International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 7, Issue 11, pp 315-320, November-2023
https://doi.org/10.47001/IRJIET/2023.711043

Prevent Registry Being Compromised by Pulling Insecure Container Images from Public Registries

[1]V.U. Kumarasiri, [2]Ms. Chethana Liyanapathirana, [3]Dr. Lakmal Rupasinghe

[1]Department of Computer Systems Engineering, Sri Lanka Institute of Information Technology, Sri Lanka
[2,3]Lecturer, Department of Computer Systems Engineering, Sri Lanka Institute of Information Technology, Sri Lanka

***Abstract* - Container technology is one of the fastest growing technologies. However, when it comes to the security of containers, it appears that their vulnerability has increased in line with its popularity. Because when users pulled images from various repositories for various need there can be images that can be harmful images to entire container environment. So, we need to verify security of those images before store them into container environments. In this paper, I describe a mechanism that can validate security level of container image before it pulled into the container. In addition, we use hash validation as well as Format String Validation for further check the image validity.**

Keywords: Docker, Container Vulnerability, Hash Validation, Format Matching.

I. INTRODUCTION

One of the most widely used technologies in the software development industry is containers since they free up developers' time to focus on their primary tasks. One of the most important tasks in the software development process is setting up dependencies for the deployment of the application. Images are files in the container environment that include the necessary code, configuration, and libraries to run an application. The instances of these images that we can define as containers all have the same underlying dependencies. These container images contain only the instructions and code required to run the application, so containers are lightweight compared to alternative approaches.

Docker is one of the most popular tools that we can use to create our own a container environment. It provides a platform as a service for developers to deploy their development without wasting any more time. With the rising popularity of Docker containers many popular organizations in the world such as google incorporating Docker Containers into their software development process. For this reason, the Docker development community created a repository containing all legitimate container images, called Docker Hub.

Software development in Docker increasingly relies on small, reusable components developed and distributed independently by different organizations. Because of that it raises concerns regarding the security of the entire Docker image distribution process. That's because some Docker images may have major security issues that affect the organization's entire Docker infrastructure. By introducing a mechanism at the Docker registry level to pull only secure images from public registries, pulling insecure container images from public registries can prevent the registry from being compromised.

The requirement of this kind of an algorithmic approach is to minimize potential malicious activity in the container environment due to unknowingly pulling insecure images. In this paper, the proposed algorithm approach is based on the layered protection approach. It is expected to focus mainly on the security of private registries. This approach will use a multi-layered algorithmic approach to verify the security level of container images.

II. BACKGROUND

Containerization has revolutionized the way software is developed, deployed, and managed, enabling rapid and scalable application deployment. Container images are a fundamental part of this process and can be described as a setup package containing an application and all its dependencies.

However, although everything can be done very easily in this container environments, it also has a dark side of this which have some security issues. Because of that reason we can see there are some efforts that have been done across the Docker community to encourage security analysis by users. But it was not very successful. So the ideal solution would be develop a tool that can interact with development cycle of Docker images directly and do the security analysis.

Due to concerns about vulnerabilities in Docker images, there are several open source tools and paid solutions that can respond to such scenarios. However, there are some security issues that have not yet been resolved by these products, and the implemented mechanism has not fully met the security needs of container users. That is the reason why we conducted

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 315-320, November-2023

https://doi.org/10.47001/IRJIET/2023.711043

this research to find the optimal algorithmic and integrated solution to meet those unsolved problems.

III. DESIGN

To address these not fully resolved security issues, we propose an extension that uses the algorithmic approach and that extension have ability to identify vulnerable docker images in pulling process and isolate those images.

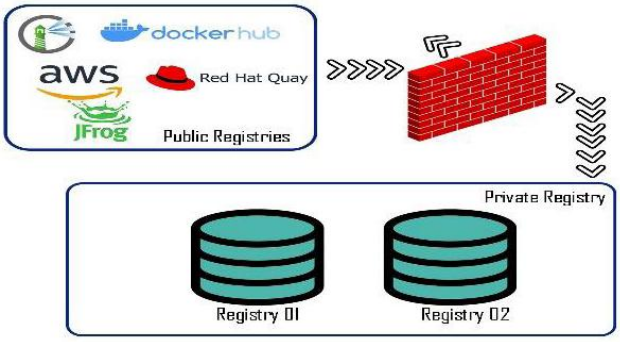

Figure 1: Overall Process

This process uses an extension as a middleware that acts as a wall between the private registry and the public registry without having direct contact with the public registry. The reason we implement this kind of process is that in the current pull request process in Docker, the user has no control over their pull request after the request is sent to public regitries. Because of that, there is a high probability that the malicious or tamered image will be pulled into the private registry via that request. So we are going to put an extension as a middleware and control the risk. There are various mechanisms in that middleware that are able to mitigate such risks.

A) Indirect Image Pulling

The first and most important function in our proposed extension is the Indirect Image Pulling process. This function enables the process of pulling images from public registries without direct interaction.

Figure 1 clearly illustrate the process of indirect image pulling. According to the diagram, after the user initiates the process of pull docker image, the request will first be directed to the extension we installed. After the extension captures the user request it analyzes the entire request using their algorithm. When the analysis part is complete it will check which secured destination to forward the user's request to. After redirecting the pull request to that source, it pulls the corresponding image from that source.

B) Format Matching

After the request is returned to the extension by the public registry as a response, there are several functions that are executed inside this extension. One of those functions is format matching. This format matching function is used to verify the image returned by the public register.

C) Hash Verification

Cryptographic hash verification plays a critical role in strengthening Docker security by ensuring the integrity and authenticity of Docker images. Docker images are identified by content-based digests, which are cryptographic hashes that uniquely represent their content. When users pull Docker images from public repositories, they can verify the integrity of the image by comparing its digest with the desired hash value. This protects against possible tampering or man-in-the-middle attacks during image distribution. By using cryptographic hashes and image signing, Docker provides a secure way to distribute and deploy containers, improving overall security and enabling users to make informed decisions about the trustworthiness of the images they use. Security scanning services offered by some registrars further contribute to Docker's overall security by detecting known vulnerabilities and ensuring secure deployments.

D) Source Verification

In an era characterized by the widespread use of containerization technologies such as Docker, ensuring the security and integrity of containerized applications has become paramount. In this context, a new approach arises: active verification of image sources before they are pulled into a container environment. This approach involves implementing an algorithm within Docker extensions, aimed at mitigating security risks associated with compromised or malicious images. By incorporating a source verification mechanism, the algorithm attempts to verify the legitimacy and reliability of image sources before deploying them. This includes evaluating the reputation of the source, assessing digital signatures and cross-referencing against known repositories of approved images. Through this method, the proposed solution not only strengthens the security posture of container ecosystems, but also contributes to preventing potential supply chain attacks. This research paper presents the conceptual framework, technical implementation, and potential benefits of integrating such an algorithm into Docker extensions, leading to improved security and reliability in containerized environments.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 315-320, November-2023

https://doi.org/10.47001/IRJIET/2023.711043

E) Image Signing Verification

Image signature verification for Docker images offers many powerful advantages that underline its key role in strengthening the security landscape of containerized applications. One of the most notable advantages is the ability to establish a resilient foundation of trust within the container ecosystem. By meticulously verifying the digital signature of images before they are pulled into the environment, this approach enables rigorous verification of their origin, authenticity, and integrity. This key step significantly mitigates the risks associated with image source compromise or substitution, effectively curbing the potential deployment of containers obtained from untrusted or tampered sources.

Public registries may contain many malicious images. Once that image is pulled into our private registries, it may create backdoors for other third parties to access our private registries and entire environment. To solve this problem, we develop a mechanism to check the reliability of the container image before pulling it into the private registries.

IV. IMPLEMENTATION

In this section, we describe the implementation of our design using docker and extension sdk.

A) Indirect Image Pulling

This process will be done in two stages. In the first part by using our developed extension it captures the users request and analyzes it and finds out what images it is requesting and which way it should proceed. In the second half of the process, after receiving the image as a response to the request from any public registry, it checks its reliability with the functions I mentioned in the design section.

By using this technology, it isolates public and private registries from each other and does not establish any direct relationship between them.

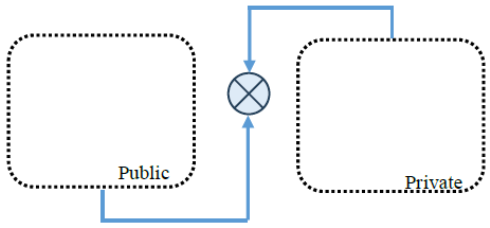

Figure: 2. Indirect Image Pulling

Figure 2 clearly shows how each registry is isolated from the other using the extension as a middleware. That is the place which is responsible all the workflows of our implementation. In this point, many processes take place to minimize the pulling of insecure Docker images. There we can categorize the processes as image pulling process, image scanning and image isolation.

As the first step in this part of our implementation, the middleware captures the requests from the private registry. After capturing the request, it analyzes the content of that request using the functionality implemented in the middleware.

If the captured request is completely legitimate, it is automatically redirected to the public registry to retrieve the corresponding Docker image. Otherwise, the request will be blocked the first time it is analyzed.

If the request passes the first layer of our implementation, it goes to the relevant public registry, such as Docker Hub, to retrieve the requested image. After getting the relevant image it will go through the extension again.

If the public registry responds to the extension, it means that the public registry returned the relevant image to our request. If that is the case the next layer of security is going to execute. In that process extension going to evaluate the security level of returned image by performing next function of our extension which is source verification.

B) Format Matching

In this process of execution, the integrity of the received image is checked. That is, there is a unique hash value for each image and its integrity is checked by comparing the hash value of the received image with the actual hash value.

If the integrity of a received image is not correctly verified, our process will stop all interaction with that image and isolate that image.

C) Hash Verification

Hash verification is another sub-process of the previous section. As mentioned before, what happens here is to check the accuracy of the image, the corresponding hash value is compared with the database.

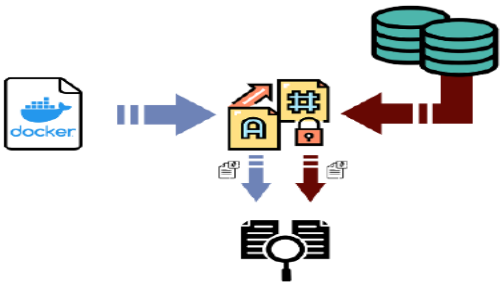

Figure 3: Hash Verification

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 315-320, November-2023

https://doi.org/10.47001/IRJIET/2023.711043

D) Source Verification

Source verification is another key part of our implementation. It is new for all container security implementations. This is the process of verifying the source of the received image by checking that the image actually came from the legitimate source that stored it.

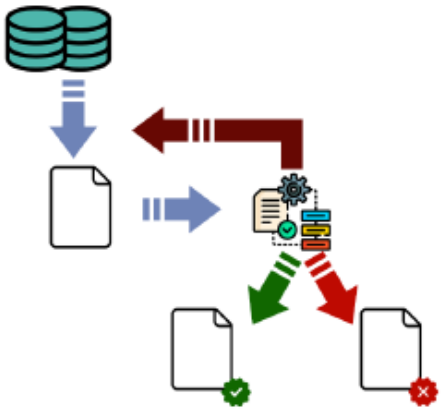

Figure 4: Source Verification

The reason why we decided to develop this kind of mechanism is that when the user requests any container image by running the corresponding command.

When the user requests a container image with a command, the first step is to execute the command that the user should use to pull the image into the current environment. For example, the docker pull command used in Docker can be pointed out. After executing the command, the user has no control over the request and they will not be able to find out from which registry the image they receive as a response.

Because of this, some images can sometimes be pulled by malicious registries. Therefore, I think that the problem can be avoided by checking the source of the image before pulling it into our environments.

V. RELATED WORKS

In the area of ensuring the security of container environments the landscape of vulnerability scanning tools and techniques has been continuously evolving. One such notable tool is "Docker Scout," which comes equipped with built-in mechanisms for vulnerability scanning. This tool performs a comprehensive assessment of Docker images to identify existing vulnerabilities. However, the tool's inability to proactively perform vulnerability scans is a notable weakness here. This means that vulnerability scanning can be done after the image is installed in the environment. This flaw could expose the system to security vulnerabilities by allowing images to be stored in the environment prior to assessment.

In the face of these challenges, researchers have explored alternative methods to strengthen container image protection. Thien-Phuc Doan and Souhwan Jung [3] present an innovative approach through their Dockerfile analysis technique. This approach involves rigorous static analysis of container images to reveal vulnerabilities. A key component of their method is the use of Probable Vulnerability Files (PVFs) in the vulnerability scanning process. This technology improves the accuracy of vulnerability detection by identifying files that are commonly associated with security issues, thereby providing a more granular assessment.

In a study by Li et al. [4], the authors propose a comprehensive risk assessment framework covering the entire container supply chain. Their work emphasizes the importance of evaluating interactions between different images in a container environment to identify potential security threats. By investigating dependencies and interactions between container images, this approach provides a detailed view of possible attack surfaces, reducing the likelihood of overlooking critical vulnerabilities.

VI. CONCLUSION

In the dynamic landscape of modern software development, containers have emerged as a key tool for developers, streamlining the process and increasing the portability of applications. However, with the widespread use of containers have come many of the inherent bugs and vulnerabilities.

Throughout this article, we have explored the potential for compromised container environment integrity due to insecure container images. This final section reflects the insights gained through our research and emphasizes the imperative of increasing container security to mitigate these risks.

Our study revealed that the use of insecure images is a major contributor to vulnerability in container environments. When developers pull images from public repositories, these images have the potential to contain hidden vulnerabilities, exposing the host system to a range of security vulnerabilities. As a result, it has become clear that the container ecosystem needs additional layers of protection to prevent introducing these unexpected risks.

Our research findings clearly emphasize the need for a multifaceted algorithmic approach to container security. We advocate for the inclusion of supplemental security features in container environments to counter threats that may emerge

ISSN (online): 2581-3048
Volume 7, Issue 11, pp 315-320, November-2023
https://doi.org/10.47001/IRJIET/2023.711043

from insecure images. Through detailed analysis and discussion, this paper has demonstrated specific security implementations suitable for immediate integration into the container security landscape. These implementations include techniques such as image hash verification and source verification, which act as strong defenses against the inadvertent introduction of compromised images.

In summary, this research confirms that container security is a complex undertaking that mandates continuous vigilance and proactive measures. By raising awareness of vulnerabilities arising from insecure images, our study underscores the importance of developers and administrators collaborating to enforce strong security practices. These practices include not only adopting advanced safety measures but also a holistic approach that spans the entire container life cycle.

Finally, this research underscores the critical role of security in the field of container technology. The importance of hash verification, source verification, and other security implementations underscores the value of proactive security measures. It is our hope that this article catalyzes a broader conversation about strengthening container security, ensuring that security remains a constant companion as the software development landscape evolves.

In summary, the path forward requires a collective commitment to improving container security and strengthening defenses against potential threats, ultimately fostering a safer and more resilient digital ecosystem.

A) Recommendations

In the current state of Docker containerization, there is a critical concern regarding the security of image pulling processes. Currently, the Docker environment lacks an indirect image pull mechanism, leaving users unwittingly exposed to potential risks associated with unwittingly pulling malicious images. To mitigate this risk, it is essential to establish a mechanism to enable images to be retrieved from public registries without requiring direct user interaction.

The proposed solution revolves around the implementation of an indirect image extraction process that promises multiple advantages. Primarily, this mechanism acts as a protective shield against inadvertent acquisition of compromised images. By intervening an additional layer between the user and the public image repository, the likelihood of users pulling malicious content is significantly reduced.

To reinforce the security posture of the Docker environment, the expected process advocates routing images across a series of processors. This multifaceted approach is designed to comprehensively test and validate the integrity of images before they are integrated into the container environment. Each processor is assigned different responsibilities such as image signature verification, hash verification and source verification. This orchestrated sequence of checks acts as a resilient gatekeeper, ensuring that only safe and legitimate images are included.

In summary, the proposed strategy of implementing an indirect imaging process holds the promise of raising container security to unprecedented levels. As the technology landscape continues to evolve, embracing these security enhancements will surely prove to be a prudent and proactive move.

REFERENCES

[1] K. Brady, S. Moon, T. Nguyen, and J. Coffman, "Docker Container Security in Cloud Computing," Engineering for Professionals, Whiting School of Engineering, Johns Hopkins University, Department of Computer and Cyber Sciences, United States Air Force Academy.

[2] Waheeda Syed Shameem Ahamed,Pavol Zavarsky ,Bobby Swar, "Security Audit of Docker Container Images in Cloud Architecture".2021.

[3] Thien-Phuc Doan , Souhwan Jung, "DAVS: Dockerfile Analysis for Container Image Vulnerability Scanning," 2022.

[4] Li, Yujie, et al. "Towards Holistic Vulnerability Assessment of Container Image Supply Chain." Proceedings of the ACM Workshop on Cyber-Physical Systems Security and Privacy. 2019.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 315-320, November-2023

https://doi.org/10.47001/IRJIET/2023.711043

Citation of this Article:

V.U. Kumarasiri, Ms. Chethana Liyanapathirana, Dr. Lakmal Rupasinghe, "Prevent Registry Being Compromised by Pulling Insecure Container Images from Public Registries" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 7, Issue 11, pp 315-320, November 2023. Article DOI https://doi.org/10.47001/IRJIET/2023.711043

****

