JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

CONTAINER BASED BROWSER USING DOCKER

Nishad Utpat#1, Utsav Rohilla*2, Kirti Bhaddarpure#3 , Mahesh Pimparkar#4 , Prof. Nikhil Dhavase#5

Information Technology, MMCOE, Savitribai Phule Pune University, Karve Nagar, Pune, India

Asst. Professor, Information Technology, MMCOE, Savitribai Phule Pune University, Karve Nagar, Pune, India

Abstract— A containerized browser based on Docker is presented in this study to improve security, privacy, portability, and development and testing efficiency. Each surfing session is isolated by the system, which improves security and safeguards user privacy. Developers can test and deploy web applications with great ease thanks to the containerized browser and the portability of Docker, which guarantees consistent configurations across several environments. Because of Docker containers' versatility, web browsing can be made safer, more private, and easier to use by enabling effective version control and browser setting modification.

Keywords— Docker , Containerization .

I. Introduction

There are several benefits to web browsing when a container-based browser is implemented with Docker, especially when it comes to security, privacy, portability, testing and development, and flexibility. The system can efficiently quarantine potential threats and malware by isolating each browser session behind its own Docker container. This ensures that even in the event that a session is hacked, the host system remains secure. Since maintaining one's privacy is of utmost importance, containerization makes sure that every browsing session runs separately, avoiding data leaks between sessions and protecting private user data. With the knowledge that their online activities are isolated and kept secret, users may browse with confidence.
To summarize, the utilization of Docker in a container-based browser improves security, protects user privacy, guarantees portability, eases testing and development, and offers flexibility in customizing and administering multiple browser versions. All of these advantages add up to make web browsing safer, more private, and easier to use. There are many advantages of using Docker to implement a container-based browser; these include improved security, privacy, portability, testing and development, and flexibility. The main benefit is that each browser session is isolated within its own Docker container, effectively containing malware and other risks. This method strengthens overall security by guaranteeing that, even in the event that a session is compromised, the integrity of the host system is preserved.

Users' most important issue, privacy, is greatly enhanced by containerization, which makes sure that every surfing session runs on its own. By preventing data leaks between sessions, this segregation protects private user information. Knowing that their sessions are isolated and their privacy is protected, users can participate in online activities with confidence. Another argument in favor of Docker's use in browser implementation is its portability. Because browser sessions are contained within containers, moving between environments is made easy and constant performance and user experience are guaranteed regardless of the underlying infrastructure. For users who frequently switch between devices or need to access their browser configurations from other locations, this portability is especially helpful. To sum up, a container-based browser that integrates Docker offers a wide range of benefits. By segregating browser sessions, it strengthens security while simultaneously protecting user privacy, ensuring portability across various environments, streamlining the testing and development process, and providing unparalleled flexibility in controlling and configuring browser versions. All of these advantages add up to make web browsing safer, more private, and more convenient.

II.BACKGROUND

A. SYSTEM OVERVIEW

The proposed containerized browser, built upon Docker, presents an innovative solution addressing the critical aspects of security, privacy, and development efficiency in web browsing. By leveraging Docker containers, each browsing session is isolated, ensuring robust security measures and safeguarding user privacy. This isolation mechanism enhances security by containing any potential threats within individual containers, thereby minimizing the risk of data breaches or unauthorized access. Additionally, the system's emphasis on privacy is underscored by its ability to isolate browsing activities, preventing cross-session tracking and enhancing user confidentiality.

Furthermore, developers benefit from the streamlined process facilitated by Docker's portability, enabling consistent configurations across diverse environments. This portability aspect significantly enhances development efficiency by simplifying the testing and deployment of web applications. With Docker, developers can ensure that applications behave consistently across different platforms, reducing compatibility issues and accelerating the development lifecycle. Moreover, the system's versatility allows for effective version control and easy modification of browser settings, offering enhanced customization options for users. This flexibility empowers users to tailor their browsing experience according to their preferences, further enhancing security, privacy, and user-friendliness. In summary, the containerized browser not only prioritizes security and privacy through session isolation but also significantly improves development efficiency by harnessing Docker's portability and configuration consistency, ultimately creating a more secure, private, and efficient web browsing experience.

B. CLIENT SIDE INTERFACE

The client-side interface of the containerized browser is meticulously crafted to offer users a seamless and intuitive experience. With a user-friendly graphical interface at their disposal, individuals can effortlessly interact with the browser, initiate browsing sessions, and fine-tune settings within the isolated Docker containers. This interface prioritizes simplicity and ease of use, enabling users to securely navigate the web, customize browser configurations, and capitalize on the advanced privacy features inherent in containerization technology. By focusing on accessibility and user-centric design principles, the interface ensures that users can navigate the browser comfortably, empowering them to enjoy a positive and fulfilling browsing experience.

Moreover, the interface's emphasis on simplicity and intuitiveness extends to its functionality, allowing users to easily modify browser settings and preferences while benefiting from the enhanced privacy measures facilitated by containerization. Through clear and intuitive navigation options, users can seamlessly manage their browsing experience, adjusting settings to suit their individual needs and preferences. This user-centric approach not only enhances usability but also fosters a sense of control and empowerment among users, enabling them to make informed decisions about their online privacy and security. Overall, the client-side interface of the containerized browser embodies a commitment to user satisfaction, ensuring that individuals can navigate the web confidently and securely while enjoying a seamless and intuitive browsing experience.

C. INTERNAL INTERFACE

The internal interface or backend of the containerized browser is intricately engineered to manage the orchestration and communication between Docker containers, forming the backbone of the entire browsing system. This backend infrastructure encompasses a robust system designed to handle various tasks, including the isolation of browsing sessions, implementation of security protocols, and facilitation of web application deployment for developers. Leveraging Docker's capabilities, the backend system ensures the consistent configurations of containers, enabling efficient version control and effortless modification of browser settings. It serves as the driving force behind creating a secure, private, and development-friendly environment by seamlessly integrating Docker functionalities into the browser architecture, thereby laying the groundwork for a smooth and reliable browsing experience.

Furthermore, the backend system's intricate design and functionality enable it to manage complex operations with precision and efficiency. By orchestrating the communication between Docker containers, it establishes a secure and isolated environment for each browsing session, safeguarding user privacy and enhancing security measures. Moreover, the backend infrastructure streamlines the deployment process for web applications, empowering developers to test and deploy their creations with ease. Through its cohesive integration of Docker functionalities, the backend system plays a pivotal role in

optimizing the browsing experience, ensuring that users can browse the web securely and efficiently while providing developers with the tools they need to innovate and create.

III.RELATED WORK

Several research papers have played an important role in shaping the development of docker containers. The research "Container Based Analysis Tool for Vulnerability Prioritization in Cyber Security Systems" [1] by M. Walkowski, M. Biskup, A. Szewczyk, J. Oko, S. Sujecki contributed significantly. The paper presents the development of a container-based analysis tool tailored for vulnerability prioritization in cyber security systems. Utilizing containerization as the core development approach, the software offers a scalable solution that enhances data processing speed, consequently reducing the time required for vulnerability remediation. Through containerization, the software architecture allows for seamless scaling, enabling efficient utilization of resources and accelerating the analysis process. The study's results affirm that substantial computational time savings can be achieved by scaling up the number of containers, highlighting the effectiveness of this approach in improving efficiency and productivity in vulnerability management tasks. Moreover, the paper underscores the suitability of the developed software for deployment within cloud computing environments. The inherent scalability and flexibility afforded by containerization align perfectly with the dynamic nature of cloud infrastructures, where resources can be provisioned and scaled on-demand. By leveraging container-based analysis tools in cloud environments, organizations can effectively manage and prioritize vulnerabilities, optimizing their cyber security posture while minimizing remediation timeframes. This convergence of containerization and cloud computing represents a significant advancement in vulnerability management practices, offering a robust and adaptable solution for addressing emerging cyber threats in an increasingly interconnected digital landscape.

Additionally, the research "Isolation in Docker through Layer Encryption" [2] by Ioannis Giannakopoulos, Konstantinos Papazafeiropoulos, Katerina Doka and Nectarios Koziris has helped understand isolation through docker. In this study, Docker emerges as a prominent solution for containerization, yet its utilization raises concerns regarding the exposure of sensitive data to potential threats from privileged users due to its reliance on union-capable file systems. To mitigate these risks, we propose a novel secure Docker image manipulation mechanism. Our approach revolves around the implementation of data-at-rest encryption throughout the entire lifecycle of Docker images, encompassing image creation, storage, and usage. By encrypting sensitive data on disk, we fortify the security posture of Docker containers, significantly reducing the likelihood of unauthorized access or data breaches. Furthermore, our proposed mechanism incorporates a selective encryption feature, which enhances the distribution and migration of Docker images. This selective encryption mechanism allows for the encryption of specific layers within Docker images, thereby preserving confidentiality while facilitating efficient image distribution and migration processes. By encrypting only the necessary layers containing sensitive data, we strike a balance between security and performance, ensuring that the system remains both robust and agile in operation. Ultimately, our system offers a comprehensive solution for securely distributing and maintaining encrypted sensitive data within Docker containers, bolstering data protection measures with minimal impact on performance.

IV. PROPOSED SYSTEM

A. System Overview

The proposed container-based web browsing system is designed with a multifaceted goal of bolstering security, privacy, portability, testing, development, and flexibility within web browsing. Its core principle revolves around isolating each browser session within its Docker container, a measure that significantly enhances security and privacy for users. By confining browsing activities within these containers, the system ensures that potential threats or vulnerabilities are contained, minimizing the risk of data breaches or unauthorized access. Moreover, this approach offers a heightened level of privacy, as each session operates independently, preventing cross-session tracking or data leakage.

In addition to its security and privacy benefits, the system prioritizes portability, enabling users to maintain a consistent browsing experience across different environments. With Docker containers, users can seamlessly transition between devices or platforms while preserving their personalized browser settings and configurations. This portability feature not only enhances user convenience but also contributes to a more cohesive and integrated browsing experience. Furthermore, the containerization approach facilitates testing and development efforts by providing a controlled and reproducible environment. Developers can conduct rigorous testing without fear of interference or contamination from external factors, leading to more robust and reliable web solutions. The system's flexibility is further underscored by its ability to manage various browser versions, allowing users to tailor their browsing experience to their preferences while ensuring compatibility with different web applications and services. Overall, the proposed container-based web browsing system represents a comprehensive solution that addresses multiple dimensions of web browsing, from security and privacy to testing and flexibility, ultimately enhancing the user experience in diverse contexts.

B. System Architecture

The system architecture consists of three primary entities: User, Container (Docker), and Docker (representing the Internet). The user interacts with the system, initiating requests. The Docker container manages the isolated web browsing environment, ensuring security and resource efficiency. The Docker entity represents the broader network context where the containerized web browser interacts with online content. Data flows are initiated by user interactions, processed within the container environment, and involve interactions with the external entity (Internet).

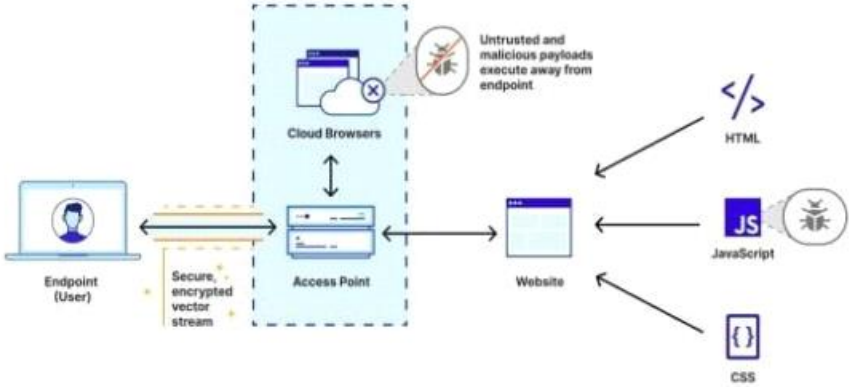

Figure 1. System Architecture

C. Use of Docker and Kubernetes

In the system architecture, the integration of Docker and Kubernetes serves as a cornerstone, enabling efficient containerization and orchestration processes. Docker, as a containerization platform, is instrumental in encapsulating the web browser and its dependencies within containers. This encapsulation ensures that the browser operates consistently and reproducibly across various environments, regardless of the underlying infrastructure. By leveraging Docker, the system achieves a level of standardization that enhances reliability and simplifies deployment processes.

Complementing Docker, Kubernetes functions as a robust container orchestration platform, further optimizing the system's performance and scalability. Kubernetes facilitates the management, scaling, and deployment of containerized applications, providing automated mechanisms for load balancing, self-healing, and resource allocation. This orchestration capability not only streamlines operations but also enhances the system's scalability, allowing it to adapt dynamically to fluctuating workloads and user demands. The combined utilization of Docker and Kubernetes imbues the system with enhanced reliability, scalability, and ease of deployment, laying the foundation for a robust and efficient container-based browsing solution.

D. Containerization of Other Applications

The system's containerization capabilities extend beyond web browsing to encompass a diverse range of applications, such as code editors and temporary Microsoft Office instances. Containerizing code editors offers developers the advantage of working within isolated environments, effectively preventing conflicts that may arise from different projects and their respective dependencies. This isolation ensures that each project remains self-contained, minimizing the risk of unintended interactions and streamlining the development process. By leveraging containerization for code editors, developers can maintain a high level of productivity and efficiency while mitigating potential issues associated with software development workflows.

Moreover, the system facilitates the use of temporary Microsoft Office containers, providing sandboxing for document editing tasks. By encapsulating Microsoft Office instances within containers, users can edit documents within secure, isolated environments, free from potential threats or vulnerabilities. This sandboxing approach enhances document security and confidentiality, safeguarding sensitive information from unauthorized access or manipulation. Additionally, the use of temporary containers ensures that any changes or modifications made to Office documents are contained within a controlled environment, minimizing the risk of data loss or corruption. Overall, the expansion of containerization to include code editors and temporary Microsoft Office instances enhances the security, efficiency, and reliability of various applications, contributing to a more robust and streamlined user experience.

E. Innovation

The innovative features of the system are multifaceted, with its foundation built upon the integration of Docker and Kubernetes. This integration establishes a robust containerization and orchestration framework, enabling seamless deployment, scaling, and management of applications. By leveraging Docker and Kubernetes, the system not only enhances operational efficiency but also provides a scalable solution capable of handling increasing workloads without compromising performance or reliability. This advanced framework ensures that applications can be deployed and managed with ease, aligning with modern demands for agile and scalable solutions in software development and deployment.

Additionally, the system's approach to containerizing a diverse range of applications, from web browsers to code editors and temporary Microsoft Office instances, demonstrates its versatility and adaptability. Users benefit from a consistent, secure, and isolated environment across various applications, fostering a unified and streamlined user experience. This versatility not only enhances usability but also promotes security and reliability, as each application operates within its own encapsulated environment, minimizing the risk of conflicts or security breaches. By offering a comprehensive suite of containerized applications, the system provides users with the flexibility to work efficiently and securely across different tasks and workflows, further enhancing productivity and user satisfaction.

Furthermore, the system's commitment to innovation is evident in its forward-thinking design philosophy, which extends beyond traditional web browsing to include a wide array of applications. This forward-looking approach addresses the evolving needs of the industry, providing a comprehensive solution for secure, flexible, and efficient application deployment and usage. By embracing containerization as a core principle, the system anticipates future trends and technologies, positioning itself as a cutting-edge solution that can adapt to the ever-changing landscape of software development and deployment.

V. CONCLUSION

The adoption of Docker containers for testing web applications has revolutionized the development process, significantly enhancing efficiency and reliability. Developers and testers have experienced a marked increase in productivity, as the controlled environment ensures that modifications and experiments are isolated, preventing interference with active browsing sessions. This separation allows for focused testing without the risk of unintended consequences, ultimately leading to the delivery of higher-quality and more dependable web solutions. Moreover, the encapsulation provided by Docker facilitates streamlined version control, simplifying the management of different browser configurations and enabling seamless adaptation to diverse online activities. With Docker, developers and testers can confidently experiment and iterate, knowing that their changes are contained within a controlled environment, thus contributing to the overall robustness and stability of web applications.

On the user front, Docker containers offer unparalleled flexibility and customization options, empowering users to tailor browser settings according to their preferences. This level of customization enhances the user experience by providing a personalized browsing environment that caters to individual needs and workflows. Furthermore, Docker's intuitive design and user-friendly interface make navigating and utilizing container-based browsers a breeze, even for less technically inclined users. Advanced security features, typically associated with Docker containers, are readily accessible without imposing technical barriers, ensuring that users can browse the web securely and with peace of mind. The ease of customization and the seamless integration of security features contribute to a positive user experience, fostering increased adoption and satisfaction among users. Additionally, the observed collaboration among businesses, developers, and the broader community further enriches the ecosystem surrounding Docker-based browsing solutions. Active engagement and contribution from users and contributors alike drive continuous improvements and innovations, ultimately enhancing the overall browsing experience for all stakeholders involved.

VI. References

[1] M. Walkowski, M. Biskup, A. Szewczyk, J. Oko and S. Sujecki, "Container Based Analysis Tool for Vulnerability Prioritization in Cyber Security Systems," 2019 21st International Conference on Transparent Optical Networks (ICTON), Angers, France, 2019, pp. 1-4, doi: 10.1109/ICTON.2019.8840441.

[2] I. Giannakopoulos, K. Papazafeiropoulos, K. Doka and N. Koziris, "Isolation in Docker through Layer Encryption," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 2017, pp. 2529-2532, doi: 10.1109/ICDCS.2017.161.

[3] W. Felter, A. Ferreira, R. Rajamony and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA, USA, 2015, pp. 171-172, doi: 10.1109/ISPASS.2015.7095802.

[4] V. K. Vavilapalli et al., "Apache hadoop yarn: Yet another resource negotiator," in Proceedings of SoCC. ACM, 2013, p. 5.

[5] B. Hindman et al., "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center." in NSDI, vol. 11, 2011, pp. 22–22.

[6] A. Verma et al., "Large-scale cluster management at Google with Borg," in Proceedings of Eurosys. ACM, 2015, p. 18.

[7] "Eclipse Che," https://www.eclipse.org/che/.

[8] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," Linux Journal, vol. 2014, no. 239, p. 2, 2014.

[9] "Docker Hub," https://hub.docker.com/.

[10] "OverlayFS," https://www.kernel.org/doc/Documentation/filesystems/overlayfs.txt

[11] "dm-crypt," https://www.kernel.org/doc/Documentation/devicemapper/dm-crypt.txt

[12] https://blog.cloudflare.com/browser-isolation-for-teams-of-all-sizes/