Techno-Science Research Journal 12 (1) (2024) P 12-18

Content list available at ITC

Techno-Science Research Journal

Journal Homepage: http://techno-srj.itc.edu.kh/

Techno-Science Research Journal

Enhancing the Accuracy and Reliability of Docker Image Vulnerability Scanning Technology

Vannaroth Korn[1*], Kimheng Sok[1] Dona Valy[2]

[1] *Department of Information and Communication Engineering, Institute of Technology of Cambodia, Russian Federation Blvd., P.O. Box 86, Phnom Penh, Cambodia*

[2] *Research and Innovation Center, Institute of Technology of Cambodia, Russian Federation Blvd., P.O. Box 86, Phnom Penh, Cambodia*

Received: 17 July 2023; Accepted: 23 September 2023; Available online: June 2024

Abstract: *With the growth of usage of Docker containers in the recent year, Vulnerability scanning is essential to scanning and detecting known flaws and vulnerabilities in that specific Docker image. Using a custom docker image with third-party libraries in our code base authenticity or knowing their flaws can cause a lot of trouble in the future. In this case, vulnerability scanning tools such as Clair, Trivy, Anchor Grype, and Snyk are used for detecting the known vulnerability of the used Docker image and its included library but one Docker image can show different results depending on which tools we use due to the different scanning techniques and different vulnerability databases with different information (CVE, NVD, RedHat) that they use. In this research, We will be focusing on building an architectural framework that improves the accuracy, and reliability of the vulnerability scanning tools with a local vulnerability database that we built using a commonly used method such as static analysis which scans by reading package name and version searching, matching known suspicious pattern or signature using a binary database. Another method of scanning is a binary analysis which includes spotting unknown suspicious properties with a predefined algorithm. Using a confusion matrix, we evaluate the vulnerability scanning tool with docker images using the ground truth vulnerabilities stored in our local vulnerability database which is enriched with vulnerability information that is improved each time a new image is scanned.*

Keywords: Docker Image; Vulnerability scanning; Binary analysis; Static analysis; Common Vulnerability and Exposure (CVE).

1. INTRODUCTION

Nowadays containerization has become a common and widely used technology in today's Modern IT landscape, for the development, deployment, or management of applications. The popular containerization platform Docker enables developers to package applications and dependencies into one image. However, security concerns have also emerged as a result of the growing popularity of Docker and containerization.

Like any other applications, The Docker images that we use may contain vulnerabilities knowingly or unknowingly which can be exploited by malicious parties for the purpose of gaining unauthorized access or disrupting operations. During the process of creating an image, these vulnerabilities can be created or caused by an outdated component or unofficial third-parties library. In this case, scanning technologies such as Anchore Grype, Trivy, Snyk, and Clair are used in order to preprocess the used Docker image before deploying them. The issues that arise when using these technologies is that due to different way of implementation, uses of different scanning technique, different CVE, and vulnerability database the result response back can be very varied depending on these factors. So in order to reduce the vulnerabilities that are underlying in our application we need a more accurate way to determine whether an issue exists between or outside of these.

Docker images can put operating system isolation and security features of an application or server at risk and also cause threats and major security issues when it is found to be insecure as stated by Jain et al [7]. In order to achieve Container Image Authenticity they need to identify the source of the image, the

[*] Corresponding author: Korn Vannaroth
E-mail: kunvannaroth@gmail.com; Tel: +855-11 373 799

authenticity of the image, and the cryptographic proof of the author of the image. To answer that they conduct a study and evaluate different Vulnerability scanning tools such as Clair, Cilium, Anchor, OpenSCAP, Dagda, and Grafes and found the advantages and disadvantages of each tool. Since combining different tools with different scanning techniques and different advantages they concluded that "the shared information about security measures introduced by Docker Inc., gives the information of verified and certified images that can improve the security of Docker hub". Even though the security measure does not improve overall Docker Hub Security, It does improve the information about vulnerabilities relating to a specific Docker image.

Our contributions are 1) Building a Local Database that improves each time a new image is scanned. 2) Build a procedure to analyze and evaluate the result of each different tool; 3) Design and build an architecture in order to increase compatibility between different scanning techniques.

In this paper, there will be 6 main points: 1). Introduction, 2). Related works, Preliminary Research, 4). Methodology, 5). Result and Discussion, and finally 6). Conclusion.

2. RELATED WORKS

In order to produce more reliable results and improve accuracy, Jain et al [7]. concluded that static security analysis of Docker images is being performed for security enhancement using various tools. These tools highlight the vulnerabilities by running them with different software and platforms, aiming to make the containers more secure. Furthermore, the creation of secure container images should be approached prudently to mitigate the potential threats from becoming a major concern for containers.

Jagelid [8] noted the necessity for a static container vulnerability scanner, considering several perspectives. He used a confusion matrix to evaluate the difference between 2 vulnerability tools by calculating its accuracy, precision, and recall which is useful for limiting security fatigue, by ensuring false alerts to developers are low.

Ohaeche [9] determines that creating a script detecting vulnerabilities and halt the deployment of the image into production. The difference between file systems does affect how dependencies are extracted, however, similar approaches can be used for container images and VM images stored in the MIF format. An advantage of container images lies in their fast adaption of standard formats. Therefore, one can focus on managing tar archives instead of supporting various different techniques, unique to every platform.

Berkovich et al [4] describe that the confusion metric is an obvious scanner evaluation metric, marking success as relevant vulnerabilities detected by the scanner under evaluation as true positives (TP), with not-relevant vulnerabilities being false positives (FP). Relevant vulnerabilities not detected by the scanner are false negatives (FN). Precision is defined as the fraction of retrieved vulnerabilities that are in fact relevant, expressed as Precision = TP / (TP + FP). Accuracy measures the proportion of correctly classified cases from the total number of vulnerabilities calculated as Accuracy = (TN + TP) / (TN + TP + FN + FP). Recall is the fraction of relevant vulnerabilities detected by the scanner, represented as Recall = TP / (TP + FN). The F-measure characterizes the combined performance of recall and precision, calculated as F-measure = (2 * Recall * Precision) / (Recall + Precision). They utilize these three metrics to assess scanner quality.

Brady et al [5] proposed an automated architectural process to scan and analyze images for vulnerabilities by combining Clair and Anchor Engine into a CI/CD pipeline. This integration aims to reduce the burden on developers by automating security analyses. They also implemented an API that scans for malware on public images, resulting in virus scans and dynamic analysis yielding results in detecting malicious behavior in Docker containers..

3. PRELIMINARY RESEARCH

3.1. Docker Image

Docker image is a lightweight, self-contained, and executable package that contains everything needed to run an application, including the code, runtime environment, system tools, libraries, and dependencies. Docker images are based on a layered file system, which enables efficient storage and sharing of common components across multiple images. Each layer represents a specific modification or addition to the previous layer that provides, resulting in a highly efficient and space-saving image distribution.

Docker images are created using a Dockerfile, which is a text file that specifies the instructions to build the image. These instructions include defining the base image, adding files, installing packages, configuring the environment, and executing commands to set up the application within the container. In other words, Custom Docker images are made up of 4 layers: layer-1 base image, layer-2 working directory, layer-3 dependencies,

and layer-4 accessibility exposure.

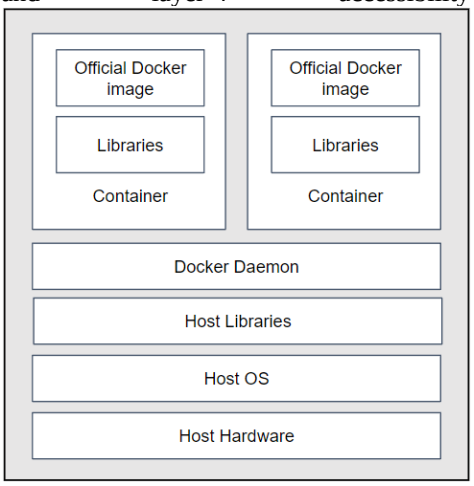

Fig. 1. Docker Container Layer Architecture

3.2. *Anchore Grype*

Anchore Grype is a container vulnerability scanning and management tool. It is designed to assist developers and organizations in identifying security issues within container images that are widely used in modern software development and deployment. It works by analyzing the contents of container images and comparing them against a database of known vulnerabilities. It detects any recognized vulnerabilities or weaknesses in the image, including operating system packages, libraries, or application dependencies. By providing this information, it empowers developers to address risks and ensure the integrity of their containerized applications.

3.3. *Trivy*

Trivy is an open-source vulnerability scanner for containers and other software artifacts such as Docker images, container runtimes, and package managers. It helps developers and security teams identify vulnerabilities in these artifacts, allowing them to proactively address security risks and ensure the integrity of containerized applications. Trivy uses a comprehensive vulnerability database that includes information from multiple sources much similar to Anchore Grype.

Trivy server has vulnerability database that are store in it github repository and Trivy client doesn't have to download vulnerability database. Trivy take a container registries or a .tar file. When analyzing a .tar file, it will then send a request for extracting missing layers which are third-party libraries that exist in this custom Docker image as shown in Fig.2. It is useful if we need to scan images at multiple locations and do not want to download the database at every location. When launching Trivy server for the first time, it downloads vulnerability database automatically fetch data.

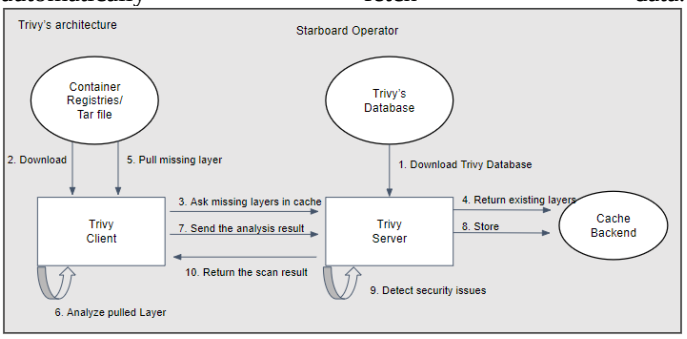

Fig. 2. Trivy's Architecture

3.4. *Common Vulnerability and Exposure(CVE)*

Common Vulnerabilities and Exposures (CVE) is a standardized system for identifying and naming security vulnerabilities in software and hardware products. It provides a unique identifier for each vulnerability, allowing organizations and security researchers to easily reference and track them. Vulnerability refers to a weakness or flaw in a system that can be exploited by attackers to compromise the security, integrity, or availability of the system. Common vulnerabilities can include software bugs, configuration errors, design flaws, or inadequate security measures. CVE aims to provide a universal language for discussing and sharing information about vulnerabilities across different organizations, platforms, and security tools. Vulnerability in the CVE system is assigned a unique CVE identifier, which consists of the prefix "CVE-" followed by a year and a unique number (e.g., CVE-2021-1234).

4. METHODOLOGY

These technology there will be 2 main method to scan for vulnerability which are: Static analysis and Binary analysis.

4.1. *Static analysis*

This research is conducted based on an investigation of Docker image scanning technology to improve the accuracy of the vulnerability scanning tool and multiple vulnerability databases. The first methods that are used in Docker image vulnerability scanning tools involve static analysis.

Static analysis in the context of Docker image vulnerability scanning entails analyzing the contents of a Docker image without actually running the image. This analysis is typically performed on the layers of the Docker image, which are the

individual components that make up the image.

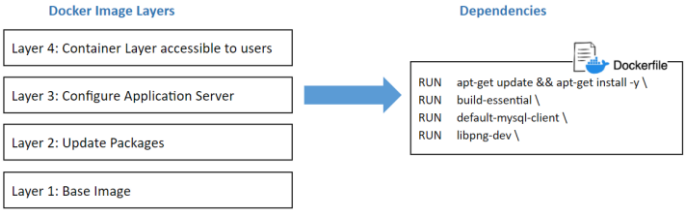

Fig. 3. Docker Container Layer Architecture

Package manager scanning is commonly used in CVE Docker image scanning tools as a form of static analysis to identify known vulnerabilities in software components installed within Docker images. These tools typically leverage package manager metadata, configuration files, and information on installed packages to match against a known vulnerability database or known vulnerable package version.

Signature-based detection: Similar to pattern matching, signature-based detection involves comparing the contents of the Docker image layers against a database of known signatures associated with vulnerabilities. Signatures are unique identifiers or characteristics that are associated with specific vulnerabilities, and static analysis tools can use these signatures to identify potential vulnerabilities in Docker images.

4.2. Binary analysis

Binary analysis is the second method its databases in Docker image vulnerability typically consist of a collection of known binary patterns, signatures, or fingerprints that are associated with known and existing vulnerabilities. These databases are used by vulnerability scanning tools to scan Docker images for potential vulnerabilities by matching the binary patterns or signatures in the images against those in the database.

Dependency analysis: This method involves examining the dependencies of binary artifacts, such as libraries or other software components, to identify any known vulnerabilities associated with those dependencies. This involve checking against known vulnerability databases, analyzing version numbers, and comparing against lists of vulnerable dependencies as shown in Fig.3. Dependency analysis helps identify vulnerabilities that may arise from using suspicious code pattern or vulnerable third-party libraries or software components.

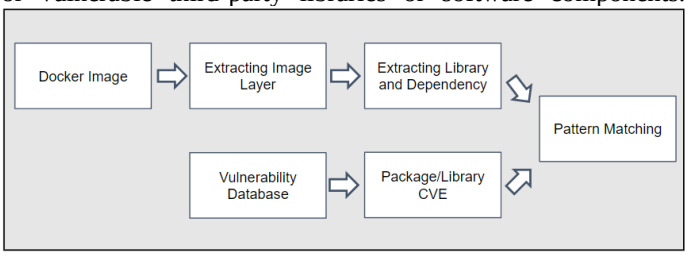

Fig. 3. Extracting Dependency Workflow

4.3. Proposed Architecture

In this architecture, User first needs to input a docker image through a UI, these will go through different flows depending on the Input of the user to scan for vulnerabilities then the input image will go through the 5 main modules.

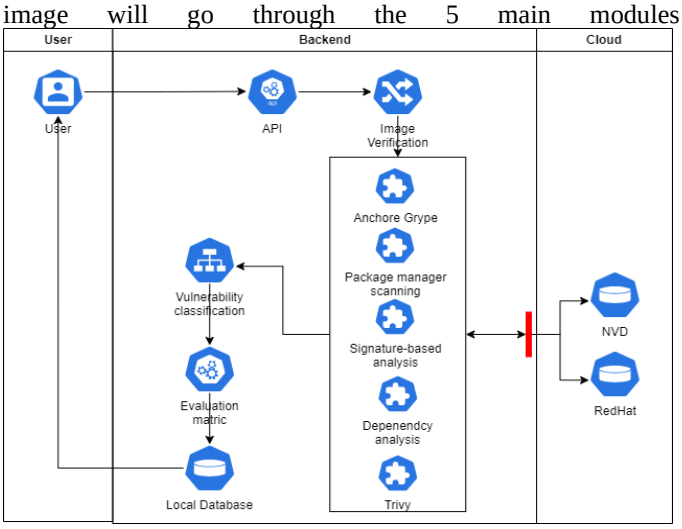

Fig. 4. Proposed Architecture

4.3.1. Image Verification

First of all, when inputting a docker image to scan for their vulnerabilities but in the case of Image verification user can either load an official base image from the docker hub or a custom docker image which they build using a combination of official docker images and other libraries with Dockerfile. Not only that but users can choose not to access the cloud and return the existing vulnerability in the local Database.

In the case of a user loading a custom image, we need to use a Dependency analysis to detect the third-party library they use and package manager scanning and signature base analysis for the official docker image they use in this custom image. While as if they were to load an official docker image as an Input we do not need to run a Dependency analysis.

4.3.2. Scanning Processes

After obtaining the image name and version or its version if the loaded image was a custom docker image, We will get the vulnerabilities with our API from NVD, Redhat, Anchore Grype, and Trivy. The vulnerability information that we got from Nvd and Redhat is a lot and we do not require all of it, So we need to unorganized and clean the API for our usability.

In Anchor Grype and Trivy processes, We can access its vulnerabilities database by installing them into our machine but xsince both Anchore Grype and Trivy can only be run through a command prompt we create an API that accesses the command

prompt to request the vulnerability information and return them according to our usage. We also need to adjust and customize both the parameters and the results, Since we can you Grype to scan for both custom Docker images and official Docker images.

Package manager scanning process is one of the analysis method that we implement into our architecture, When it comes to custom images, The Custom Docker image is composed of an official image and included with the necessary dependencies to run that application. Usually, these custom images are composed into a .tar file and so first we extract the Third Layer from the image which contains all of the dependencies. After having the dependency's name. We can further it by extracting the vendor, cache, and configuration from these depending. When we finally have all of this information. We stored them in a temporary thread awaiting information from Cloud Vulnerability Database to do Pattern Matching. But when the input from the user is an official Docker image then this process will not run, Since there are no dependencies installed in the official one.

Signature-based analysis process is another analysis method that we implement and it is by far the simplest out of the 3. The difference between static analysis of Package manager scanning and Signature-based detection is that the Package work on the Third-Layer of the Docker image while Signature-based work on the First-Layer which is the Base Image layer. When scanning for vulnerabilities using this method, It read the name of the used Docker image and search for existing vulnerability relating to this image in the Cloud Database (NVD, CVE) if the input is parsed with the image version then it will also search for that specific version otherwise it will by default search for vulnerabilities of the latest version.

Lastly Dependencies analysis process , It works similarly to the Package manager scanning process. The main difference is that dependencies analysis installs these libraries and scans for a collection of binary patterns and signatures associated with known vulnerabilities. We know that not all custom libraries with existing vulnerabilities are stored in the database but these libraries may use existing attack patterns, So in such cases, we can prevent these malicious libraries by searching for an existing vulnerability pattern that are used for attacking in the past.

The API call different type of process depending on the input given by the user. This process is unique from one another because it utilizes different analysis methods. In this case, we can scale our architecture by adding more processes that use other methods in the future for more accuracy.

4.3.3. Cloud Database

All the vulnerability information utilized in the analysis methods above originates from a Cloud database. While NVD and Red Hat have their own APIs for inquiring about vulnerability information, including CVE, CPE, and vendor details.

Anchore Grype and Trivy, on the other hand, lack APIs. These tools access the vulnerabilities database at intervals whenever we request information through a command prompt. However, they also rely on the vulnerability database to retrieve information.

Although using the same vulnerability database might lead to redundant data, that isn't always the case. One potential reason for redundancy could arise from differences in how vulnerability scanning tools prioritize the severity of vulnerabilities. Another factor influencing differing results could be the frequency of updates to the vulnerability database.

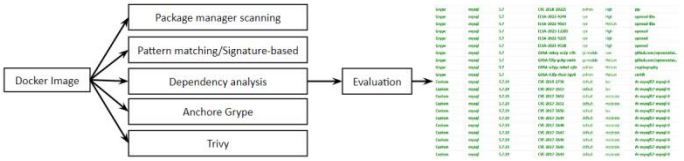

Fig. 5. Building Ground Truth WorkFlow

4.3.4. Evaluation

In this phase, we will classify all of the vulnerabilities found by each into True Positive, True Negative, False Positive, and False Negative, including the vulnerabilities that are found using the modified analysis method above, and evaluate our work using a Confusion matrix. As we conclude the our Local Database is the Ground truth, Thus Vulnerabilities found that exist in our Ground truth are marked as True Positive(TP), Vulnerabilities that are found but does not exist in our Ground truth are False Negative(FN), exist in ground truth and not found are classify as False Positive(FP) and finally existing in ground truth does not exist and are not found as True Negative(TN).

4.3.5. Local Database

Finally, After processing all of the vulnerabilities using our API that accesses the Cloud database and command prompt, we store them in our local database as a ground truth of the combination of all of the information from different databases and tools for evaluation purposes using different API and also reducing time complexity when the next request contains a third-parties library or base images that already exist in our local database.

5. RESULTS AND DISCUSSION

In the evaluation phase, We have to collect a docker image with existing vulnerabilities to create a ground truth and then we can analyze the quality of the vulnerability scanning tool with the result returned by each tool using a confusion matrix to classify the vulnerability found and finding their precision, recall, and f-measure.

As shown in Fig.6, We have tested 10 Docker images some of which are custom Docker images such as Php, Node, and OpenJDK. This means they contain a third-party library as you can see the spike difference between these 3 Docker images

compared to the other 7. In the below figure, you can clearly tell the difference in results between technology that is because in the previous discussion, we already established that some technology specializes in some Docker image, and also the same vulnerability can exist on multiple vendors.

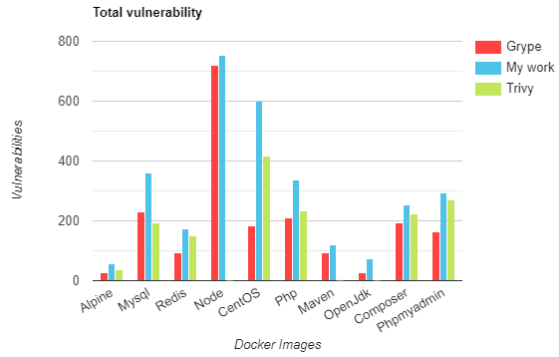

Fig. 6. Total Vulnerabilies of Docker Image with each Scanning Tools

Table 1. Evaluation of found vulnerability in Docker Image

		Alpine 3.13.5	Mysql 5.7.22	Redis 6.2	Node 16	CentOS 8.4	Php 8.1	Maven 3.8	OpenJDK 17	Composer 2.1.11	Phpmy admin 5.21
Grype	True Positive	27	229	92	719	184	211	93	26	194	163
	True Negative	0	0	0	0	0	0	0	0	0	0
	False Positive	8	189	23	754	176	103	121	72	3	23
	False Negative	18	132	82	35	191	125	28	46	31	130
Trivy	True Positive	37	192	151	0	416	233	0	0	222	270
	True Negative	0	0	0	0	0	0	0	0	0	0
	False Positive	18	132	82	35	191	125	28	46	31	130
	False Negative	8	189	23	754	176	103	121	72	3	23

In this last phase, In order to accurately identify the quality of the tools were are using we need to classify the found vulnerabilities to TP, TN, FN and FP. Since in our case, there is no prediction value and our primary uses it for classifying vulnerability to calculate the precision and performance of a tool and compared it to another tools. We can calculate the recall and F-measure using confusion matrix in order to compare it against our work. In this case, Grype and Trivy are calculated to compare against our work. The goal of this evaluation is that when adding a new technology , new analysis methods and its scanning technique into our architecture, we can use this evaluation method to compare them since the result of the newly added technology is still unknown in our local database.

As mention in [4] and [8], we use confusion matrix to evaluate the performance of the vulnerability scanning tool and compare it result with the relevance and irrelevance vulnerability between the detected and the actual.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (Eq.1)$$

$$Precision = \frac{TP}{TP+FN} \qquad (Eq.2)$$

$$Recall = \frac{TP}{TP+FN} \qquad (Eq.3)$$

$$Fmeasure = \frac{(2*Recall*Precision)}{(Recall+Precision)} \qquad (Eq.4)$$

In Fig.6 after we got the information on existing vulnerabilities on docker image, we classify these vulnerabilities into TP, TN, FN, FP using our evaluation process and the result is as shown above Table 1. The result in Table 2 and Table 3 are calculated based on (Eq.1), (Eq.2), (Eq.3) and (Eq.4) using the data listed in Table 1.

Table 2. Confusion Matrix of Grype's Result

	Precision	Accuracy	Recall	F-measure
Mysql	0.7714	0.5094	0.6	0.675
Php	0.6719	0.4806	0.6279	0.6492
Redis	0.8	0.4670	0.5287	0.6366

Table 3. Confusion Matrix of Trivy's Result

	Precision	Accuracy	Recall	F-measure
Mysql	0.5925	0.3742	0.5039	0.5446
Php	0.6508	0.5054	0.6934	0.6714
Redis	0.6480	0.5898	0.8678	0.7420

Table 1 shows that there is a very vast different be between the result of Php image scanned by Grype and Trivy, this is cause by of the third-party library that are installed within them. Vulnerabilities in alpine image is fewer compared to other due to it being an base image and not containing any dependencies installation. Table 2 and Table 3 shows that the accuracy

between different image using different tools can differ resulting in inaccurate result or missing vulnerability.

As the result shows above, We can see that adding more analysis methods to our architecture can increase the number of vulnerabilities found in a docker image. Even using a redundant vulnerabilities database we can still be able to catch a few more vulnerabilities even if they are not notable or high severity and Thus allow users to be able to decide which docker image or version they want to use.

6. CONCLUSIONS

In conclusion, we can improve the vulnerability scanning tools greatly by using more sources of vulnerability databases including the database used by other tools to confirm if the vulnerability exists, a bigger database, implementing more analysis methods, and utilizing this bigger database to improve the result, designing an architecture that supports all of these tools together, writing an algorithm to optimize the process of accessing multiple vulnerability databases to compensate for a bigger source, Building a Local Database that are enriched with vulnerabilities and is improved constantly every time a new Docker image is scanned, Implementing a Confusion matrix to evaluate and compare the result to our when a new technology is added in the architecture, Designing and Building an architecture to add different scanning technique between different tools in order to improved result.

ACKNOWLEDGMENTS

This research was conducted in the Information Security and Application Design lab of the Department of Information and Communication Engineering at the Institute of Technology of Cambodia.

REFERENCES

[1] Abhishek, M. K., & Rao, D. R. (2021, July). Framework to secure docker containers. In 2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4) (pp. 152-156). IEEE.

[2] Al-Asli, M., & Ghaleb, T. A. (2019, April). Review of signature-based techniques in antivirus products. In 2019 International Conference on Computer and Information Sciences (ICCIS) (pp. 1-6). IEEE.

[3] Alyas, T., Ali, S., Khan, H. U., Samad, A., Alissa, K., & Saleem, M. A. (2022). Container Performance and Vulnerability Management for Container Security Using Docker Engine. Security and Communication Networks, 2022.

[4] Berkovich, S., Kam, J., & Wurster, G. (2020, August). UBCIS: Ultimate benchmark for container image scanning. In Proceedings of the 13th USENIX Conference on Cyber Security Experimentation and Test (pp. 10-10).

[5] Brady, K., Moon, S., Nguyen, T., & Coffman, J. (2020, January). Docker container security in cloud computing. In 2020 10th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0975-0980). IEEE.

[6] Cox, J., Bouwers, E., Van Eekelen, M., & Visser, J. (2015, May). Measuring dependency freshness in software systems. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (Vol. 2, pp. 109-118). IEEE.

[7] Jain, V., Singh, B., Khenwar, M., & Sharma, M. (2021, April). Static vulnerability analysis of docker images. In IOP Conference Series: Materials Science and Engineering (Vol. 1131, No. 1, p. 012018). IOP Publishing.

[8] Jagelid, M. (2020). Container Vulnerability Scanners: An Analysis.

[9] Kwon, S., Harrison, K., Kweon, S. J., & Williams, D. S. B. (2020). Divds: Docker image vulnerability diagnostic system. IEEE Access, 8, 42666-42673.

[9] Ohaeche, J. U. (2022). Docker Container Image-Vulnerability Scanning

[10] T. Combe, A. Martin, and R. Di Pietro, "To Docker or Not to Docker: A Security Perspective," in IEEE Cloud Computing, vol. 3, no. 5, pp. 54-62, Sept.-Oct. 2016, doi: 10.1109/MCC.2016.100.

[11] Wenhao, J., & Zheng, L. (2020, September). Vulnerability analysis and security research of docker container. In 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE) (pp. 354-357). IEEE.

[12] Xu, J., Wu, Y., Lu, Z., & Wang, T. (2019, July). Dockerfile tf smell detection based on dynamic and static analysis methods. In 2019 ieee 43rd annual computer software and applications conference (compsac) (Vol. 1, pp. 185-190).