

Discernment and Enumeration of Security Vulnerabilities Present in Docker Images

Shraddha Chopra¹, Sreedevi A. G² and Udhayakumar Shanmugam³

¹⁻³Department of Computer Science and Engineering, Amrita School of Computing,
Amrita Vishwa Vidyapeetham, Chennai, India

Email: shraddha5chopra@gmail.com, agsreedevi@gmail.com, s_udhayakumar@ch.amrita.edu

Abstract—Docker containers have emerged pivotal in the packaging, distributing and deployment of applications ushering into a new era of flexibility and scalability across diverse computing environments. Consequently, this widespread integration has led to formidable challenges in the security domain. These are primarily concerned with the detection and remediation of vulnerabilities inherently present within the Docker images. This study presents an automated pipeline for Docker image analysis, focusing on vulnerability detection and severity assessment. Employing a methodical research framework, the study undertakes a systematic scrutiny of Docker images sourced from the most comprehensive antecedent, DockerHub. The images are then examined using a blend of manual inspection and automated vulnerability scanning methodologies, aiming to enhance the security posture by identifying and categorizing vulnerabilities based on severity, and to provide insights into the image composition. The threats are then cross-referenced with the common vulnerabilities and exposures (CVE) database to ascertain their presence in known security advisories and to facilitate accurate risk assessment and mitigation strategies.

Index Terms— Docker security, CVEs, DockerHub, Docker Images, Containerization, DockerFile, DevSecOps, CI/CD pipelines.

I. INTRODUCTION

In the landscape of modern software development, containerization technologies, particularly Docker, have revolutionized the deployment and management of applications by offering unprecedented levels of agility, scalability, and efficiency to development teams worldwide. Docker has become a cornerstone of modern DevOps practices, enabling developers to package applications and their dependencies into self-contained units known as Docker images, which can be deployed as Docker containers across various computing environments. The burgeoning ecosystem of open-source software components and the rapid evolution of Docker itself contribute to the intricacy of securing Docker images. Since, despite its widespread adoption, Docker's architecture brings forth a new frontier of security challenges [1], primarily stemming from the construction of Docker images and the runtime isolation of Docker containers. It is therefore imperative to delve into the critical aspects of Docker security, focusing on the identification of vulnerabilities within Docker images, and the corresponding Common Vulnerabilities and Exposures (CVEs) associated with these weaknesses. Before the exploration of these flaws, it is essential to understand the intricacies of Docker image construction and the security considerations therein. Central to Docker's architecture is the DockerFile, which is a text-based script

that defines the configuration and dependencies of a Docker image. The DockerFile [2] specifies the instructions for building the image layer by layer, incorporating base operating systems, application frameworks, libraries, and the application code. Whilst, Docker simplifies the process of image creation, inadequate security practices during DockerFile composition can instigate vulnerabilities that may compromise the integrity and confidentiality of the resulting Docker image. Common pitfalls [3] include the inadvertent inclusion of unnecessary packages, outdated dependencies, and misconfigurations, which can expose the image to known exploits and attacks. Moreover, these Docker images then serve as the foundation for Docker containers, encapsulating the application and its dependencies into a portable and reproducible format. Containers, [4] in turn, provide an isolated runtime environment for the smooth execution of these images, ensuring consistency across different computing environments. However, the reliance on shared kernel resources and the dynamic nature of containerized workloads [5] introduces security considerations that must be addressed to mitigate the potential risks effectively. While significant strides have been made in understanding and mitigating security risks associated with Docker, several challenges still persist in securing [6] containerized environments effectively. Foremost among these challenges is the dynamic nature of containerized workloads, which necessitates continuous monitoring and remediation of vulnerabilities throughout the container lifecycle. Additionally, the proliferation of third-party Docker images from public repositories like DockerHub introduces uncertainties regarding the trustworthiness and integrity of containerized applications. Vulnerability assessment [7] plays a crucial role in analysing the Docker images for security weaknesses. Conventional vulnerability scanning tools [8], while effective in traditional environments, often falter in the dynamic realm of containerization. This deficiency leaves organizations vulnerable, with yawning gaps in their security posture and an elevated risk of exploitation by threat actors. Advanced tools [9] such as Anchore, Clair, Grype, and Trivy leverage comprehensive databases [10] of CVEs, which catalogues the vulnerabilities identified in software components, including operating systems, libraries, and application dependencies. By scanning the Docker images against CVE databases, organizations can proactively identify and remediate vulnerabilities before deploying containers into production environments, thereby minimizing the risk of exploitation and data breaches. This research endeavours to bridge this chasm by embarking on a comprehensive analysis of security vulnerabilities residing within Docker images. Through the judicious application of cutting-edge vulnerability scanning techniques and methodologies, our aim is to forge a systematic approach to not only identify but also categorize and meticulously list the array of vulnerabilities within Docker images.

II. LITERATURE REVIEW

The current approach to identifying flaws and vulnerabilities in Docker images predominantly relies on static analysis, vulnerability scanning, and manual inspection. While these methods offer some level of protection, they also come with notable limitations and gaps. Addressing these gaps requires the development and adoption of more advanced and comprehensive approaches to threat discernment and enumeration in Docker images. Emphasizing the evolution and limitations of the current systems, this section identifies gaps in Docker Containerization and Image Security, highlighting the need for more robust, adaptive, and efficient solutions. Scanning Docker images for vulnerabilities and security issues is crucial for ensuring the integrity and safety of containerized applications. Numerous tools are available for this objective, each presenting distinct features and functionalities. In order to test the strengths and weaknesses of some of these tools, a comprehensive examination of 59 container images using three distinct open source container scanners: Clair, Anchore, and Microscanner was done in [11]. The findings highlight the differences in both: the scope and the precision among the evaluated tools, emphasizing the benefits of employing a blend of static and dynamic analyses. A similar tool, the Docker Image Vulnerability Diagnostic System (DIVDS) was developed by Soonhong Kwon et al. [12] with the aim of enabling users to assess the vulnerability status of individual Docker images via a systematic evaluation framework. This evaluation methodology hinges on analysing the interplay between vulnerable software packages and pertinent vulnerability data contained within Docker images. DIVDS serves as a robust mechanism for ensuring the integrity of Docker-based application development environments, effectively safeguarding against the inadvertent acquisition or dissemination of vulnerable Docker images within repository ecosystems. Similarly, [13] introduces SEAF (Scalable, Efficient, Application-independent Framework), which employs differential analysis of Docker image layers to swiftly detect alterations, thereby diminishing redundant analysis of identical layers across numerous images. The researchers utilized SEAF to amass metadata from over 12 million images, yet could present findings for only a limited selection of case studies with scant elaboration on the vulnerabilities themselves. Additionally, the authors conducted comparative analyses using two distinct scanners, Clair and Dagda. Notably, Dagda has not received updates for nearly three

years (as of April 2024) and is plagued by several unresolved issues, potentially hindering its effectiveness. Moreover, Anchore, previously employed in numerous studies, ceased support in 2023 and has been partially succeeded by Gripe, underscoring the necessity for multiple scanning tools to facilitate comparative analyses, particularly as certain open-source tools may lack up-to-date maintenance. Taking the structural route, Manish K A et al. [14] have introduced an architectural framework aimed at fortifying the security posture of Docker containers. This framework incorporates a dedicated plugin designed to seamlessly integrate with a Continuous Integration/Continuous Deployment (CI/CD) pipeline, thereby orchestrating the deployment process while concurrently enforcing rigorous security measures. From the inception of application development to deployment, the architecture meticulously safeguards against potential vulnerabilities, with the inclusion of a plugin specifically tailored for Docker image construction. Applications are encapsulated within images alongside essential dependencies, subsequently deposited into a Docker registry. The framework scrutinizes threats across both static and dynamic resource allocations, employing an entirely automated implementation devoid of manual intervention. Furthermore, Vipin Jain et al. [15] have proposed a mechanism for applying Docker security levels, facilitating the evaluation of diverse platforms and quantifying security attributes pertinent to resource accessibility. Transitioning to the assessment criteria, [16] examine the prevalence of reported vulnerabilities and their actual exploitation, aiming to identify trends and models that can predict which vulnerabilities are more susceptible to exploitation. It advocates for a shift away from relying solely on vulnerability severity assessments and exploring factors such as the software vendor, attack type, and exploit availability, which influences the likelihood of exploitation. Despite the influence of Common Vulnerability Scoring System (CVSS) scores on exploit probability, ranging from 1.3% for a score of 2 to 18.4% for a score of 9, the authors emphasize the importance of considering broader contextual factors. They propose an Exploit Prediction Scoring System (EPSS) in their subsequent work [17] to address the limitations of the CVSS framework, highlighting the necessity for a more comprehensive approach to vulnerability assessment and mitigation. While the aforementioned studies are informative, they often only offer a single picture of the security posture of container images as of the time of writing, and they fail to take into account how containers are evolving as real-time software installations that might receive upgrades in the future. With the growing significance of image security, particularly as foundational components of applications, concerns regarding vulnerabilities have intensified. It is imperative to verify the security of images sourced from various origins, especially as development processes increasingly transition to cloud environments. Therefore, this study conducts a comprehensive analysis and aims at shedding light on the common vulnerabilities prevalent in Docker images and listing them, since these images encompass a wide range of potential security risks that may compromise the integrity and confidentiality of containerized applications.

III. PROPOSED SYSTEM

This section delineates the cohesive framework designed to streamline the management and analysis of Docker container images, focusing on vulnerability assessment and package scrutiny. Through rigorous analysis and validation of findings, the system will facilitate the generation of actionable insights and recommendations to enhance Docker image security effectively.

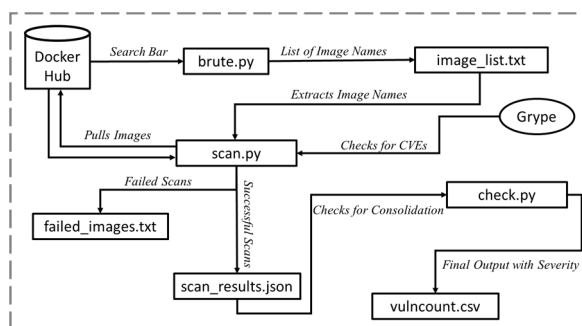


Figure 1. Proposed Architecture

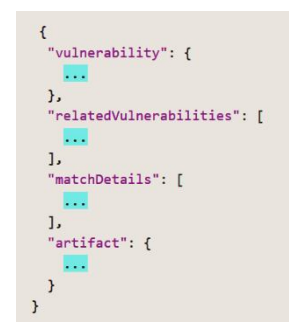


Figure 2. Basic Gripe Vulnerability Data Shape

A. Technological Stack and Workflow Orchestration

The workflow (refer to Fig. 1) of the proposed system is meticulously designed to ensure seamless execution of image management and vulnerability analysis tasks. It commences with the automated generation of a diverse array of Docker container image names from the DockerHub registry, systematically querying based on

predefined criteria. Leveraging custom scripts, this process guarantees comprehensive coverage, minimizing the likelihood of overlooking critical images. Subsequently, Docker's command-line interface (CLI) is invoked programmatically to pull the identified images sequentially, ensuring efficient retrieval from the repository. Concurrently, specialized vulnerability scanning tool Gripe (refer to Fig. 2) is engaged to conduct thorough assessments of each pulled image, meticulously scrutinizing packages and dependencies for known vulnerabilities. This phase is pivotal, as it forms the foundation for subsequent analysis and decision-making. Upon completion of the vulnerability scans, the analysis scripts meticulously parse the results, categorizing vulnerabilities based on severity levels and enumerating distinct vulnerabilities per image. This granular analysis provides stakeholders with actionable insights into the security posture of Docker container deployments. Additionally, structured reports and visualizations are generated using Python libraries for data manipulation and visualization, such as pandas and Matplotlib, facilitating clear communication of analysis findings to stakeholders. Through this orchestrated workflow, the proposed system empowers organizations to proactively identify and remediate security vulnerabilities, thereby enhancing the resilience and security posture of containerized environments.

IV. IMPLEMENTATION

This section delineates on the practical execution of the proposed system. It elucidates the methodical steps taken to automate image management, conduct vulnerability assessments, and derive actionable insights, illustrating the system's efficacy in fortifying security within Docker container environments.

A. Image Discovery

Image discovery is vital for containerization, necessitating a systematic exploration strategy to identify relevant Docker images. The approach utilized a custom Python script to orchestrate a comprehensive search on DockerHub. By iteratively querying Docker Hub's repository using a combination of alphabetical characters, broad coverage across diverse categories and functionalities was ensured, facilitating exhaustive exploration and maximizing image diversity for various deployment scenarios. The Python script executed API calls to retrieve metadata for official images based on specified filter criteria, systematically traversing DockerHub's repository and compiling a list of potential Docker images. This iterative process adhered to predefined search patterns, minimizing the risk of overlooking pertinent images. Additionally, error handling mechanisms were incorporated to manage API call failures, enhancing the robustness of the image discovery workflow. The image discovery process yielded a comprehensive list of Docker images meeting the specified filter criteria, encompassing various software stacks, frameworks, and utilities across domains such as web servers, databases, and development environments. The systematic approach enabled efficient exploration of DockerHub's repository, providing stakeholders with a rich selection of official images for containerization endeavours.

B. Image Pulling

The image pulling process automates the retrieval of the identified Docker images from DockerHub. The script deployed iteratively executes Docker pull commands for each identified image, streamlining the acquisition process and minimizing manual intervention. This automation not only enhances operational efficiency but also reduces the likelihood of human errors, ensuring reliability of the image pulling workflow.

```
failed to pull image: mysql
failed to pull image: java
Exception on: java
failed to pull image: aerospike
Exception on: aerospike
failed to pull image: elasticsearch
Exception on: elasticsearch
failed to pull image: consul
Exception on: consul
```

Figure 3. Image names on which the scan failed

Error Handling: Robust error handling mechanisms were integrated into the Python script to mitigate the impact of pulling failures, ensuring the continuity of the image acquisition process. In cases where a Docker image could not be successfully pulled, the script logged relevant information, such as the image name (refer to Fig. 3) and the nature of the failure. Furthermore, the script implemented retry strategies for transient errors, attempting to pull the image again after a brief delay.

C. Vulnerability Scanning

The vulnerability scanning process commenced with the invocation of Grype, targeting each pulled Docker image for assessment. Grype's scanning engine systematically analyzed the image's contents, detecting known vulnerabilities and generating detailed reports. Leveraging its database of vulnerability definitions, Grype cross-referenced package manifests and configuration files within the container, flagging instances of vulnerabilities and associated metadata. This systematic approach ensured thorough coverage of potential security risks inherent in containerized applications.

D. Analysis and Reporting

The analysis phase encompasses multiple facets, including quantifying vulnerability counts, categorizing vulnerabilities based on severity levels, and correlating vulnerabilities with respective Docker images. Through automated data processing, various metrics were extracted, such as the total number of vulnerabilities per image, distribution of vulnerabilities across severity levels, and identification of critical vulnerabilities. This analysis enabled a comprehensive understanding of the security posture of the containerized environment, facilitating informed decision-making regarding risk mitigation strategies and resource allocation.

```
{
  "geonetwork": {
    "matches": [
      {
        "vulnerability": {
          "id": "CVE-2010-1244",
          "severity": "Medium",
          "links": [
            "http://activemq.apache.org/activemq-531-release.html",
            "http://secunia.com/advisories/39223",
            "https://exchange.xforce.ibmcloud.com/vulnerabilities/57398",
            "https://issues.apache.org/activemq/browse/AMQ-2613",
            "https://issues.apache.org/activemq/browse/AMQ-2625"
          ],
          "description": "Cross-site request forgery (CSRF) vulnerability in createDestination.action in Apache ActiveMQ before 5.3.1 allows remote attackers to hijack the authentication of unspecified victims for requests that create queues via the JMSDestination parameter in a queue action.",
          "cvssV2": {
            "baseScore": 6.8,
            "exploitabilityScore": 8.6,
            "impactScore": 6.4,
            "vector": "AV:N/AC:M/Au:N/C:P/I:P/A:P"
          }
        }
      }
    ]
  }
}
```

Figure 4. A snippet of the findings of Docker Image 'geonetwork'

The findings from the vulnerability analysis were compiled into structured reports for dissemination and further action. Utilizing Python's data processing capabilities, the analysis results were formatted into several report formats, including JSON (refer to Fig. 4) and CSV. These reports provided stakeholders with clear insights into the security landscape of the Docker images, highlighting critical vulnerabilities, prevalent security risks, and recommended remediation measures. Additionally, severity-based categorization facilitated prioritization of vulnerability resolution efforts, ensuring efficient resource utilization and risk management.

V. RESULTS AND DISCUSSION

This section delves into the outcomes of the comprehensive experiments and analyses conducted on the Docker Images, yielding insightful results.

A. Vulnerability Counts

The analysis revealed a significant enumeration of vulnerabilities within the Docker images. By iterating through the retrieved images and conducting vulnerability scans using Grype, the script effectively quantified the extent of potential security risks. This comprehensive enumeration serves as a foundational step towards understanding the security landscape and prioritizing remediation efforts. The subsequent generation of a vulnerability count CSV file provides stakeholders with a clear overview (refer to Table I) of the vulnerability distribution within the Docker images, facilitating informed decision-making regarding risk mitigation strategies.

TABLE I. TOP FIVE IMAGES WITH THE GREATEST NUMBER OF VULNERABILITIES

Image Name	Number of CVEs present
rails	7341
iojs	6627
node	2997
silverpeas	2476
django	1897

It underscores the importance of rigorous security measures, including regular updates, patches, and vulnerability assessments, to mitigate the potential risks posed by this image. Moreover, it emphasizes the need

for vigilance and proactive measures in managing and securing Docker images, especially those with a considerable number of CVEs, to safeguard against potential security breaches and ensure the integrity and confidentiality of sensitive data and systems.

B. Package Use

The examination of package usage within Docker images provides valuable insights into the software composition and associated security risks. By parsing vulnerability data and aggregating package information, the script effectively identifies common software components and their respective vulnerabilities. This analysis (refer to Fig. 5) sheds light on potential points of weakness within the Docker ecosystem, guiding decision-making regarding package selection and dependency management. The subsequent generation of a package count CSV file offers stakeholders a comprehensive overview of package usage patterns and associated security implications, informing strategies for risk mitigation and system hardening.

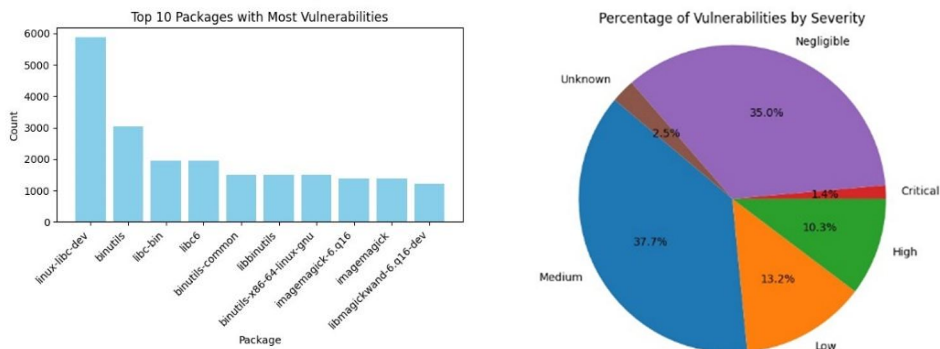


Figure 5. Top ten packages with most vulnerabilities Figure 6. Visualization of the Severity Analysis

C. Image Severity Assessment and Grouping

The study facilitates the grouping of Docker images based on the severity of vulnerabilities detected within each, providing stakeholders with a nuanced understanding of their security posture. By categorizing images according to severity levels ranging from Critical to Negligible (refer to Fig. 6), organizations can prioritize remediation efforts and implement targeted risk mitigation strategies. Additionally, the script's capability to generate severity assessment reports empowers stakeholders with actionable insights for enhancing security posture and mitigating potential threats. This comprehensive approach to severity assessment aligns with industry standards and best practices, facilitating informed decision-making and proactive risk management within Dockerized environments.

VI. CONCLUSION

In conclusion, the research provides a robust framework for conducting vulnerability assessments and enhancing security practices within Dockerized environments. Through systematic image retrieval, vulnerability scanning, and severity assessment, it offers valuable insights into the security landscape of Docker images. By leveraging Grype for vulnerability scanning and subsequent analysis, it facilitates the identification of vulnerabilities categorized by severity levels, ranging from Critical to Negligible. This enables stakeholders to prioritize remediation efforts effectively, focusing on addressing high-risk vulnerabilities and reducing the overall attack surface. Moreover, the experimental framework exhibits significant potential for future advancements in Docker security practices. Future iterations could focus on integrating continuous security practices, such as automated vulnerability scanning and real-time monitoring, into the CI/CD pipelines. Additionally, there is scope for enhancing vulnerability remediation strategies by automating patch management workflows and establishing baseline security standards for Docker images. Leveraging machine learning algorithms for threat prediction and integrating with SIEM systems for comprehensive security monitoring are also promising avenues for further development. Furthermore, fostering community collaboration and contributions to the codebase can facilitate innovation and drive continuous improvement in Docker security practices. By embracing these future directions, organizations can strengthen their security posture, adapt to evolving threat landscapes, and ensure the integrity of their Dockerized environments.

REFERENCES

- [1] S. Chopra and A. G. Sreedevi, "Scrutinization of Threats from a Cloud-based Network Traffic Environment," 2024 11th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2024, pp. 377-383, doi: 10.23919/INDIACom61295.2024.10498231.
- [2] Doan, Phuc & Jung, Souhwan. (2022). DAVS: Dockerfile Analysis for Container Image Vulnerability Scanning. *Computers, Materials & Continua*. 72. 1699-1711. 10.32604/cmc.2022.025096.
- [3] Lee, Haneul, Soonhong Kwon, and Jong-Hyouk Lee. 2023. "Experimental Analysis of Security Attacks for Docker Container Communications" *Electronics* 12, no. 4: 940.
- [4] Yadav, Mahendra Pratap, Nishant Pal and Dharmendra Kumar Yadav. "A formal approach for Docker container deployment." *Concurrency and Computation: Practice and Experience* 33 (2021): n. pag.
- [5] L. Zhang et al., "A Novel Hybrid Model for Docker Container Workload Prediction," in *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 2726-2743, Sept. 2023, doi: 10.1109/TNSM.2023.3248803.
- [6] D. S. Manikanta Narayana, S. Bharadwaj Nookala, S. Chopra and U. Shanmugam, "An Adaptive Threat Defence Mechanism Through Self Defending Network to Prevent Hijacking in WiFi Network," 2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS), Bangalore, India, 2023, pp. 133-138, doi: 10.1109/ICAECIS58353.2023.10170470.
- [7] J. Wenhao and L. Zheng, "Vulnerability Analysis and Security Research of Docker Container," 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 2020, pp. 354-357, doi: 10.1109/ICISCAE51034.2020.9236837.
- [8] K. M. Yatheendra Pravan, U. Shanmugam and P. V. Rajaraman, "INSIGHTS! - a modern deep learning approach to data analysis using Feature Name Substitution Network," 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), Chennai, India, 2019, pp. 1-4, doi: 10.1109/ICCIDS.2019.8862071.
- [9] D. B. Cruz, J. R. Almeida and J. L. Oliveira, "Open Source Solutions for Vulnerability Assessment: A Comparative Analysis," in *IEEE Access*, vol. 11, pp. 100234-100255, 2023, doi: 10.1109/ACCESS.2023.3315595.
- [10] Liu, W., Sreedevi, A.G. Special issue on evolutionary computation-based methods and applications for data processing. *Neural Comput & Applic* 35, 24959-24960 (2023).
- [11] Javed, Omar and Salman Zubair Toor. "Understanding the Quality of Container Security Vulnerability Detection Tools." *ArXiv abs/2101.03844* (2021): n. pag.
- [12] S. Kwon and J. -H. Lee, "DIVDS: Docker Image Vulnerability Diagnostic System," in *IEEE Access*, vol. 8, pp. 42666-42673, 2020, doi: 10.1109/ACCESS.2020.2976874.
- [13] Libo Chen, Yihang Xia, Zhenbang Ma, Ruijie Zhao, Yanhao Wang, Yue Liu, Wenqi Sun, and Zhi Xue. 2022. SEAF: A Scalable, Efficient, and Application-independent Framework for container security detection. *J. Inf. Secur. Appl.* 71, C (Dec 2022).
- [14] M. K. Abhishek and D. Rajeswara Rao, "Framework to Secure Docker Containers," 2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4), London, United Kingdom, 2021, pp. 152-156, doi: 10.1109/WorldS451998.2021.9514041.
- [15] Jain, V., Singh, B., Choudhary, N. (2022). Audit and Analysis of Docker Tools for Vulnerability Detection and Tasks Execution in Secure Environment. In: Balas, V.E., Sinha, G.R., Agarwal, B., Sharma, T.K., Dadheech, P., Mahrishi, M. (eds) *Emerging Technologies in Computer Engineering: Cognitive Computing and Intelligent IoT*. ICETCE 2022. *Communications in Computer and Information Science*, vol 1591. Springer, Cham.
- [16] J. Jacobs, S. Romanosky, I. Adjerid, and W. Baker, "Improving vulnerability remediation through better exploit prediction," *Journal of Cybersecurity*, vol. 6, no. 1, p. tyaa015, Jan. 2020.
- [17] J. Jacobs, S. Romanosky, B. Edwards, I. Adjerid, and M. Roytman, "Exploit Prediction Scoring System (EPSS)," *Digital Threats*, vol. 2, no. 3, p. 20:1-20:17, Jul. 2021.