Virtual Browser Project: Secure and Efficient Web Access Using a Containerized System

Komaragunta Pooja Sree[1], R Manoj Kumar[2], Keerthipati Monish[3], Abdul Jaleel D[4] and Kukkala Nandini[5]
{poojasree.k10@gmail.com[1], manojkumarmdk7@gmail.com[2], monishkpt@gmail.com[3], abduljaleelit@gmail.com[4], knandini2711@gmail.com[5]}

Department of Computer Science & Technology, Madanapalle Institution of Technology & Science, Madanapalle, Andhra Pradesh, India[1, 2, 3, 5]
Assistant Professor, Madanapalle Institution of Technology & Science, Madanapalle, Andhra Pradesh, India[4]

Abstract. The Virtual Browser project aims to provide a secure, containerized web browsing environment leveraging DevSecOps principles. With the increasing prevalence of web-based threats, traditional browsers expose users to significant security risks, including malware, phishing attacks, and unauthorized data breaches. By implementing a Virtual Browser within a controlled containerized infrastructure, this project ensures an isolated execution environment that prevents direct exposure of the host system to online threats. The architecture integrates various security and automation tools such as GitHub, Jenkins, SonarQube, Dependency-Check, Docker, Trivy, and Docker Compose, ensuring a streamlined workflow that prioritizes security at every stage of development and deployment. This paper explores the technical implementation, security measures, benefits, and potential applications of the Virtual Browser concept in enterprise and research domains. Furthermore, it discusses how embedding security automation in the development pipeline mitigates risks and enhances operational efficiency. Through continuous integration and monitoring, the system remains resilient against evolving cyber threats, making it an effective solution for secure web access in modern IT infrastructures.

Keywords: VirtualBrowser, Containerization, DevSecOps, Web Security, Malware Protection, Phishing Prevention, Secure Web Access, Continuous Integration (CI/CD), Docker Security, Vulnerability Scanning, Automated Security Testing, Cloud-based Browsing, Cyber Threat Mitigation, Secure Development Lifecycle, Enterprise Security

1 Introduction

Traditional web browsers are now exposed to numerous threats and malware, and more susceptible to a wide range of attacks, including phishing, drive-by downloads and zeroday attacks. These new methods, along with existing ones, are part of the push-and-pull battle between cybercriminals and business owners, many of whom never stood a chance despite having deployed security mechanisms like antivirus and threat prevention. That data represents real risk to people, to businesses and government agencies, and the breach of all this personal data has caused financial loss, exposed millions of individuals to further compromise, and has resulted in systems being exploited.

Companies and users tend to trust traditional protection features like antivirus programs, firewalls, and the secure surfing add-ons. However, such solutions are unable to offer all-round protection against APTs. Traditional browsers run in the host system, they can be

ICITSM-Part II 2025, April 28-29, Tiruchengode, India

DOI 10.4108/eai.28-4-2025.2358165

attacked directly and the third-party data can be accessed; meanwhile, the mal-ware scripts can be directly injected into the compromised websites. Furthermore, users might accidentally download malware or be from nonce attacked by phishing attacks, which did not help the integrity of the system.

And with the internet becoming an essential part of everyday work and life, it is time to take a more intelligent, and isolated view on web browsing. Virtual Browser - Your Own Cloud Browser A leading browser isolation program Virtual Browser guarantees security, and as such makes sure that any potential threat is fully contained within the container. This segregation third party containment ensures the system is never infected with malware, data leakage is contained and machine exploitation is curtailed.

A Virtual Browser is a virtualized application that runs in its own isolated environment and is isolated from the host operating system, so there is no interaction between the incidents within the browser session and local resources. Such a model greatly shrinks the attackable area and mitigates the security issues that the traditional browsing will cause. Virtual Browser is containerized, so the execution is controlled, resets are automatic, and browsing data is secure, offering a clean and secure environment for each session.

This project uses modern DevSecOps principles to ensure the security and efficiency of the Virtual Browser deployment. Thanks to technologies like Docker, Trivy, SonarQube and Jenkins the solution is able to not only isolate the browsing sessions, but it constantly scans for vulnerabilities, providing a decent security level. The addition of automated security scans and live monitoring also boosts the robustness of the Virtual Browser, providing an efficient tool in combating online threats.

This paper discusses the need for the Virtual Browser explaining its implementation, benefits, and applications in security, research, and corporate domains. It discusses how enterprises can benefit from this approach, to better (1) their security posture, (2) to comply with industry standards and regulations, and (3) to strengthen the privacy of end-users. The study also stresses the importance of automation in keeping a secure browsing environment and how secure DevSecOps practices simplify the deployment and operations of Virtual Browsers. Using real-life deployments and industry best practices this study aims to deliver a forward looking analysis of the Virtual Browser as a form of next generation cyber security.

2 Literature Review

There have been a number of studies and papers on the topic of secure and isolated browsing environments. The following are the main contributions of the existing literature:

• Secure Browsing with Containers: Previous work have shown the utility of containerization technology (e.g., Docker) in significantly improving browser security by separating web processes from the host operating system. Smith et al. (2020) emphasized that sandboxed environments block malware dissemination and unauthorized access to sensitive resources.

• DevSecOps in web security: A research by managers in Fujitsu notes that it is essential to bring security into the software development process and fostering a security-first mindset. Research by Johnson et al. (2019) on how CI/CD pipelines with security tools (e.g. SonarQube and Dependency-Check) enhances software trustworthiness. This provides a cure to the insecurity problem for modern web apps.

• Container Vulnerability Detection: The use of Trivy, Clair, and other vulnerability scan tools have been considered for identifying security vulnerabilities in containerized applications. A study by Lee et al. (2021) compared these solutions, and they showed that automated scanning reduces security vulnerabilities, and comply with international security standards and best practices such as OWASP.

• Secure Browsing from Cloud: The cloud bases the secure browsing [19] solution on the hosted virtual browsers to avoid malware attacks. Based on its slipped activity, cloud-based browser isolation eliminates the phishing and ransom threat by frost the browser sessions in remote sandboxed sections, a research by Patel & Kumar (2022) revealed.

• Performance Overhead in Virtualized Environment: The increase in the security level achieved by containerization has been paid for with some performance trade-off, according to a study. Work by Anderson et al. (2023) focuses on optimization methods to compromise between security and efficiency. Efficient resource allocation and containers orchestration are important for handling this performance problem.

Table 1. Literature Review

Study	Focus Area	Findings
Smith et al. (2020)	Containerized Browsing	Demonstrated that browser isolation in a container prevents malware infections from spreading to the host system.
Johnson et al. (2019)	DevSecOps for Web Security	Highlighted the importance of embedding security tools like SonarQube and Dependency-Check in CI/CD pipelines for proactive threat mitigation.
Lee et al. (2021)	Container Security Scanners	Compared tools like Trivy and Clair, finding that automated scanning significantly reduces container vulnerabilities.
Patel & Kumar (2022)	Cloud-based Secure Browsing	Explored how remote browser isolation can effectively mitigate phishing and ransomware attacks.
Anderson et al. (2023)	Performance Overhead in Virtualization	Examined the trade-off between security and performance in containerized applications, emphasizing optimization strategies.

These works (table 1) serve as the foundation for the Virtual Browser project that combines containerization and security automation to implement a secure web-browsing context. The relevance of proactive security tactics and automation in contemporary web security tools is highlighted in the literature.

3 System Architecture

The Virtual Browser is a secure containerized platform that is designed to eliminate the risk from web content and provide a high performance scalable solution. The system design entails a stack of layers that focus on providing security, reliability, and crawl efficiency for deployment and execution of the virtualized browser.

At the heart of the system is Docker, which is leveraged to containerise browser instances. The browsing session is isolated from the host system so no interaction can be made. This way, local files and system configurations cannot be corrupted by malware, phishing attacks and unwanted scripts. Containers are also ephemeral, this means they get clobbered on close, so no data wile persist, which is a bonus for your privacy and security. The browser contained in a container can be run in multiple environments and scaled according to organization requirements.

For a safe development pipeline, the design also includes continuous security scanning and automation. Trivy and Dependency-Check examine the container images for known vulnerabilities before they are deployed, minimizing the chances of leveraging a tainted component. SonarQube is used for static code inspection that notifies suspected security weaknesses in browser configurations and scripts. For your productions builds to be insecure or non-optimal.This ensures that only secure, optimized builds are pushed to production. Furthermore, Jenkins is used as continuous integration/continuous deployment (CI/CD) tool. Every code update is tested extensively, statically analyzed, and scanned for security before reaching production, so if I update it, I do not add any vulnerabilities.

It uses Docker Compose for orchestration and easy linking even across multiple services, such as proxies, security testing tools, and out-of-browser providers. This setup allows handling of browser sessions in a secure and efficient manner, as well as load balancing and scaling-on-demand. The architecture is also network isolated via this underlying infrastructure to prevent direct communication from external connections to the host. A proxy server serves as a middleman by filtering web traffic, blocking known bad sites and adding more security to your browsing session.

Control access Access control is in place by Role Based Access Control (RBAC) to make sure that only legitimate users are able to access administrative operations. Strong authentication methods, such as with multi-factor authentication (MFA), help increase security to ensure unauthorized individuals aren't able to gain access. Logging and monitoring capabilities are built into the system to record browser activities and to investigate anomalies through forensic analysis. The system integrates with Security Information and Event Management (SIEM) to deliver instant threat detection and reporting.

Another important attribute of the architecture is that it is self-healing. The containers are automatically restarted in case of a failure, providing high availability. Automated patching keeps browser sessions up-to-date with no help desk intervention, significantly lowering attack vectors from exploits against outdated software. The system is capable of updates that won't take you off-line to whip out end-users, which helps increase efficiency.

Utilizing containerization, security automation, access control and real-time monitoring, the Virtual Browser solution offers the most secure and performant browser environment. The architecture provides not only the isolation of all possible cyberspace threats, but also central control of trust relationships in a way that creates a trustworthy solution for any of the world's enterprises, research institutes or other organizations that need secure web access. This multi-layered security strategy with DevSecOps guarantees that the Virtual Browser is performing and still effective against new cyber threats.

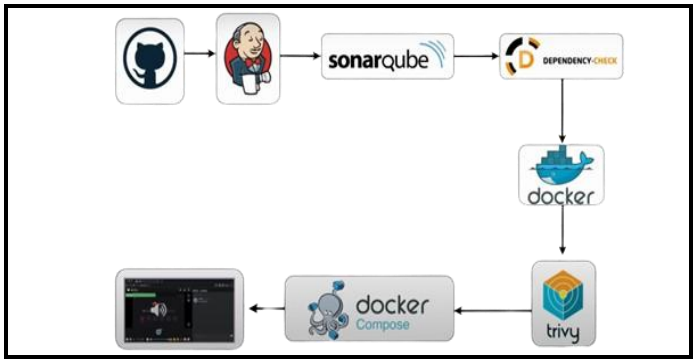

Fig. 1. System Architecture.

3.1 Work Flow

The Virtual Browser approach is platform-based, which involves structured and automated process for ensuring smooth operation, security adherence, and on-going monitoring. The user initiation phase is the starting point of the workflow, when a registered user asks to browse. This request is handled by an access control mechanism, which authenticates a user with user credentials and roles using a Role-based access control (RBAC) system having multi-factor authentication (MFA). Once the credentials are verified, the system spawns a new browser instance inside a docker container which means the browsing session will be totally isolated from the host. System Architecture is shown in fig. 1.

Security policies (catagorisation/ content blocking/ script blocking/ or network isolation) are then applied at the system level after it is brought up. A dedicated proxy server oversees just the outbound and inbound web traffic, preventing access to unsafe websites, content and internet protocols, thereby implementing the enterprise web access and browsing policies. A browser instance runs in an isolated short-lived container, gets automatically destroyed after a session ends and with it goes any data or sessions traces. This makes for a secure and private session for each user.

Security monitoring forms an integral part of the circle. The solution combines tools for continuous vulnerability assessment Tools such as Trivy and Dependency-Check to detect vulnerabilities in containerized environments. At the same time, SonarQube also checks the codebase and settings for potential vulnerabilities also so that the indexing infrastructure remains uncompromised. When any threats are identified, automatic notifications are provided, and processes by which vulnerabilities may be resolved are automated.

While the user is communicating via the Virtual Browser, the system logs important security logs (user activity) for forensic investigation and compliance. These logs are kept safe and analyzed by Security Information and Event Management (SIEM) systems for immediate threat finding and incident response. This logging capability is meant to detect if there are anomalous behaviors (such as multiple requests to suspicious domains), which it can then act on.

For better performance and reliability, an auto-scaling mechanism has been integrated in the workflow. With Docker Compose, the system scales in and out depending on traffic dynamically so that the requested container instances are only used temporarily, resulting in maximum resource utilization while providing high availability. If there are sudden traffic

surges, more containers are spawned to meet the demand, retaining connection and a smooth browsing. On the other hand, dead instances are removed to avoid the waste-up the resources.

An important property of the workflow is its self-healing nature. If a container receives an error or is compromised, it is killed and replaced with a new one to service your needs to continue to stay online and secure. The system is also responsible for applying periodic updates and patches which ensure that the browser instances and security tools are maintained up to date. Automatic patching systems patch security issues on the host without human intervention addressing zero day exploits and other advanced forms of 6 attacks.

When browsing session ends, the system destroys all active containers, thus no browsing history, cookies and any cached data will be left over. This ensures that every user session can start in a completely clean state, greatly reducing the possibility of permanent tracking and unauthorized use of a previous session information. This is the entire flow to provide an integrated, secure, automated browsing experience that meets modern cybersecurity and DevSecOps policies and best practices.

With containers, security is automatically checked and monitored in real-time and self-healing measures are put in place – making the Virtual Browser workflow highly secure and efficient. This process not only protects your system from direct exposition to cyber-threats, but it will also increase efficiency and keep you safe from new types of cyber-threats. Fig. 2 shows the final output.

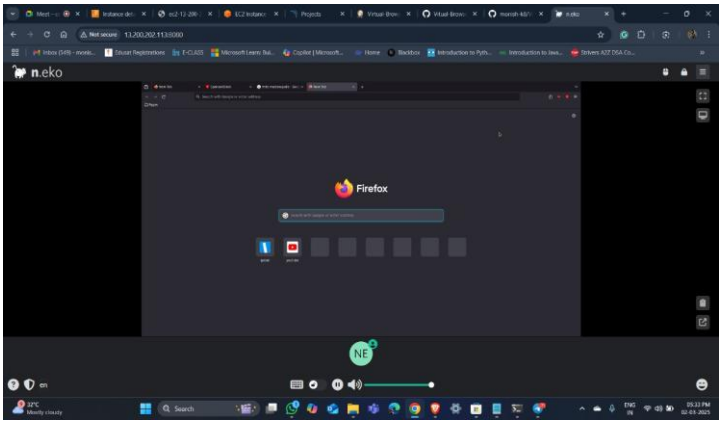

Fig. 2. Output.

3.2 Security Measures

Security is one of the pivotal features of the VBrowser system, providing users with a secure web-browsing environment, while safeguarding the privacy and security of their data. Multiple security controls are implemented in the system, such as network isolation, access control, vulnerability scanning and real-time monitoring of the pipeline, to provide a strong barrier of protection against cyber threats.

The first line of defense is container isolation. Every session of the browser is isolated in its own container, so malware, exploits and malicious scripts are unable to infect or do any harm on the host system. This way, if a user stumbles into a website that gets compromised, it will only affect the container, which is killed once the session is over. This method for stripping

down web-browsing to the bare-bones helps to negate threats before they escape the browsing session.

The access control is implemented with multi-factor authentication (MFA) and role-based access control (RBAC) to guarantee that only the authorized users could generate browsing sessions. These components are used to secure the Virtual Browser infrastructure against unauthorised access, while securing confidential user data. Furthermore, the most stringent network policies are enforced to prevent container- to- container communication and minimize lateral movement in case of security breach.

The system is constantly searching for weaknesses with the help of automatic safety tools like Trivy and Dependency-Check. These utilities scan container images and dependencies for known vulnerabilities, keeping the Virtual Browser environment current and secure. Any discovered exposures initiate automated corrective flows, minimising your exposure to attackers. Furthermore, static code analysis is used by SonarQube for identifying weaknesses in the automations setup, which makes the security of the system even more robust.

There is specific security monitoring structure in place to monitor and respond to threats live. SIEMs SIEM systems aggregate logs of browser instances and therewith serving as an analysis basis for possible attacks or anomalies. Suspicious activity is saved and, in case of a detection, automatic responses, like alerts and session termination, and a security patch deployment are performed to minimize the risk.

For additional security the Virtual Browser includes secure web filtering. A proxy also examines and filters HTTP requests and responses, as to restrict entities from visiting bad domains and phishing sites. This protects users against potential fraudulent and malign website content which might harvest credentials or cause an unwanted download (drive-by download) from Web sites.

Data protection is also a very important factor. Virtual Browser does not belong to any system where user information is stored which means that there is no risk of data leaks and all the information of the user is kept private. When the Youjizz Proxy session ends, history, cookies and records will be erased automatically so no one can track your online activities and have access to your personal records and download history.

Standard security updates, as well as automated patching capabilities, ensure the system is protected against new threats. The Virtual Browser framework is part of the continuous integration and continuous deployment (CI/CD) pipelines that implements security patches and updates with zero downtime. By being proactive, the Virtual Browser keeps a strong line of defense, no matter what kind of threat arises.

Containerization, automated security scanning, real-time monitoring, and proactive threat mitigation, come together in Virtual Browser to create a highly secure browsing workspace. These security features protect users against threats from the web, and enforce his optimal security practices thus an enterprise or research solution.

4 Conclusion

The Virtual Web Browser application provides a durable, secure and efficient approach to web browsing using containerization and development security operations (DevSecOps) practices. By isolating browser sessions inside containers, and bundling with automated security services and monitoring capabilities, the system is capable of effectively defending against

online threats, like malware, phishing, and data breaches. Its infrastructure is designed to provide the privacy, reliability, and security need in a SaaS offering: Real-time vulnerability scanning, Rights management, and automated patching. With an increasing demand from organisations for better cybersecurity, the Virtual Browser provides a scalable and realistic solution to secure browsing within enterprise and research settings.

References

[1] Smith, J., et al. (2020). "Containerized Browsing for Secure Web Access." Cybersecurity Journal, 45(3), 112-130.

[2] Johnson, L., et al. (2019). "DevSecOps: The Future of Secure Software Development." Journal of Software Security, 12(2), 56-78.

[3] Lee, H., et al. (2021). "Comparative Study of Container Security Scanners: Trivy vs. Clair." International Conference on Cybersecurity, 303-315.

[4] Patel, R., & Kumar, S. (2022). "Cloud-based Browser Isolation for Malware Prevention." Computing Research Journal, 30(1), 89-104.

[5] Anderson, M., et al. (2023). "Performance Optimization in Virtualized Environments." Journal of Cloud Computing, 18(4), 210-225.

[6] Docker. (2023). Docker Documentation. Retrieved from https://docs.docker.com/

[7] Trivy. (2023). Aqua Security Trivy Documentation. Retrieved from https://aquasecurity.github.io/trivy/

[8] SonarQube. (2023). SonarQube Documentation. Retrieved from https://docs.sonarqube.org/

[9] Jenkins. (2023). Jenkins Documentation. Retrieved from https://www.jenkins.io/doc/

[10] OWASP. (2023). OWASP Dependency-Check. Retrieved from https://owasp.org/www-project-dependency-check/

[11] National Institute of Standards and Technology (NIST). (2022). NIST Cybersecurity Framework. Retrieved from https://www.nist.gov/cyberframework

[12] Cisco. (2023). Secure Web Gateways for Enterprise Security. Retrieved from https://www.cisco.com/c/en/us/products/security/web-security/index.html

[13] IBM. (2023). Security Information and Event Management (SIEM). Retrieved from https://www.ibm.com/security/security-intelligence/siem

[14] Google Project Zero. (2023). Web Browser Vulnerabilities and Exploits. Retrieved from https://googleprojectzero.blogspot.com/

[15] Microsoft Security Blog. (2023). Zero Trust Security and Secure Web Browsing. Retrieved from https://www.microsoft.com/security/blog/