06/11/2018	.../xslt/txMozillaTextOutput.cpp
	CVE-2018-5181	Firefox	06/11/2018	.../widget/nsDragServiceProxy.cpp
Xen 4.14.0	CVE-2011-3346	QEMU	04/01/2014	.../scsi/scsi-disk.c
	CVE-2013-4532	QEMU	01/02/2020	.../hw/stellaris_enet.c
	CVE-2016-2841	QEMU	06/16/2016	.../hw/ne2000.c

5. Conclusions and future scope

Advancements in vulnerability detection systems, such as those demonstrated by tools like VulCNN, highlight the ongoing progress in leveraging innovative methodologies to improve software security. By combining scalability and accuracy, modern approaches are addressing critical challenges in detecting vulnerabilities in large-scale codebases. The integration of techniques like converting source code into representations suitable for deep learning analysis has proven effective in bridging the gap between traditional static analysis and cutting-edge AI-driven solutions. These advancements have not only outperformed existing tools in accuracy and speed, but also showcased the potential to uncover previously unreported vulnerabilities, emphasizing their practical relevance in real-world applications.

Future research should aim to expand on these developments by exploring alternative AI architectures, such as hybrid models, to further refine the detection process. Additionally, optimizing methodologies to handle diverse programming languages and integrating such tools seamlessly into software development workflows, including Continuous Integration/Continuous Deployment (CI/CD) pipelines, will be crucial for fostering a proactive approach to security. Another promising avenue is incorporating threat intelligence data to predict emerging vulnerabilities and enhance real-time detection capabilities.

In conclusion, the field of vulnerability detection continues to evolve rapidly, with tools like VulCNN setting a benchmark for innovation. The combination of scalability, accuracy, and adaptability in these systems highlights the potential for developing more comprehensive and efficient solutions, paving the way for a more secure software ecosystem in the face of ever-increasing security challenges.

References

[1]. "CVE metrics," CVE org, 2023. [Online]. Available: https://www.cve.org/about/Metrics. [Accessed 10 November 2024].

[2]. Z. Simas, "Unpacking the MOVEit Breach: Statistics and Analysis," EMSISoft, 18 July 2023. [Online]. Available: https://www.emsisoft.com/en/blog/44123/unpacking-the-moveit-breach-statistics-and-analysis/. [Accessed 10 November 2024].

[3]. OWASP, "OWASP Top Ten," OWASP, [Online]. Available: https://owasp.org/www-project-top-ten/. [Accessed 1 November 2024].

[4]. Tenable IO, "Tenable One," Tenable IO, October 2022. [Online]. Available: https://www.tenable.com/products/tenable-one. [Accessed 1 November 2024].

[5]. "Leveraging AI-informed Cybersecurity to Measure, Communicate, and Eliminate Cyber Risk," Qualys, 9 November 2023. [Online]. Available: https://blog.qualys.com/qualys-insights/qualys-security-conference/2023/11/09/leveraging-ai-informed-cybersecurity-to-measure-communicate-and-eliminate-cyber-risk. [Accessed 12 November 2024].

[6]. K. Lynas-Blunt, "Securely Build AI/ML Applications in the Cloud with Rapid7 InsightCloudSec," Rapid7, 22 December 2023. [Online]. Available: https://www.rapid7.com/blog/post/2023/12/22/securely-build-ai-ml-applications-in-the-cloud-with-rapid7-insightcloudsec/. [Accessed 1 November 2024].

[7]. "The all-in-one open source security scanner," AquaSec, [Online]. Available: https://trivy.dev. [Accessed 28 October 2024].

[8]. Panca, Rizki, Perkasa., Evangs, Mailoa, "Adopsi devsecops untuk mendukung metode agile menggunakan trivy sebagai security scanner docker image dan dockerfile," Jurnal Indonesia : Manajemen Informatika dan Komunikasi, vol. 4, no. 3, pp. 856-863, 2023.

[9]. "Snyk Code: Developer-focused, real-time SAST," Snyk, [Online]. Available: https://snyk.io/product/snyk-code/. [Accessed 1 November 2024].

[10]. "Finding security vulnerabilities and errors in your code with code scanning," GitHub, 2024. [Online]. Available: https://docs.github.com/en/code-security/code-scanning. [Accessed 12 November 2024].

[11]. A. Sharma, "What Is a Code Vulnerability Analyzer?," Armur AI, 3 September 2024. [Online]. Available: https://armur.ai/blogs/posts/code_vulnerability_analyzer/. [Accessed November 2024].

[12]. D. McInerney, M. Salvati, "Vulnhuntr GitHub repository," ProductAI, 9 November 2024. [Online]. Available: https://github.com/protectai/vulnhuntr. [Accessed 14 November 2024].

[13]. H. Hanif, S. Maffeis, "Vulberta: Simplified source code pre-training for vulnerability detection," in International Joint Conference on Neural Networks (IJCNN), 2022.

[14]. Y. Wu, D. Zou, S. Dou, S. Dou, W. Yang, D. Xu, H. Jin, "VulCNN: An Image-inspired Scalable Vulnerability Detection System," 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE), pp. 2365-2376, 2022.

[15]. "NIST Software Assurance Reference Dataset," Software Assurance Metrics And Tool Evaluation, [Online]. Available: https://samate.nist.gov/SARD. [Accessed 10 November 2024].

[16]. "National Institute of Standards and Technology," US Department of Commerce, [Online]. Available: https://www.nist.gov. [Accessed 10 November 2024].

[17]. "Libav GitHub repository," Libav, [Online]. Available: https://github.com/libav/libav. [Accessed 15 October 2024].

[18]. "Xen Project archives," Xen Project, [Online]. Available: https://xenproject.org/xen-project-archives/. [Accessed 1 November 2024].

[19]. "Seamonkey Project," [Online]. Available: https://www.seamonkey-project.org/. [Accessed 15 October 2024].