future internet

MDPI

Article

Database Security and Performance: A Case of SQL Injection Attacks Using Docker-Based Virtualisation and Its Effect on Performance

Ade Dotun Ajasa *, Hassan Chizari * and Abu Alam

School of Computing & Engineering, University of Gloucestershire, The Park, Cheltenham GL50 2RH, UK; aalam@glos.ac.uk

* Correspondence: adeajasa@connect.glos.ac.uk (A.D.A.); hchizari@glos.ac.uk (H.C.)

Abstract: Modern database systems are critical for storing sensitive information but are increasingly targeted by cyber threats, including SQL injection (SQLi) attacks. This research proposes a robust security framework leveraging Docker-based virtualisation to enhance database security and mitigate the impact of SQLi attacks. A controlled experimental methodology evaluated the framework's effectiveness using Damn Vulnerable Web Application (DVWA) and Acunetix databases. The findings reveal that Docker significantly reduces the vulnerability to SQLi attacks by isolating database instances, thereby safeguarding user data and system integrity. While Docker introduces a significant increase in CPU utilisation during high-traffic scenarios, the trade-off ensures enhanced security and reliability for real-world applications. This study highlights Docker's potential as a practical solution for addressing evolving database security challenges in distributed and cloud environments.

Keywords: security; performance; virtualisation; databases

Academic Editor: Luis Javier Garcia Villalba

Received: 20 December 2024
Revised: 20 March 2025
Accepted: 21 March 2025
Published: 2 April 2025

Citation: Ajasa, A.D.; Chizari, H.; Alam, A. Database Security and Performance: A Case of SQL Injection Attacks Using Docker-Based Virtualisation and Its Effect on Performance. *Future Internet* **2025**, *17*, 156. https://doi.org/10.3390/fi17040156

1. Introduction

Database Management Systems (DBMSs) often store sensitive information, such as company secrets, financial data, and personal privacy details [1]. However, these systems are vulnerable to both external and internal threats. External threats include social engineering attacks, while internal threats arise from unauthorised access, privilege abuse, and vulnerabilities like SQL injection [2].

The authors of the research paper by Kaneko et al. [3] proposed a privacy-enhancing approach involving the deployment of resource-efficient Docker containers within cloud environments. This method effectively isolates sensitive information, enabling flexible implementation of enhanced privacy protections and management strategies as needed. Given the huge increase in the number of network security threats, such as vulnerabilities, attacks, data breaches, and privacy violations [4], this research aims to address these challenges by developing a robust security framework for databases. This framework will mitigate risks associated with external attacks [5], while also safeguarding user data from the destructive potential of SQL injection (SQLi) [6].

On the other hand, cloud computing has enabled distributed computing, outsourcing infrastructure installation, pricing, maintenance, and server scalability. This has led to a profitable computing industry, where data owners often relinquish control over their data despite significant financial investment [7]. To address this, a comprehensive evaluation of security frameworks is necessary, particularly in scenarios requiring frequent user access to

data for real-world applications [8]. While various research papers have explored the use of Docker as a framework, there remains a gap in understanding how to leverage Docker from a user-centric perspective to protect both user data and the underlying database. This research aims to bridge this gap by fostering trust between users and the authorities managing centralised databases.

This paper is organised with the following structure: Section 1 talks about the introduction of the Database, Docker as a framework, SQL injection (SQLi), research questions, research objectives, Docker containers, and security. Section 2 describes related works, research objectives, suitability of the Docker framework, Docker containers, and the security aims of the research. Section 3 consists of the implementation, the hypotheses, the type of variables, the schematic diagram of the laboratory, ethics approval and GDPR, the environment, databases, network analysis, and data collection. Section 4 presents the results, statistical analysis, tables, and graphs. Section 5 is a summary of findings, while Section 6 elaborates on the conclusions.

2. Related Works

Table 1 highlights the software used in related works, especially, the use of Docker images and Docker containers to conduct their implementation, just as this research paper has used the Docker framework for its implementation. The authors of the research paper by Zhao et al. [1] highlighted the significant challenges posed by data breaches in 2016. Real-time processing databases, designed to handle constantly changing workloads, are particularly susceptible to these threats. Moreover, both external and internal attacks pose serious risks to databases, often targeting sensitive information, such as commercial secrets, bank details, and personal privacy [1]. The authors of the research paper by Wagner et al. [2] further corroborated these findings in their 2017 research. Database Management Systems (DBMSs) are employed to process and store user data.

Furthermore, security mechanisms and access controls, such as audit logs, cannot always be relied upon to prevent data breaches. Users, both legitimate and malicious, may abuse their privileges in order to compromise database security. Databases must be capable of promptly detecting breaches and collecting evidence to facilitate investigations [2]. According to the authors Said and Mostafa [9], insider threats, particularly those involving the misuse of legitimate account privileges, remain a persistent challenge in database security. Detecting such breaches can be extremely difficult. To address this issue, Said et al. proposed an adaptive and efficient database intrusion detection algorithm inspired by the Negative Selection algorithm from artificial immune systems and the Danger Theory model [9].

Two recent studies published in 2021 by research authors Neto et al. [10] and Algarni et al. [11] emphasise the persistent challenges of data breaches and database security. The increasing reliance on the internet has exacerbated the risks of data leakage and cyber threats. The research paper by Neto et al. [10] analysed data breaches involving personal information between 2018 and 2019, including the 2019 cyberattack on Capital One, which exposed sensitive customer information [9]. Similarly, the research paper by Algarni et al. 2021 [11] examined the vulnerabilities of modern business systems to cybersecurity threats. While cybersecurity solutions can mitigate attacks on database storage, human errors, such as the loss or theft of devices containing sensitive data or accidental exposure of security credentials, remain significant contributors to security breaches. Common cyberattacks include cross-site scripting (XSS), privilege escalation, and SQL injection (SQLi).

The authors of the research paper by Mahrouqi et al. [12] conducted a notable study in 2016, simulating an SQLi attack in a virtual environment using tools such as VirtualBox, VMware Workstation, Wireshark, and GNS3. This research aimed to identify websites vul-

nerable to SQLi attacks. Subsequently, the authors of the research paper by Grubbs et al. [6] advocated for the development of strategies to limit the damage SQLi attacks inflict on user data stored in databases. The research paper by Williams et al. [13] underscored the importance of acquiring knowledge about software vulnerabilities, given the heavy reliance of academic institutions, the private sector, and government entities on database-related software. This research builds upon this foundation by proposing a method to mitigate SQLi attack damage to user data held in databases [6].

Hospitals and banks are among the primary victims of SQLi attacks [14]. Databases accessible via the internet are particularly vulnerable to various types of attacks [15]. As proposed in the research paper by Kaneko et al. [3], an innovative approach involves storing private information within Docker containers. These containers, which are resource-efficient and designed for cloud environments, enhance privacy protection and can be deployed as needed. This approach aligns with the increasing prevalence of network vulnerabilities, attacks, and data privacy issues in today's IT sphere, driven by advancements in network technology [4]. This research aims to develop a robust security framework that protects databases from both internal and external threats, specifically addressing SQLi attack mitigation [6].

Moreover, the widespread adoption of distributed computing via the cloud has transformed the computing industry. While outsourcing installation, maintenance, and scalability has proven financially advantageous, data owners often lack direct control over their data despite paying high costs for its security [7]. This research evaluates a security framework through use cases that simulate real-world scenarios, ensuring frequent and secure access to data [8].

The injection of carefully crafted malicious SQL code through web page input can potentially lead to the destruction of a database [11]. In contrast to the findings of this research paper, Wang and Reiter [16] and Hassanzadeh et al. [17] provide a more alarming assessment of data breaches and database security. A significant number of data breaches have been reported globally, with 3950 incidents occurring between November 2018 and October 2019, resulting in the exposure of 60% of victim identities, due to 1665 breaches in various credential databases [16]. This widespread issue has raised serious concerns among both companies and individuals. High-profile data breaches, such as those experienced by Yahoo in 2013 and 2014, LinkedIn in 2012, Marriott International in 2014 and 2018, and Equifax in 2017, demonstrate that many organisations continue to implement inadequate cybersecurity measures, despite advancements in security technology and increased awareness [17].

Similarly, NoSQL databases, which store data in a format different from relational tables, have also been targeted by hackers in 2021. While these databases disrupted the market with their scalability, performance, and availability, compromises were often made in other areas, including privacy. NoSQL databases are designed to handle unstructured data which lack a predefined organisation or data model. This flexibility, however, can come at the cost of privacy features. Data privacy, the right to control how data is collected and disclosed, is a critical concern that cannot be easily overlooked [18]. Furthermore, traditional database services have now moved online. In other words, a database is an organised collection of user data. After all, if the security employed for the database is capable of protecting the data that it is hosting, malicious attacks or threats from hackers can prevent the data being stolen. This was stated in 2021 in the research paper published by Crooks [19], though in 2016 an investigation on how to build a database that would not be vulnerable to internal or external attacks was proposed in the research paper by Toapanta et al. [5]. Meanwhile, it is possible to consider network problems and extremely high computer loads with the introduction of modern applications; however, there has

to be continuity in the flow of data to ensure that the service is guaranteed. The constant evolution of the scalability and availability of information technology (IT) services is why Docker containers can be of great value to the information technology industry [20].

Table 1. Comparison of this research paper and other research papers that have employed Docker.

Software	What the Software Was Used for
Docker machine [21]	Apache Airavata (an open source software suite).
Docker containers [22]	Run the Docker containers from different databases.
Docker containers [23]	Docker server and Docker client.
Docker container networking [24]	Monitoring server and anomaly monitoring system.
Docker framework [25]	SDN-Docker-based architecture for IoT.
Docker [26]	Virtual machine.
Docker [20]	DNS server, HAproxy, and mail server.
Docker containers [27]	Run the Docker containers.
Docker [28]	Optimise development and increase efficiency of methods.
Docker [29]	5G mobile networks using Docker containers.
Docker [30]	Picto Web would be used within a Docker container.

2.1. Suitability of the Docker Framework in an Environment with High Computational Demands

The authors of the research paper by Moysiadis et al. [31] consider the association of Docker containers as a virtual technology with traditional framing, cloud computing, and Information Communication Technologies (ICT). They also consider the advantages that could be derived from this, including reduction in production costs, boosting of productivity, better security, and scalability with regard to future upgrades and increasing performance. As a result, cloud computing can support the demand for handling large user data and a large number of end devices [31]. Another key thing to remember is that servers handling the increase in resources used relating to big data applications increase in storage due to increase in the applications being stored on the servers and balancing of the load within each server running these applications. Setting up of a Docker Swarm would be most appropriate to solve these problems as it can deploy multiple Docker containers on multiple computer hosts in a very short period [28]. The current debate, that was highlighted in the research paper by Singh et al. [28] published this year (2023), considers the merits of Docker containers as follows:

- A Docker Swarm efficiently deals with the deletion and duplication of containers [28].
- Docker containers work well with load balancing and when dealing with applications that are complex [28].
- A Docker Swarm can effectively manage the employment of Docker containers [28].

Correspondingly, the research paper by Zou et al. [24], in comparison to the research paper by Alyas et al. [32], also mentions the security side of Docker containers. The issues of the stability and security of the container have become important issues with the rejection of thousands of apps and websites, for example, the collapse of the Amazon cloud, which was built upon a virtual machine cluster and container [33]. More researchers are using Docker because of the advantages it has been reported as having over traditional virtual machines [24]. Furthermore, according to the research paper by Hersyah et al. [34], it is known that with the aid of Docker containers supported with automation, the availability, deployment, redundancy, and granting of authorisation can be applied securely. Addition-

ally, care should be taken to avoid becoming complacent when using cloud services [34]. To exemplify, the same tools that are meant to help with security and easy navigation within the cloud have been known to be reverse-engineered and used by hackers for attacks against the cloud services. Furthermore, the cloud service provider has to make sure that security at their end is up to date, with regard to their software, hardware, and their staff being highly trained. To this end, virtualisation and the cloud are very closely associated with each other [34]. In addition, the research paper by Qian et al. [35] refers to storing users' information in the cloud, which involves the inclusion of virtualisation and cloud computing. In other words, compared to the traditional way of utilising virtualisation, although Docker virtualisation performance expenses are lower, the deployment and delivery rate are faster. To clarify, with the aid of cloud computing, the following advantages can be gained: cloud computing can handle large amounts of data traffic and reduce the delay in network transmission [35]. Furthermore, the research paper by Singh et al. [28] supports the research paper of da Silva and Lima [36] by explaining how a Docker Swarm provides an architecture which is decentralised and fault tolerant, and with the aid of Swarm mode, we can create a Docker Swarm which consists of Docker hosts. In other words, a Docker Swarm can be seen as improving the quality of Docker regarding availability, security, maintainability, reliability, and scalability. A Docker Swarm can reassign containers if it discovers that any one of the containers has failed within the Docker Swarm [28].

2.2. Docker Containers and Security

The primary purpose of Docker containers is to isolate individual microservices effectively [31]. Docker provides a service called (https://hub.docker.com/ accessed on 20 March 2025), the world's largest library of container images, which allows users to share or search for container images. This service is accessible via the Docker Hub website. The authors of the research paper by Alyas et al. [32] support the findings of the research paper by Moysiadis et al. [31], while emphasising the importance of addressing security, sovereignty, and compliance in cloud computing. This highlights the need for solutions to resolve privacy and security concerns, which remain key barriers to the effective adoption of mobile cloud computing (MCC) in healthcare environments [37].

Moreover, a single computer can host only a limited number of virtual machines, whereas the same computer can run thousands of Docker containers simultaneously, offering superior scalability and mobility. Docker's ability to operate on almost any platform, coupled with its ease of deployment and maintenance, further underscores its advantages [24].

2.3. Aims of Research and Research Objectives

This research aims to design and evaluate a high-performance and secure database system. Specifically, the study focuses on three key objectives as presented below. By addressing these challenges, this research seeks to enhance the overall security and performance of database systems, safeguarding critical information and ensuring reliable service delivery.

- **Research Objective 1—RO1**—Investigation into the development of a database that would not be vulnerable to internal or external attacks and then developing a security framework.
- **Research Objective 2—RO2**—Proposing a way to limit the damage Structured Query Language injection (SQLi) causes to users' data that are being held in a database.
- **Research Objective 3—RO3**—Evaluating and proposing a security framework with the use case where users will have access to high-traffic applications, to closely represent a real world scenario.

This review involved searching for existing research, critically evaluating relevant studies, and synthesising the findings. The next step involved formulating alternative hypotheses (H_a) and selecting appropriate statistical tests. For example, to evaluate the effectiveness of Docker in mitigating SQL injection attacks, we launched controlled SQLi attacks against (https://www.acunetix.com/, Acunetix, accessed on 19 December 2024) and (https://github.com/digininja/DVWA, DVWA, accessed on 19 December 2024) databases (without Docker protection) and recorded the data exfiltration levels. Subsequently, we then implemented (https://www.docker.com/company Docker, accessed on 19 December 2024) protection for the same databases and repeated the SQLi attacks, measuring and comparing the data exfiltration in both scenarios. Statistical tests (e.g., *t*-tests) were then used to determine if the observed differences between the two settings were statistically significant.

We, therefore, used a quantitative approach, specifically, a true experimental design with a between-groups post-test-only structure. This design utilises two groups: a control group and an experimental group. Both groups are measured on a dependent variable, which reflects the outcome of interest. The independent variable, representing the manipulation introduced by the experiment, is applied only to the experimental group. Following the post-test, the main analysis focuses on the differences between the control and experimental groups in terms of the dependent variable. To determine if the observed differences are statistically significant, a *t*-test is employed. This test is suitable when comparing the means of two independent groups with normally distributed, continuous (metric) data (as outlined by (https://datatab.net/tutorial/paired-t-test DATAtab, accessed on 19 December 2024)).

3. Implementation

3.1. The Hypotheses

- *H_0—The null hypothesis—There is no significant difference between the means of the dependent variables—CPU_total_(%) for both (https://github.com/digininja/DVWA DVWA accessed on 19 December 2024) installed in Docker and (https://github.com/digininja/DVWA DVWA accessed on 19 December 2024) not installed in Docker, as well as (https://www.acunetix.com/ Acunetix accessed on 19 December 2024) installed in Docker and (https://www.acunetix.com/ Acunetix accessed on 19 December 2024) not installed in Docker.*
- *H_a—The alternative hypothesis—There is a significant difference between the means of the dependent variables—CPU_total_(%) for both (https://github.com/digininja/DVWA DVWA accessed on 19 December 2024) installed in Docker and (https://github.com/digininja/DVWA DVWA accessed on 19 December 2024) not installed in Docker, as well as (https://www.acunetix.com/ Acunetix accessed on 19 December 2024) installed in Docker and (https://www.acunetix.com/ Acunetix accessed on 19 December 2024) not installed in Docker.*

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu_1 \neq \mu_2$$

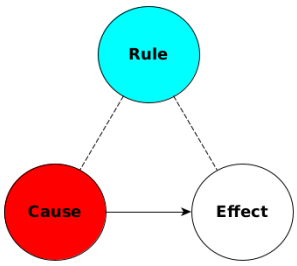
3.2. Justification Why CPU Is the Most Relevant Metric

Firstly, Table 2 explains the type of variables used: uptime_minute is the independent variable; CPU_total_(%) is what is going to be measured and is the dependent variable. Additionally, any confounding variables which are correlated with both the independent variable (uptime_minute) and the dependent variable (CPU_total_(%)) require to be held constant throughout the experiment. Here, the variable (Virtual operating system—VirtualBox) does not change. Secondly, Figure 1 explains deduction, which is synonymous with the quantitative method. Deduction can be expressed as, given a rule and the cause, deduce the effect [38]. If the rule is—during a Structured Query Language injection (SQLi) attack, the cause is—it is crucial that the uptime_minutes are measured during the Structured

Query Language injection (SQLi) attack and the *effect* will be a *CPU_total_(%)* increase in temperature. The author of the research paper [20] measured the (CPU_total_(%)) as one of the variables. Lastly, Figure 2 represents the schematic diagram of the laboratory which, was designed and built in VirtualBox [39].

Table 2. Type of variables.

Independent variables (cause)	uptime_minutes
Dependent variable (effect)	CPU_total_(%) (what is going to be measured)
Confounding variable—Correlation between the independent and dependent variables	Ubuntu 20.04.2 LTS
Controlled or constant variable—This experimental variable does not change	Virtual operating system—VirtualBox



(**a**) Rule, cause and effect

Deduction	**Example**
Rule	During an SQLi attack
Cause	uptime_minutes are crucial
Effect	*CPU_total_(%) will increase*

(**b**) An example of deduction

Figure 1. Deduction [38].

Table 3 illustrates a breakdown and the specifications used to carry out the implementation.

Table 3. Specifications of the host computer and virtual machines.

Hardware	**CPU**	**Ram**	**Graphics**	**Operating System**
Host	Intel i7	16GB	Intel/Nvidia 2GBRam	Parrot OS 4.11
VM Hacker	Intel i7	4096MB	VMSVGA	Oracle (64-Bit)—Parrot OS 4.11
VM Database—Docker	Intel i7	2048MB	VMSVGA	Ubuntu (64-Bit)
VM Database—No Docker	Intel i7	2048MB	VMSVGA	Ubuntu (64-Bit)

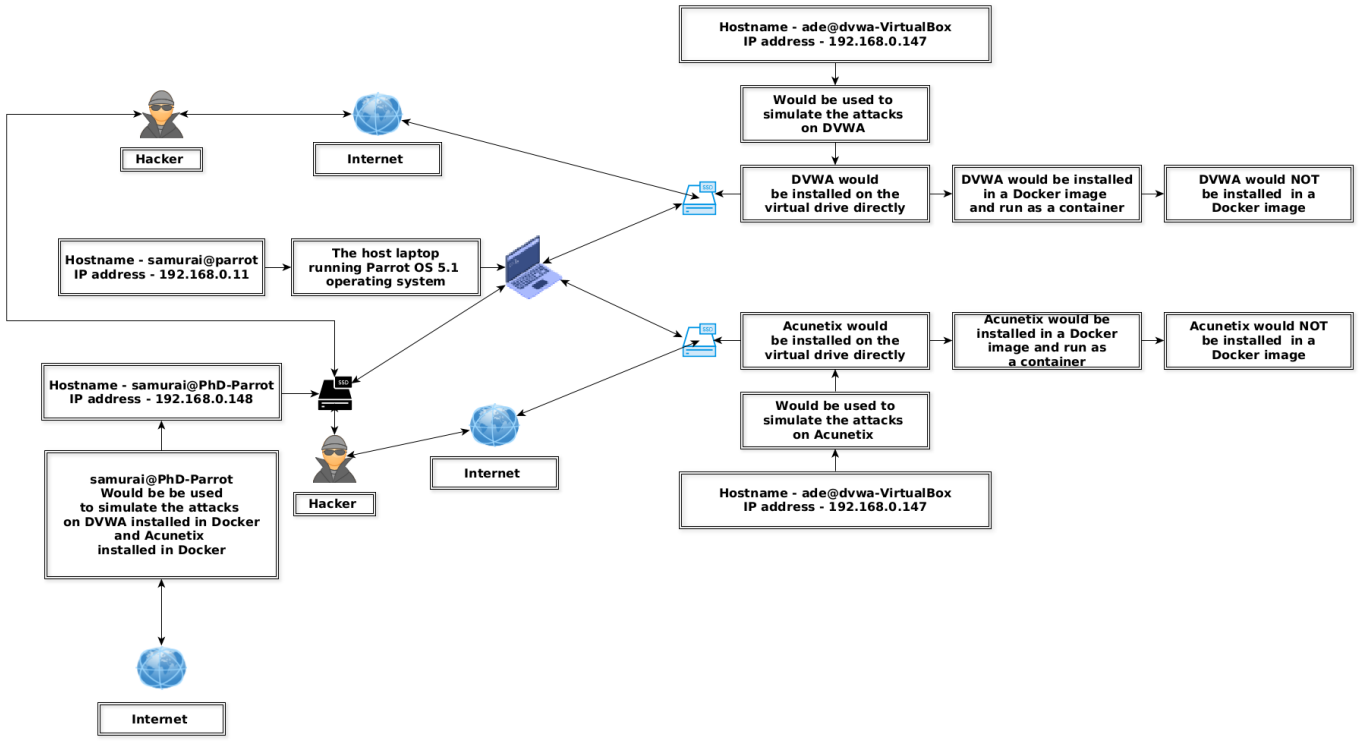


Figure 2. Schematic diagram of the laboratory. Note: all IP addresses in Figure 2 are unique addresses that identify a device on the local network.

3.3. SQL Injection Attack

Firstly, Figure 3 explains the schematics towards building the secure framework (quantitative research). Secondly, DVWA—Database Without Docker Installed—Figure 4 represents the gathering of information from the Damn Vulnerable Web Application, Figure 5 represents the launching of an SQL injection attack from the information gathered from Figure 4, and Figure 6 shows that the SQL injection attack was successful. Additionally, Figure 7 shows the running DVWA (Database with Locker Installed) in a Docker container, Figure 8 shows the gathering of information to launch an SQL injection attack, Figure 9 shows the launching of an SQL injection attack, and lastly, Figure 10 shows the SQL injection attack was not successful.

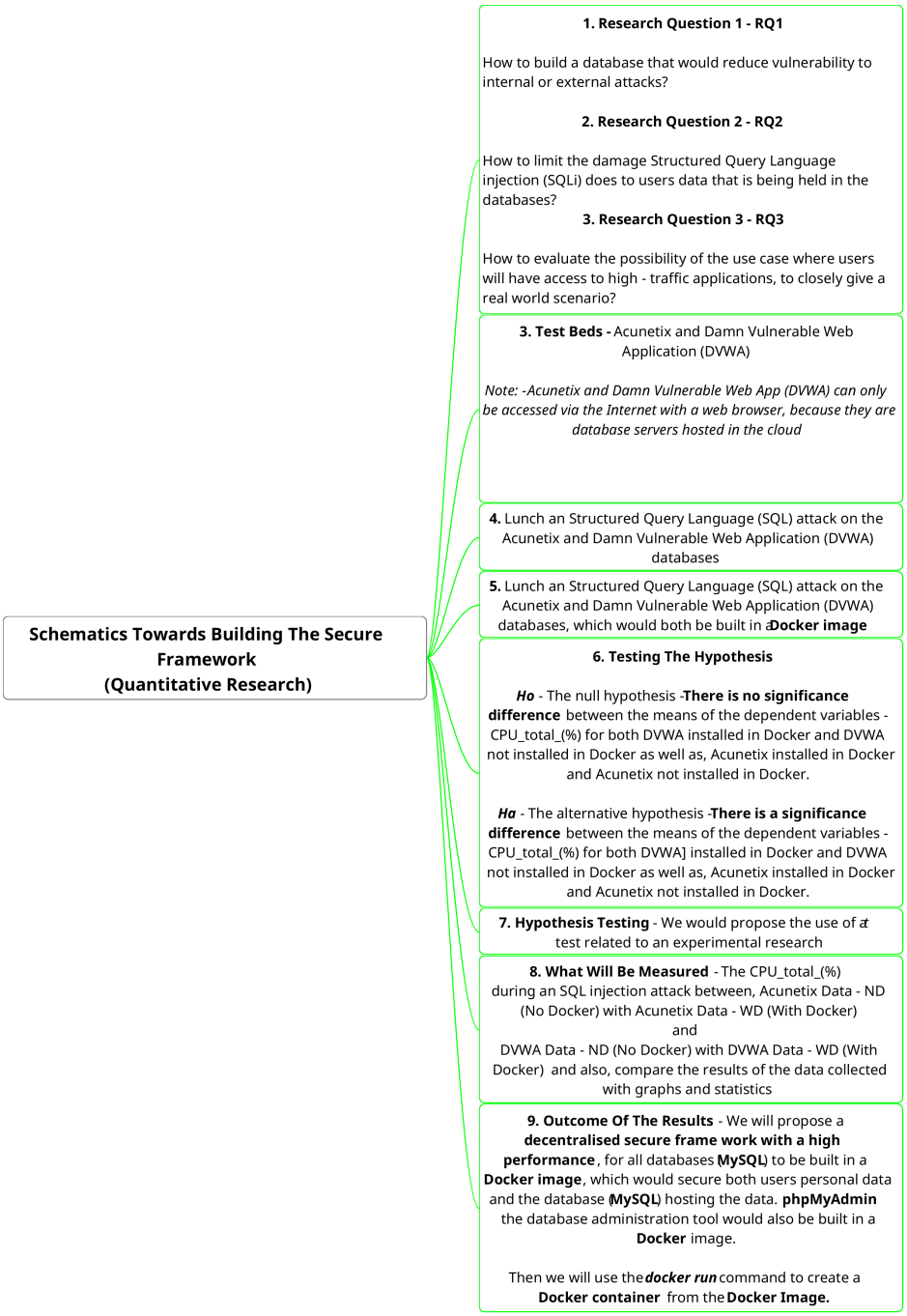


Figure 3. Schematics towards building the secure framework (quantitative research).

3.4. DVWA—Database Without Docker Installed

All implementations were conducted in VirtualBox.

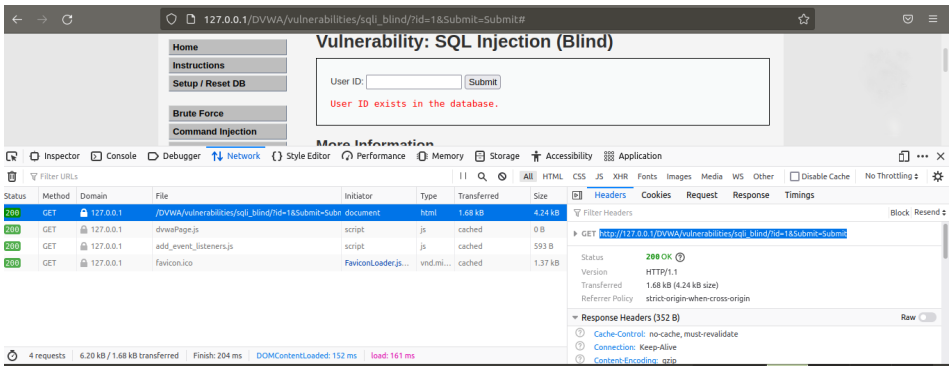


Figure 4. Gathering information from Damn Vulnerable Web Application.

Figure 5. Launching an SQL injection attack.

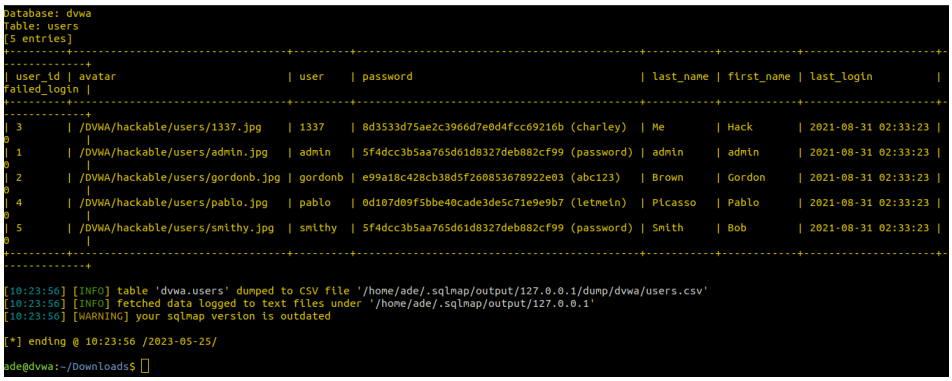


Figure 6. SQL injection attack was successful.

3.5. DVWA—Database with Docker Installed

All implementations were conducted in VirtualBox.

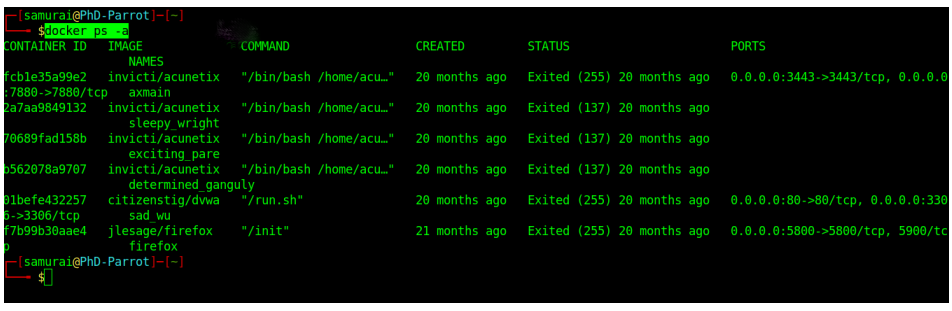


Figure 7. Running DVWA in a Docker container.

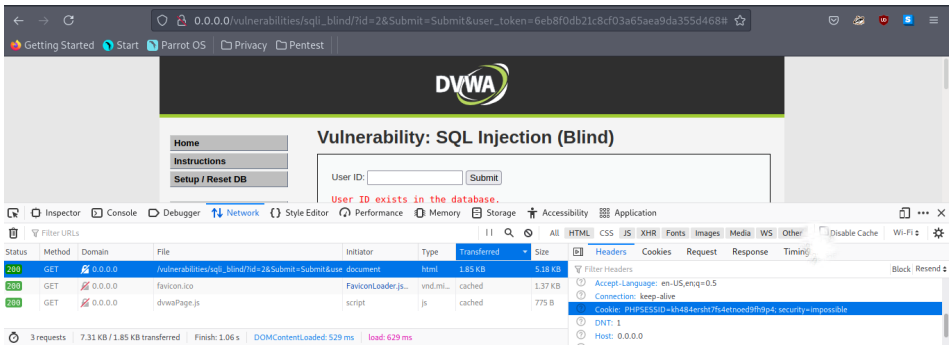


Figure 8. Gathering information to launch an SQL injection attack.

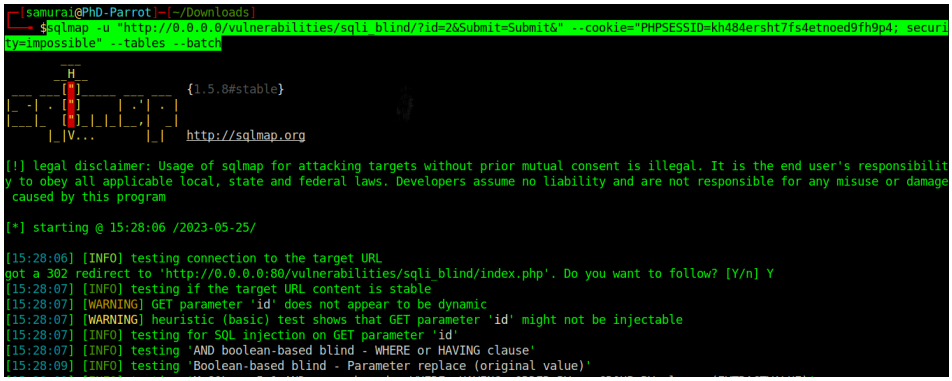


Figure 9. Launching an SQL injection attack.

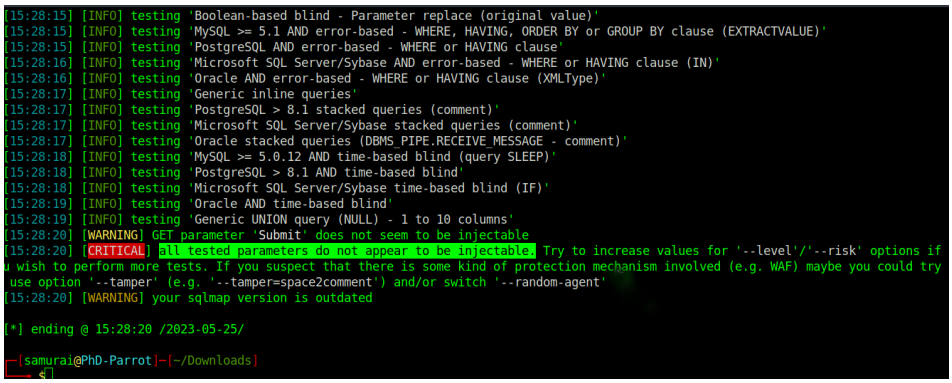


Figure 10. SQL injection attack was not successful.

3.6. Ethics Approval and GDPR

The ethics approval process and GDPR (General Data Protection Regulation) evaluation regarding this research followed the regulations of the UK (United Kingdom) GDPR [40] and the General Data Protection Regulation of the European Union's new data protection law [41]. Because this study does not involve human or animal participants, the research proposed follows the rules set by (https://www.glos.ac.uk/information/knowledge-base/research-ethics-a-handbook-of-principles-and-procedures/ (accessed on 19 December 2024) Research Ethics: A Handbook of Principles and Procedures), applying the guidelines to the research methods process meeting all the regulations required [42]. Lastly, Figure 2 was built in a VirtualBox [39] containment so as ensure that in the event of a mishap during the implementation, the mishap is contained in the VirtualBox [39] and does not leak into the host computer.

3.7. Environment

This research uses two key technologies to facilitate the experimental setup: Docker and VirtualBox as discussed below.

Docker is a free, open-source platform for software containerisation. Docker allows package applications to be packaged with all their dependencies into standardised units called containers. These containers are lightweight and portable, making them ideal for replicating and deploying in our experimental environment consistently across different systems. Docker enjoys widespread adoption, often coming pre-installed in popular Linux distributions. Notably, several studies have successfully utilised Docker images and containers for application development and deployment (see Table 4).

Table 4. Other research papers that have employed Docker.

Software	What the Software Was Used for
Docker containers [43]	Analysing of data.
Docker containers [22]	Run the Docker containers from different databases.
Docker Swarm [44]	Build a virtual system used for simulation.
Docker containers [45]	Capturing the behavior of a container's life cycle.
Docker swarm [46]	Used as a testbed for Distributed Denial-of-Service (DDoS) attacks.
Docker-compose [47]	Docker-compose.yml files and Dockerfiles.
Docker images [48]	MDSplus—a set of software tools.
Docker swarm [49]	Checking the security of misbehaving manager nodes.
Docker swarm [50]	Multiple clouds consisting of distributed systems.
Docker containers [51]	Evolution and maintenance of Docker containers.
Docker [52]	To help Q&A forum users.
Docker [53]	Attacks against hCaptcha Systems.

VirtualBox is the framework in which all the images used in this research paper would be built and comes installed by default in most Linux distributions. VirtualBox is a powerful virtualisation software solution developed and maintained by Oracle. VirtualBox allows the creation and management of virtual machines, essentially emulating entire computer systems within the host machine. While Docker containers share the host operating system kernel, virtual machines create isolated environments with their own guest operating systems. VirtualBox provides a robust platform for building the foundation upon which our Docker images will operate. Several researchers have employed VirtualBox in their studies (see Table 5).

Table 5. Other research papers that have employed VirtualBox.

Software	What the Software Was Used for
VirtualBox [54]	VirtualBox and host performance comparison.
VirtualBox virtualiser [55]	Educational purposes (laboratory).
Virtualisation laboratory [56]	Computer networking with undergraduates.
VirtualBox [57]	VMware and VirtualBox comparison.
VirtualBox [58]	Linux applications running in Windows.

3.8. Databases

In this research, we used two databases for experimental research on SQLi attacks as follows:

Damn Vulnerable Web Application (DVWA) is a free, open-source web application intentionally designed to be insecure. This controlled environment allows security professionals to test their penetration testing tools in a safe, legal manner. Similarly, researchers can utilise DVWA to obtain practical experience with web application security. DVWA has been used by several researchers in this field [59–61].

Acunetix is a commercial web vulnerability scanner designed to identify security weaknesses in web applications. It offers a comprehensive suite of features for automated scanning, allowing security professionals to efficiently detect potential vulnerabilities that could be exploited by attackers. Several research studies have employed Acunetix to evaluate its effectiveness in vulnerability detection, including the following research papers: [62,63].

To work with the databases, we used (https://www.mysql.com/downloads/ (accessed on 19 December 2024) MySQL), a free and open-source Relational Database Management System (RDBMS). (https://www.phpmyadmin.net/ (accessed on 19 December 2024) phpMyAdmin) is also used to work with MySQL. We used Linux OS as our platform to access databases.

3.9. Network Analysis

Wireshark is a free and open-source network protocol analyser used to monitor incoming network traffic on the target Ubuntu 20.04 system. Wireshark is a popular tool among network security professionals worldwide, offering deep insights into network communication by capturing and analysing network packets. By capturing the traffic on the victim's computer, Wireshark enables us to observe the attacker's interactions and potentially identify vulnerabilities exploited during the attack. The research papers by Sandhya et al. [64] and Das and Tuna [65] show how researchers have used Wireshark in their network security experiments. Wireshark is used to monitor the incoming network traffic on the victims computer running (https://releases.ubuntu.com/focal/ (accessed on 19 December 2024) Ubuntu 20.04.6 LTS (Focal Fossa)).

Nmap is a powerful open-source network scanner and security auditing tool commonly used by system administrators and security professionals. Operating as a command-line tool on Linux systems, Nmap offers a comprehensive suite of features for network exploration and vulnerability identification. This research will utilise Nmap to perform security scans on the target system, potentially revealing open ports, services running on those ports, and potential security weaknesses. The research conducted by [66–70] shows the diverse applications of Nmap in security research. This tool would be used on the virtual machine running the penetrating operating system (https://parrotsec.org/ (accessed on 19 December 2024) Parrot OS) to see if it can capture the victim's IP address.

3.10. Data Collection

This research utilises Glances (version 3.9 or higher) [71], a system monitoring tool written in Python, to gather crucial performance metrics throughout the experiment. Glances provides functionalities for monitoring system performance through a web interface, remote access via the Linux terminal, and a client/server mode for broader data collection. During the experiment, Glances is used to capture real-time measurements of key performance indicators (KPIs), including CPU utilisation, data throughput, and time. Glances has been successfully used for data collection in previous research efforts, as demonstrated by the research papers by Kok et al. [72] and Manore et al. [73].

This research uses JASP 0.18 (Jeffrey's Amazing Statistics Program) [74], which is an open-source statistical software package developed with support from the University of Amsterdam. JASP provides a user-friendly interface and robust statistical analysis capabilities, making it a valuable tool for researchers like ourselves. Additionally, its integration with the Open Science Framework (OSF) facilitates data sharing and transparency, aligning with our commitment to open science practices. JASP's growing popularity within the scientific community is evident in its utilisation by various recent studies (e.g., Houminer-Klepar et al. [75]). We also used https://www.lock5stat.com/StatKey/index.html StatKey (accessed on 19 December 2024) as a tool to bootstrap the difference in means.

4. Results

This section presents the statistical analysis conducted to evaluate the impact of Docker on CPU utilisation during SQL injection attacks against DVWA and Acunetix web applications. The type of sampling used to collect the data was one of the non-probability sampling methods called convenience sampling. The results are readily available and easy to collect. Additionally, in comparison to other sampling techniques, convenience sampling can help overcome most limitations associated with research. For example, if old people always assemble in a park, the researcher can then go to the park and collect the samples—the park acts as a convenient place to collect samples [76]. Furthermore, convenience sampling is inexpensive (cheap), which can be seen as an advantage. Convenience sampling is easy to execute and efficient, but in this research, limitations include a small sample size drawn from a large sample pool, which limits generalisability [77]. In the case of the Damn Vulnerable Web App (DVWA) database results, the first 20 (number of values) data items were collected, while, for the Acunetix database results, the first 15 (number of values) data items were collected.

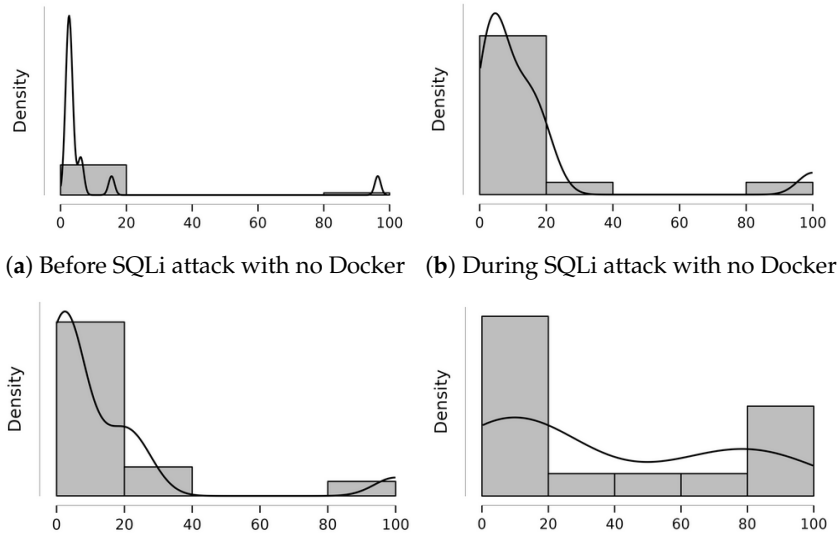
- **Paired Samples *t*-Test:** A paired-samples t-test was employed for each dataset (DVWA and Acunetix) to compare the means of CPU utilisation (CPU_total_(%)) between scenarios with and without Docker protection (ND—No Docker, WD—With Docker). This test is appropriate as we are analysing data from the same set of systems measured under two different conditions (attack with and without Docker). Additionally, a two-tailed test was chosen as we are not pre-determining the direction of the difference (improvement or degradation) in CPU utilisation.
- **Hypothesis Testing:** The null hypothesis (H_0) states that there is no significant difference in the mean CPU utilisation between the No Docker (ND) and With Docker (WD) scenarios for both DVWA and Acunetix. The alternative hypothesis (H_a) proposes that there is a significant difference.
- **Significance Level and Critical Value:** The level of significance (α) was set at 0.05, indicating a 5% chance of rejecting the null hypothesis when it is actually true. Based on the degrees of freedom (df = n − 1, where n is the number of samples in each group), the critical values for the two-tailed test were determined from a t-distribution table.

Table 6 presents the descriptive statistics for the experiments conducted on Acunetix and DVWA, both with and without Docker protection. A total of 15 experiments were performed on Acunetix, and 20 experiments were conducted on DVWA. Each experiment was repeated twice: once with Docker and once without. CPU utilisation was measured both before the SQL injection attack (Before) and during the attack (During). While the maximum CPU usage reached 100% in all cases, the mean and median CPU utilisation values were significantly higher during the attack when Docker was used. This increased CPU usage was further evidenced by the higher standard deviation observed in the "During" phase with Docker, indicating greater variability in CPU utilisation during these periods.

Figures 11 and 12 present histograms illustrating the distribution of CPU utilisation for all eight experimental scenarios: before and during SQL injection attacks on Acunetix and DVWA, both with and without Docker protection. A visual inspection of these histograms reveals a notable trend: the CPU utilisation exceeds 50% more frequently in scenarios involving Docker compared to those without Docker. This suggests that Docker, while enhancing security, may also incur a performance overhead in certain scenarios. This is considered further in the next paragraph.

Table 6. Descriptive statistics.

Database	**Acunetix**				**DVWA**			
No. Cases	**15**				**20**			
Container	**No Docker**		**With Docker**		**No Docker**		**With Docker**	
Scenario	Before	During	Before	During	Before	During	Before	During
Median	2.9	7	2.9	17.5	5.55	11.25	5.4	37.5
Mean	10.207	14.753	13.713	36.9	10.56	18.735	11.46	49.745
Std. Deviation	24.091	24.378	25.332	36.065	21.927	21.035	21.557	38.633
Skewness	3.746	3.462	3.191	0.519	4.285	3.358	4.031	0.187
Std. Error of Skewness	0.58	0.58	0.58	0.58	0.512	0.512	0.512	0.512
Kurtosis	14.251	12.702	11.075	−1.593	38.818	12.664	17.044	−1.847
Std. Error of Kurtosis	1.121	1.121	1.121	1.121	0.992	0.992	0.992	0.992
Minimum	1.8	1.8	1.4	1.4	1.4	5.1	2.2	3.9
Maximum	96.4	100	100	96.6	100	100	100	100



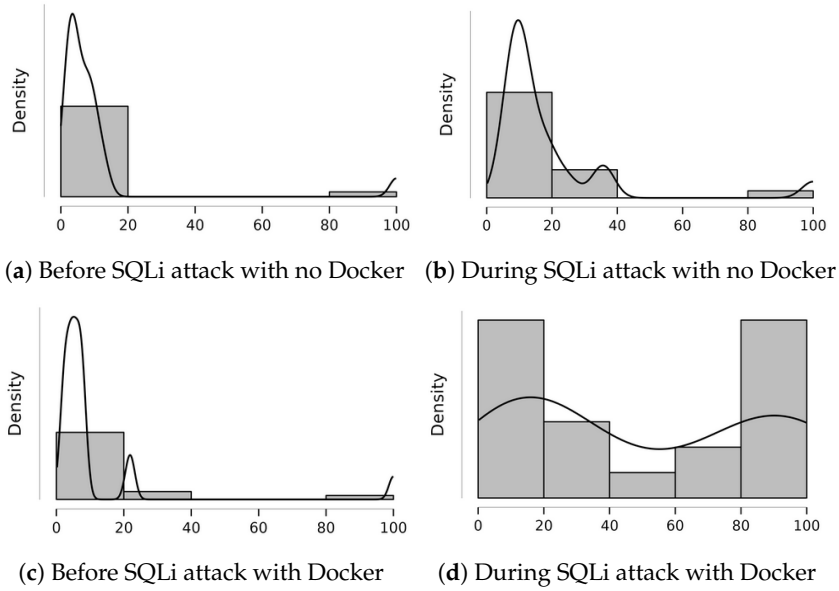
(**a**) Before SQLi attack with no Docker (**b**) During SQLi attack with no Docker

(**c**) Before SQLi attack with Docker (**d**) During SQLi attack with Docker

Figure 11. Distribution histogram of CPU usage in attacking Acunetix.

As shown in Table 7, for the DVWA data, the calculated t-statistic (t = 3.307) exceeded the critical value (CV = ±2.093). This statistically significant result ($p < 0.05$) rejects the null hypothesis, indicating a significant difference in CPU utilisation between the No Docker and With Docker scenarios when attacking DVWA. Similarly (see Table 8), for the Acunetix data, the t-statistic (t = 2.339) exceeded the critical value (CV = ±2.145), leading to rejection of the null hypothesis ($p < 0.05$). This suggests a statistically significant difference in CPU

utilisation between the two scenarios for Acunetix as well. As such, the statistical analysis confirms that Docker plays a role in influencing CPU utilisation during SQL injection attacks (see Table 9).



(**a**) Before SQLi attack with no Docker (**b**) During SQLi attack with no Docker

(**c**) Before SQLi attack with Docker (**d**) During SQLi attack with Docker

Figure 12. Distribution histogram of CPU usage in attacking DVWA.

Table 7. Legend for paired-samples *t*-test results for DVWA.

Formula	**-**	**Interpretation**
CV(19) = ±2.093	-	$t(19) = 3.307$ exceeds the CV
$df = n - 1$	-	n = 19
$p < 0.05$	-	$p = 0.004$ is less than $p < 0.05$
95%CI [11.4, 50.6]	-	Does not contain 0
Cohen's *d*	-	0.74 (Effect—between moderate and large)
95%CI	-	Confidence Interval
CV(19)	-	Critical Value
df	-	Degrees of freedom
n = 20	-	Number of values
p	-	The *p*-value
t	-	The paired-samples *t*-test

In terms of the effectiveness of an SQLi attack, both DVWA and Acunetix were successfully compromised when Docker was not installed. This indicates that these applications, in their default state, are susceptible to SQL injection attacks (as expected). When Docker was installed on the systems hosting DVWA and Acunetix, the SQL injection attacks were unsuccessful. This suggests that Docker, in this context, effectively mitigated the risks associated with SQL injection. By isolating the application within a container, Docker can help prevent attackers from exploiting vulnerabilities and compromising the underlying system. This is due to Docker's ability to restrict access to resources and isolate the application from the host environment. Table 9 presents the readings from the paired samples *t*-test results for DVWA and the readings from the paired samples *t*-test results for Acunetix. Additionally, in Table 9, the sampling used was non-probability (planning) convenience

sampling, which reflects the easy availability of the samples that were chosen from all the samples available.

This is why the Critical Value (CV) used is CV(19) for DVWA and CV(14) for Acunetix.

Table 8. Legend for paired-samples *t*-test results for Acunetix.

Formula	**-**	**Interpretation**
CV(14) = ±2.145	-	$t(14) = 2.339$ exceeds the CV
$df = n - 1$	-	n = 14
$p < 0.05$	-	$p = 0.04$ is less than $p < 0.05$
95%CI [1.84, 42.5]	-	Does not contain 0
Cohen's *d*	-	0.60 (Effect—between moderate and large)
95%CI	-	Confidence Interval
CV(14)	-	Critical Value
df	-	Degrees of freedom
n = 15	-	Number of values
p	-	The *p*-value
t	-	The paired-samples *t*-test

Table 9. Readings from the paired-samples *t*-test results.

Database	***t*()**	***p***	**95%CI**	***d***
DVWA	$t(19) = 3.307$	0.004	[11.4, 50.6]	0.74
Acunetix	$t(14) = 2.339$	0.04	[1.84, 42.5]	0.60

Additionally, we used bootstrapping to compare the means between CPU usage in different situations to further examine the hypothesis. Figures 13–20 (Bell curves) prove that there are a significance difference with DVWA - (before an SQLi attack without Docker, During an SQLi attack without Docker), (before an SQLi attack with Docker and During an SQLi attack with Docker). There is also a significance difference with Acunetix - (before an SQLi attack without Docker, During an SQLi attack without Docker), (before an SQLi attack with Docker and During an SQLi attack with Docker). Additionally, there is also a significance difference (without Docker installed: before an SQLI attack with both databases combined, During an SQLI attack both databases combined. Furthermore, there is also a significance difference (with Docker installed: before an SQLI attack with both databases combined, During an SQLI attack both databases combined (see Table 10).

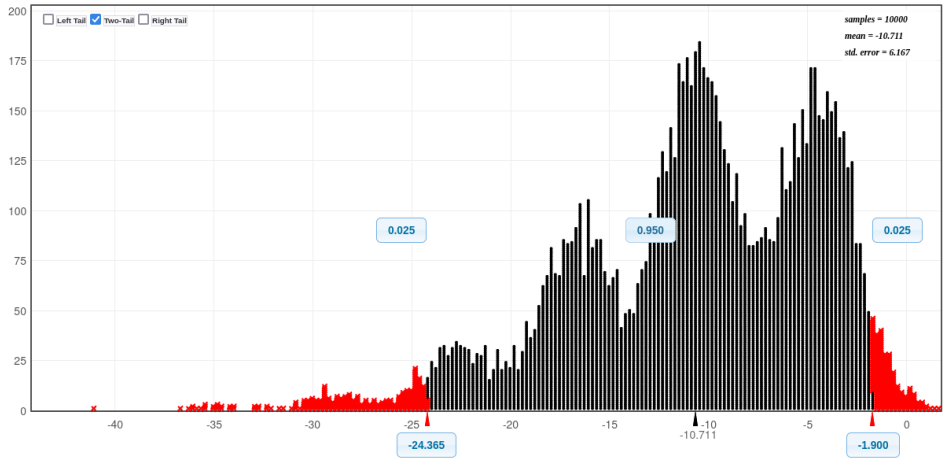


Figure 13. Acunetix no Docker installed.

Table 10. StatKey legend.

+ +	Significant Difference
− −	Significant Difference
+ −	No Significant Difference
− +	No Significant Difference

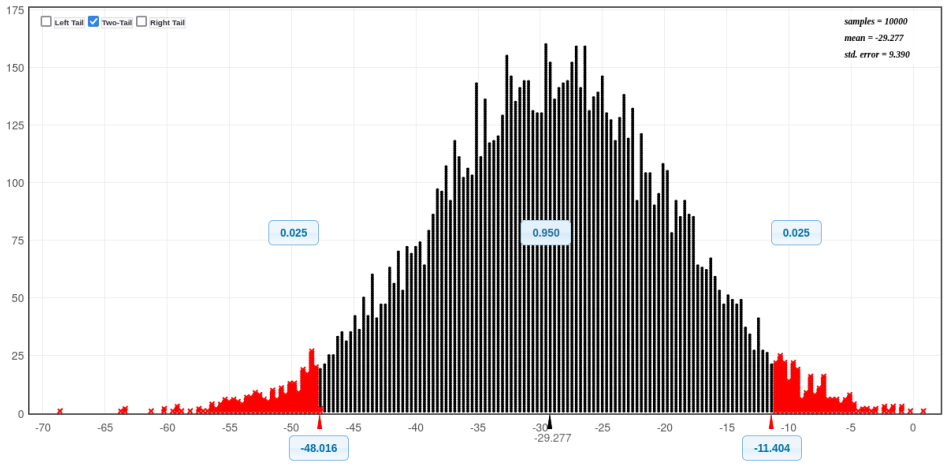


Figure 14. Acunetix with Docker installed.

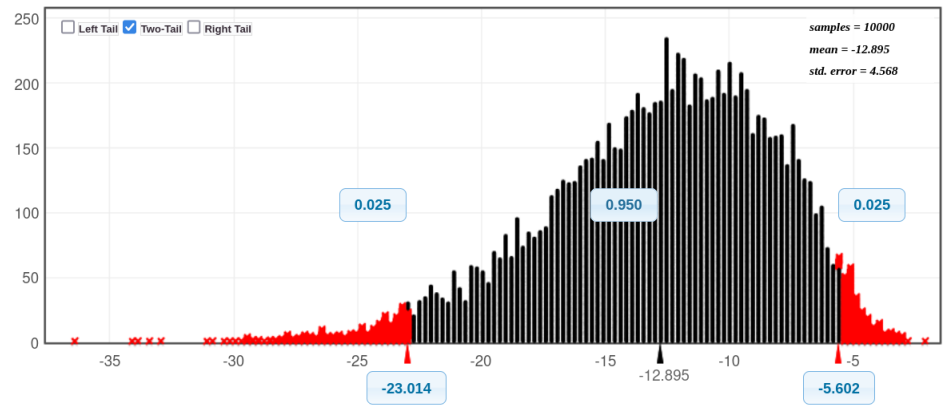


Figure 15. DVWA no Docker installed.

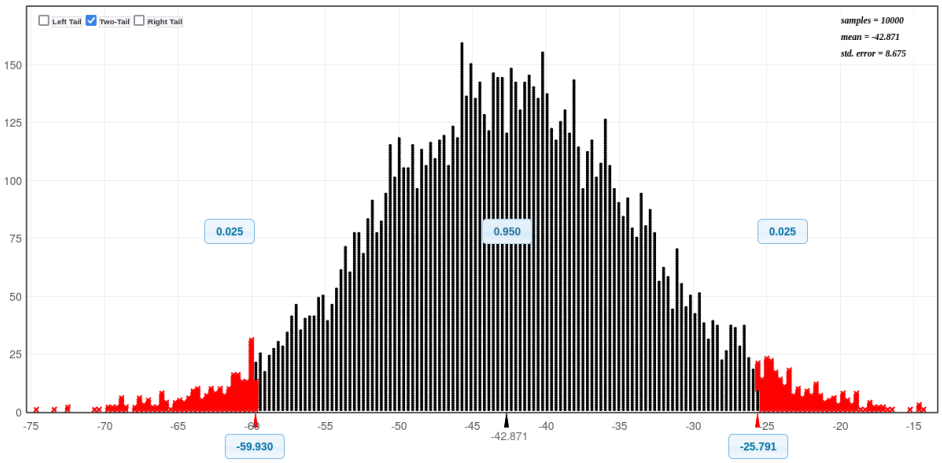


Figure 16. DVWA with Docker installed.

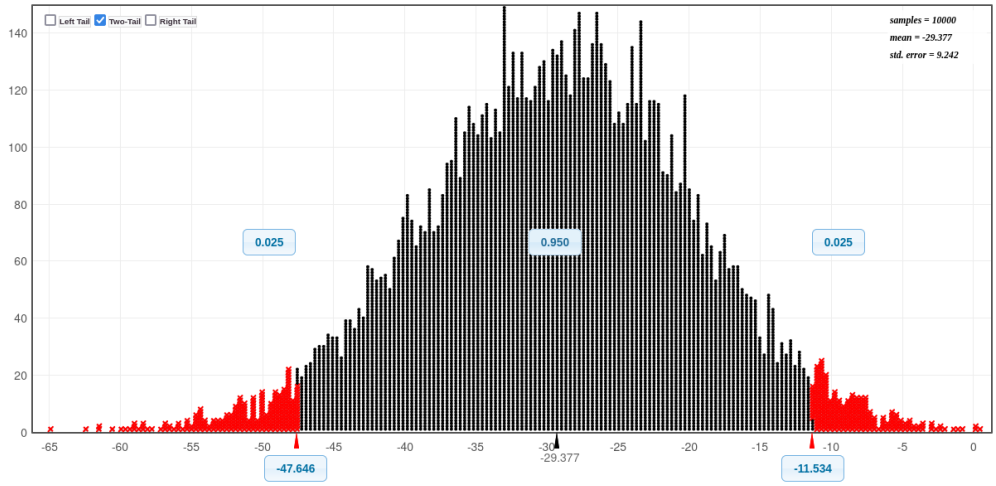


Figure 17. Acunetix with Docker before an SQLi attack.

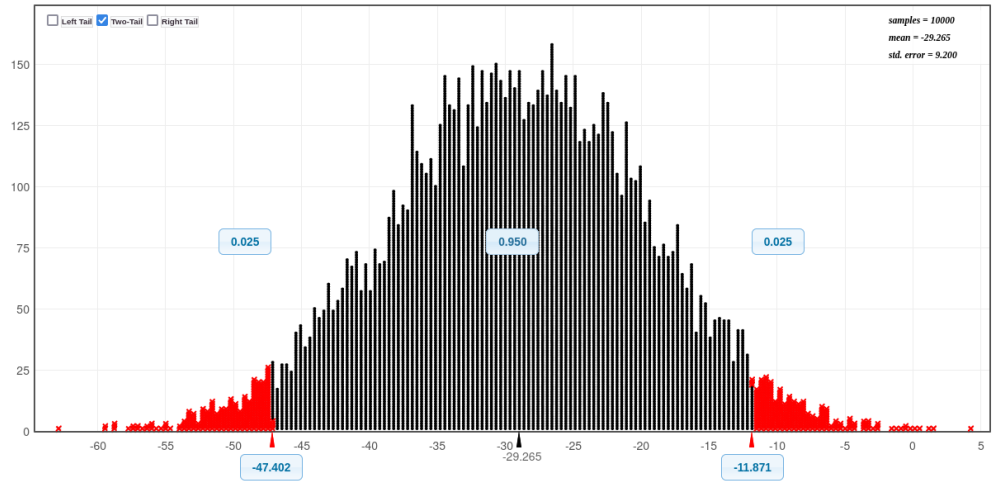


Figure 18. Acunetix with Docker during an SQLi attack.

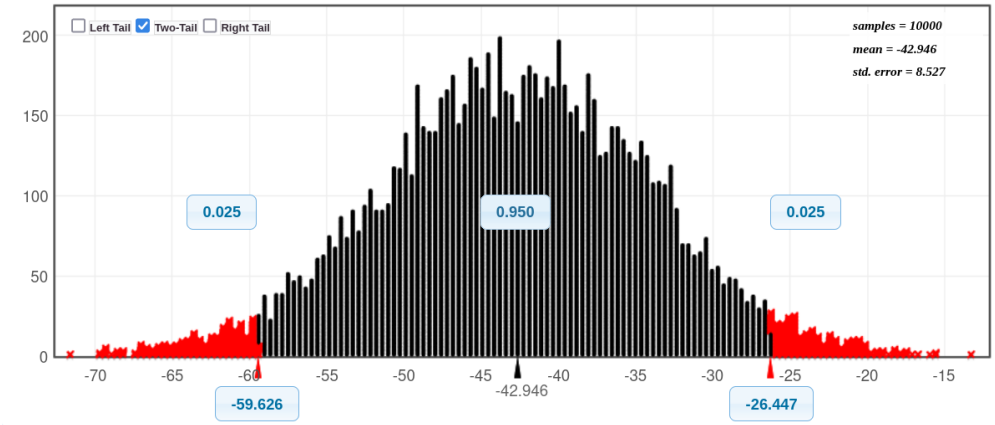


Figure 19. DVWA with Docker before an SQLi attack.

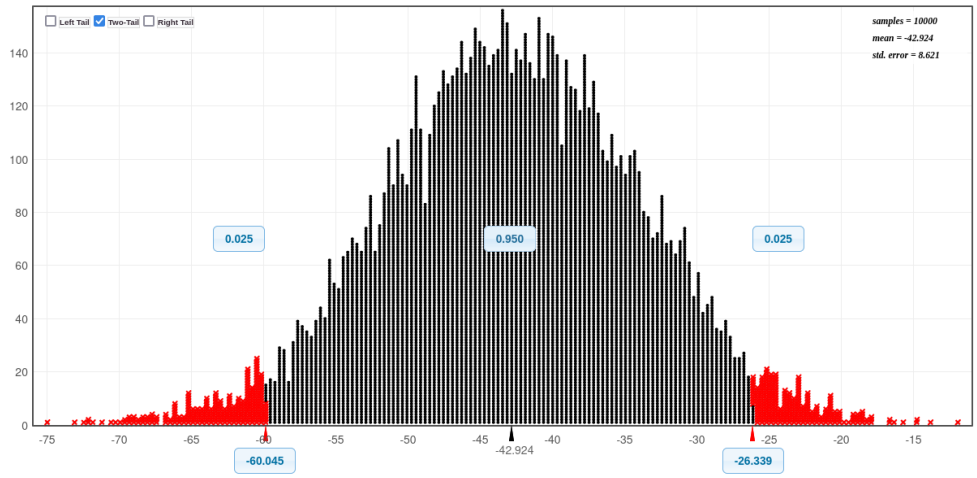


Figure 20. DVWA with Docker during an SQLi attack.

5. Summary of Findings

This research aimed to design and evaluate a database system that prioritises performance, security, and user accessibility, particularly in the face of SQL injection attacks and high-traffic demands. While numerous studies have explored the use of Docker as a framework for various applications, there remains a significant gap in understanding how Docker can be used to enhance database security.

The findings of this study demonstrate the efficacy of a decentralised database approach built upon the Docker framework. By isolating database instances within Docker containers, the risk of external attacks, including SQL injection, is significantly reduced, addressing Research Question 1. Additionally, such an environment provides a robust defence against SQL injection attempts, safeguarding user data and database integrity, thus addressing Research Question 2. Furthermore, the Docker-based database showed high CPU utilisation during attack, which needs to be considered when used in high-traffic applications, fulfilling Research Question 3. A key advantage of this approach is that it does not necessitate modifications to the traditional database infrastructure. By introducing Docker as a layer of abstraction, the existing database remains unaffected. The Docker framework offers improved security. Although, this research was designed from the user's perspective as a decentralised database without altering the existing centralised database, developers or administrators of a centralised database should look at the practical benefits of being able to run Docker images with any assigned IP (Internet Protocol) address of their choice; this improves the security of a database running in a Docker container. Meanwhile, the authors of the research paper Gore et al. [43] obtained a positive result

when utilising Docker containers to handle data in a network environment compared to the results of the research paper by Velasquez et al. [22] that showed Microsoft Azure, Amazon Web Services, OpenStack, IBM, VMware and Google Compute Engine, which are all cloud providers, are now being supported by Docker. Additionally, the research paper by Reis et al. [47] reported that developers are not keen on using different types of tools to build their Docker-compose.yml files and Dockerfiles. Lastly, as presented in Table 3, under graphics Intel/Nvidia 2GBRam, the more memory (Ram) a video card (Nvidia (GPU) Graphics Processing Unit) has, the more this would help in reducing the high-traffic bottle necks encountered where the CPU (Central Processing Unit) is the only processor processing information within a database. A limitation of this research was the use of convenience sampling. Positive ideas for future work/research include measuring the computer memory and investigating the potential for more data output.

6. Conclusions

This research evaluated the performance and security aspects of traditional and Docker-based database systems. The primary objective was to mitigate the risks associated with SQL injection attacks and ensure reliable access to high-traffic applications. By adopting a user-centric approach to database design, we demonstrated that a Docker-based database can effectively limit the impact of SQL injection attacks on user data. Additionally, the decentralised nature of this approach empowers users with greater control over their data, improving accessibility and reducing the risk of unauthorised access. The findings of this research suggest that a Docker-based database can provide a robust and secure solution for various applications. By isolating database instances within containers, Docker can significantly reduce vulnerability to external attacks, including by SQL injection.

Author Contributions: Conceptualization, A.D.A.; Methodology, A.D.A.; Validation, A.D.A.; Formal analysis, A.D.A.; Writing – original draft, A.D.A., H.C. and A.A.; Writing – review & editing, A.D.A. and H.C.; Visualization, A.D.A.; Supervision, H.C. and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhao, X.; Lin, Q.; Chen, J.; Wang, X.; Yu, J.; Ming, Z. Optimizing security and quality of service in a Real-time database system using Multi-objective genetic algorithm. *Expert Syst. Appl.* **2016**, *64*, 11–23. [CrossRef]
2. Wagner, J.; Rasin, A.; Glavic, B.; Heart, K.; Furst, J.; Bressan, L.; Grier, J. Carving database storage to detect and trace security breaches. *Digit. Investig.* **2017**, *22*, S127–S136. [CrossRef]
3. Kaneko, I.; Yuda, E.; Okada, H. Docker Vectorization, a Cloud-Native Privacy Agent—The Analysis of Demand and Feasibility for Era of Developing Complexity of Privacy Management. *Appl. Sci.* **2023**, *13*, 3235. [CrossRef]
4. Tao, X.; Liu, Y.; Zhao, F.; Yang, C.; Wang, Y. Graph database-based network security situation awareness data storage method. *Eurasip J. Wirel. Commun. Netw.* **2018**, *2018*, 294. [CrossRef]
5. Toapanta, S.M.T.; Gallegos, L.E.M.; Trejo, J.A.O. Security analysis of civil registry database of Ecuador. In Proceedings of the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 3–5 March 2016 ; pp. 1024–1029. [CrossRef]
6. Grubbs, P.; Ristenpart, T.; Shmatikov, V. Why Your Encrypted Database Is Not Secure. In Proceedings of the 16th Workshop on Hot Topics in Operating Systems. Association for Computing Machinery, HotOS '17, Whistler, BC, Canada, 7–10 May 2017; pp. 162–168. [CrossRef]
7. Alves, P.G.M.R.; Aranha, D.F. A framework for searching encrypted databases. *J. Internet Serv. Appl.* **2018**, *9*, 1. [CrossRef]

8. Santos, N.; Younis, W.; Ghita, B.; Masala, G. Enhancing Medical Data Security on Public Cloud. In Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience (CSR), Rhodes, Greece, 26–28 July 2021; pp. 103–108. [CrossRef]
9. Said, W.; Mostafa, A.M. Towards a Hybrid Immune Algorithm Based on Danger Theory for Database Security. *IEEE Access* **2020**, *8*, 145332–145362. [CrossRef]
10. Neto, N.N.; Madnick, S.; Paula, A.M.G.D.; Borges, N.M. Developing a Global Data Breach Database and the Challenges Encountered. *J. Data Inf. Qual.* **2021**, *13*, 1–33. [CrossRef]
11. Algarni, A.M.; Thayananthan, V.; Malaiya, Y.K. Quantitative Assessment of Cybersecurity Risks for Mitigating Data Breaches in Business Systems. *Appl. Sci.* **2021**, *11*, 3678. [CrossRef]
12. Mahrouqi, A.; Tobin, P.; Abdalla, S.; Kechadi, T. Simulating SQL-Injection Cyber-Attacks Using GNS3. *Int. J. Comput. Theory Eng.* **2016**, *8*, 213–217. [CrossRef]
13. Williams, M.A.; Dey, S.; Barranco, R.C.; Naim, S.M.; Hossain, M.S.; Akbar, M. Analyzing Evolving Trends of Vulnerabilities in National Vulnerability Database. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 3011–3020. [CrossRef]
14. Gonzalez, C.; Jung, G. Database SQL Injection Security Problem Handling with Examples. In Proceedings of the 2019 6th International Conference on Computational Science and Computational Intelligence (CSCI 2019), New York, NY, USA, 5–7 December 2019; pp. 145–149. [CrossRef]
15. Odirichukwu, J.C.; Asagba, P.O. Security concept in web database development and administration—A review perspective. In Proceedings of the 2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON), Owerri, Nigeria, 7–10 November 2017; pp. 383–391, ISSN 2377-2697. [CrossRef]
16. Wang, K.C.; Reiter, M.K. Using Amnesia to Detect Credential Database Breaches. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Vancouver, BC, Canada, 11–13 August 2021; p. 18.
17. Hassanzadeh, Z.; Biddle, R.; Marsen, S. User Perception of Data Breaches. *IEEE Trans. Prof. Commun.* **2021**, *64*, 374–389. [CrossRef]
18. Goel, K.; Hofstede, A.H.M.T. Privacy-Breaching Patterns in NoSQL Databases. *IEEE Access* **2021**, *9*, 35229–35239. [CrossRef]
19. Crooks, N. A Client-centric Approach to Transactional Datastores. In Proceedings of the 2021 International Conference on Management of ACM, Virtual, 20–25 June 2021; pp. 3–5. [CrossRef]
20. Perri, D.; Simonetti, M.; Gervasi, O. Deploying Efficiently Modern Applications on Cloud. *Electronics* **2022**, *11*, 450. [CrossRef]
21. Saha, P.; Govindaraju, M.; Marru, S.; Pierce, M. Integrating Apache Airavata with Docker, Marathon, and Mesos. *Concurrency Computat. Pract. Exper.* **2016**, *28*, 1952–1959. [CrossRef]
22. Velasquez, W.; Munoz-Arcentales, A.; Salvachua Rodriguez, J. A Case Study: Ingestion Analysis of WSN Data in Databases using Docker. In Proceedings of the 2018 1st International Conference on Computer Applications & Information Security (ICCAIS' 2018), Riyadh, Saudi Arabia, 4–6 April 2018.
23. Mentz, L.L.; Loch, W.J.; Koslovski, G.P. Comparative experimental analysis of Docker container networking drivers. In Proceedings of the 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), Piscataway, NJ, USA, 9–11 November 2020; pp. 1–7. [CrossRef]
24. Zou, Z.; Xie, Y.; Huang, K.; Xu, G.; Feng, D.; Long, D. A Docker Container Anomaly Monitoring System Based on Optimized Isolation Forest. *IEEE Trans. Cloud Comput.* **2022**, *10*, 134–145. [CrossRef]
25. Bedhief, I.; Kassar, M.; Aguili, T. Empowering SDN-Docker Based Architecture for Internet of Things Heterogeneity. *J. Netw. Syst. Manag.* **2022**, *31*, 14. [CrossRef]
26. Leahy, D.; Thorpe, C. Zero Trust Container Architecture (ZTCA): A Framework for Applying Zero Trust Principals to Docker Containers. *Int. Conf. Cyber Warf. Secur.* **2022**, *17*, 111–120. [CrossRef]
27. Aleksandrovs-Moisejs, D.; Ipatovs, A.; Grabs, E.; Rjazanovs, D. Evaluation of a Long-Distance IEEE 802.11ah Wireless Technology in Linux Using Docker Containers. *Elektron. Elektrotechnika* **2022**, *28*, 71–77. [CrossRef]
28. Singh, N.; Hamid, Y.; Juneja, S.; Srivastava, G.; Dhiman, G.; Gadekallu, T.R.; Shah, M.A. Load balancing and service discovery using Docker Swarm for microservice based big data applications. *J. Cloud Comput.* **2023**, *12*, 4. [CrossRef]
29. Ramanathan, S.; Bhattacharyya, A.; Kondepu, K.; Fumagalli, A. Enabling containerized Central Unit live migration in 5G radio access network: An experimental study. *J. Netw. Comput. Appl.* **2024**, *221*, 103767. [CrossRef]
30. Yohannis, A.; Kolovos, D.; García-Domínguez, A. Exploring complex models with picto web. *Sci. Comput. Program.* **2024**, *232*, 103037. [CrossRef]
31. Moysiadis, V.; Tsakos, K.; Sarigiannidis, P.; Petrakis, E.G.M.; Boursianis, A.D.; Goudos, S.K. A Cloud Computing web-based application for Smart Farming based on microservices architecture. In Proceedings of the 2022 11th International Conference on Modern Circuits and Systems Technologies (MOCAST), Bremen, Germany, 8–10 June 2022; pp. 1–5. [CrossRef]
32. Alyas, T.; Tabassum, N.; Iqbal, M.w.; Alshahrani, A.; Alghamdi, A.; Shahzad, S.K.; Waseem, M. Resource Based Automatic Calibration System (RBACS) Using Kubernetes Framework. *Intell. Autom. Soft Comput.* **2022**, *35*, 1165–1179. [CrossRef]

33. Prigg, M. Amazon's Cloud Service Partial Outage Affects Certain Websites. 2017. Section: Science. Available online: https://www.reuters.com/article/technology/disruption-in-amazon-s-cloud-service-ripples-through-internet-idUSKBN1672E1/ (accessed on 19 December 2024).
34. Hersyah, M.H.; Hossain, M.D.; Taenaka, Y.; Kadobayashi, Y. A Risk Assessment Study: Encircling Docker Container Assets on IaaS Cloud Computing Topology. In Proceedings of the 2023 6th Conference on Cloud and Internet of Things (CIoT), Lisbon, Portugal, 20–22 March 2023; pp. 225–230, ISSN 2159-6972. [CrossRef]
35. Qian, J.; Wang, Y.; Wang, X.; Zhang, P.; Wang, X. Load balancing scheduling mechanism for OpenStack and Docker integration. *J. Cloud Comput.* **2023**, *12*, 67. [CrossRef]
36. da Silva, L.F.; Lima, J.V.F. An evaluation of relational and NoSQL distributed databases on a low-power cluster. *J. Supercomput.* **2023**, *79*, 13402–13420. [CrossRef]
37. Shabbir, M.; Shabbir, A.; Iwendi, C.; Javed, A.R.; Rizwan, M.; Herencsar, N.; Lin, J.C.W. Enhancing Security of Health Information Using Modular Encryption Standard in Mobile Cloud Computing. *IEEE Access* **2021**, *9*, 8820–8834. [CrossRef]
38. Udacity. Deduction, Induction, Abduction—Georgia Tech—KBAI: Part 5. 2015. Available online: https://www.youtube.com/watch?v=-nn3XMoPC7s (accessed on 19 December 2024).
39. Oracle VirtualBox. 2025. Available online: https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html (accessed on 19 December 2024).
40. ICO. *UK GDPR Guidance and Resources*; ICO: Wilmslow, UK, 2025.
41. What is GDPR, the EU's New Data Protection Law? 2018. Section: GDPR Overview. Available online: https://gdpr.eu/what-is-gdpr/ (accessed on 19 December 2024).
42. University of Gloucestershire. *Research Ethics: A Handbook of Principles and Procedures;* University of Gloucestershire: Cheltenham, UK, 2022. Available online: https://www.glos.ac.uk/information/knowledge-base/research-ethics-a-handbook-of-principles-and-procedures/ (accessed on 19 December 2024).
43. Gore, R.; Banerjea, S.; Tyagi, N.; Saurav, S.; Acharya, D.; Verma, V. An Efficient Edge Analytical Model on Docker Containers for Automated Monitoring of Public Restrooms in India. In Proceedings of the 2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), New Delhi, India, 14–17 December 2020; pp. 1–6. [CrossRef]
44. Naik, N. Building a virtual system of systems using docker swarm in multiple clouds. In Proceedings of the 2016 IEEE International Symposium on Systems Engineering (ISSE), Edinburgh, UK, 3–5 October 2016; pp. 1–3. [CrossRef]
45. Pratap Yadav, M.; Pal, N.; Kumar Yadav, D. A formal approach for Docker container deployment. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e6364. [CrossRef]
46. Tomar, A.; Mishra, P.; Bisht, R.; Kumar, P.S. A Step Towards Generation of DoS/DDoS Attacks Dataset for Docker-Centric Computing. *Int. J. Math. Eng. Manag. Sci.* **2022**, *7*, 81–91. [CrossRef]
47. Reis, D.; Piedade, B.; Correia, F.F.; Dias, J.P.; Aguiar, A. Developing Docker and Docker-Compose Specifications: A Developers' Survey. *IEEE Access* **2022**, *10*, 2318–2329. [CrossRef]
48. Lane-Walsh, S.; Stillerman, J.; Santoro, F.; Fredian, T. Introduction to MDSplus using Docker. *Fusion Eng. Des.* **2021**, *165*, 112121. [CrossRef]
49. Farshteindiker, A.; Puzis, R. Leadership Hijacking in Docker Swarm and Its Consequences. *Entropy* **2021**, *23*, 914. [CrossRef]
50. Naik, N. Performance Evaluation of Distributed Systems in Multiple Clouds using Docker Swarm. In Proceedings of the 2021 IEEE International Systems Conference (SysCon), Vancouver, BC, Canada, 15 April–15 May 2021; pp. 1–6, ISSN 2472-9647. [CrossRef]
51. Ksontini, E.; Kessentini, M.; Ferreira, T.d.N.; Hassan, F. Refactorings and Technical Debt in Docker Projects: An Empirical Study. In Proceedings of the 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), Melbourne, Australia, 15–19 November 2021; pp. 781–791, ISSN 2643-1572. [CrossRef]
52. Melo, L.; Wiese, I.; d'Amorim, M. Using Docker to Assist Q amp;A Forum Users. *IEEE Trans. Softw. Eng.* **2021**, *47*, 2563–2574. [CrossRef]
53. Hossen, M.I.; Hei, X. A Low-Cost Attack against the hCaptcha System. In Proceedings of the 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 27 May 2021; pp. 422–431. [CrossRef]
54. Dordevic, B.; Timcenko, V.; Pavlovic, O.; Davidovic, N. Performance comparison of native host and hyper-based virtualization VirtualBox. In Proceedings of the 2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 17–19 March 2021; pp. 1–4. [CrossRef]
55. Abdulsattar, O.L.; Al-Hemiary, E.H. Virtualized Network Management Laboratory for Educational Purposes. *ISC Int'L J. Inf. Secur.* **2019**, *11*, 6.
56. Ionescu, V.M.; Petrini, A.C. Virtualization Laboratory for Computer Networks at Undergraduate Level. In *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16*; Advances in Intelligent Systems and Computing; Graña, M., Lopez-Guede, J.M., Etxaniz, O., Herrero, A., Quintian, H., Corchado, E., Eds.; Springer International Publishing: Cham, Switzerland, 2017; Volume 527, pp. 776–784. [CrossRef]

57. Vojnak, D.T.; Eordevic, B.S.; Timcenko, V.V.; Strbac, S.M. Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation. In Proceedings of the 2019 27th Telecommunications Forum (TELFOR), Belgrade, Serbia, 26–27 November 2019; pp. 1–4. [CrossRef]
58. Ionescu, V.M.; Patel, M.; Hindocha, D. Alternatives for Running Linux Applications in Windows. In Proceedings of the 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 27–29 June 2019; pp. 1–4. [CrossRef]
59. Costa, G.; Russo, E.; Valenza, A. Damn Vulnerable Application Scanner. In Proceedings of the 5th Italian Conference on Cyber Security (ITASEC), Online, 7–9 April 2021; p. 15.
60. Tyagi, S.; Kumar, K. Evaluation of Static Web Vulnerability Analysis Tools. In Proceedings of the 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC), Solan, India, 20–22 December 2018; pp. 1–6. [CrossRef]
61. Makino, Y.; Klyuev, V. Evaluation of web vulnerability scanners. In Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Warsaw, Poland, 24–26 September 2015; pp. 399–402. [CrossRef]
62. Vyamajala, S.; Mohd, T.K.; Javaid, A. A Real-World Implementation of SQL Injection Attack Using Open Source Tools for Enhanced Cybersecurity Learning. In Proceedings of the 2018 IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 3–5 May 2018; pp. 0198–0202. [CrossRef]
63. Nagpure, S.; Kurkure, S. Vulnerability Assessment and Penetration Testing of Web Application. In Proceedings of the 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 17–18 August 2017; pp. 1–6. [CrossRef]
64. Sandhya, S.; Purkayastha, S.; Joshua, E.; Deep, A. Assessment of website security by penetration testing using Wireshark. In Proceedings of the 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 6–7 January 2017; pp. 1–4. [CrossRef]
65. Das, R.; Tuna, G. Packet tracing and analysis of network cameras with Wireshark. In Proceedings of the 2017 5th International Symposium on Digital Forensic and Security (ISDFS), Tirgu Mures, Romania, 26–28 April 2017; pp. 1–6. [CrossRef]
66. Ramirez, R.; Chang, C.K.; Liang, S.H. PLC Cybersecurity Test Platform Establishment and Cyberattack Practice. *Electronics* **2023**, *12*, 1195. [CrossRef]
67. Hines, C.D.; Chowdhury, M.M. Uncover Security Weakness Before the Attacker Through Penetration Testing. In Proceedings of the 2022 IEEE International Conference on Electro Information Technology (eIT), Mankato, MN, USA, 19–21 May 2022; pp. 492–497, ISSN 2154-0373. [CrossRef]
68. Ayyoub, B.; Abu-Ein, A.; Zahran, B.; Nader, J.; Al-Hazaimeh, O. Enhance Linux Security Server Misconfigurations and hardening Methods. *Inf. Sci. Lett. Info* **2023**, *12*, 1285–1298. [CrossRef]
69. Sharma, I.; Pahuja, V. Comparative Analysis of Open-Source Vulnerability Assessment Tools for Campus Area Network. In Proceedings of the 2023 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 1–3 March 2023; pp. 1–6. [CrossRef]
70. Mohan, A.; Swaminathan, G.A.; Shafana, N.J. Automated Tools and Techniques in Vulnerability Assessment. In Proceedings of the 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 20–22 January 2022; pp. 533–540. [CrossRef]
71. Glances—An Eye on Your System. 2025. Available online: https://nicolargo.github.io/glances/ (accessed on 19 December 2024).
72. Kok, G.X.; Choong, K.N.; Vethanayagam, C.; Owada, Y.; Sato, G. An Analysis of a Large Scale Wireless Image Distribution System Deployment. In Proceedings of the 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Kota Kinabalu, Malaysia, 27–28 April 2019; pp. 150–155. [CrossRef]
73. Manore, C.; Manjunath, P.; Larkin, D. Performance of Single Board Computers for Vision Processing. In Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 27–30 January 2021; pp. 0883–0889. [CrossRef]
74. JASP—A Fresh Way to Do Statistics. 2025. Available online: https://jasp-stats.org/ (accessed on 19 December 2024).
75. Houminer-Klepar, N.; Bord, S.; Epel, E.; Baron-Epel, O. Are pregnancy and parity associated with telomere length? A systematic review. *BMC Pregnancy Childbirth* **2023**, *23*, 733. [CrossRef]
76. Taherdoost, H. Sampling Methods in Research Methodology; How to Choose a Sampling Technique for Research. *SSRN Electron. J.* **2016** . Available online: https://www.semanticscholar.org/paper/Sampling-Methods-in-Research-Methodology%3B-How-to-a-Taherdoost/7f2379b5698c9bda5c0e58956a976bb8873e9214 (accessed on 2 June 2024).
77. Jager, J.; Putnick, D.L.; Bornstein, M.H. II. More Than Just Convenient: The Scientific Merits of Homogeneous Convenience Samples. *Monogr. Soc. Res. Child Dev.* **2017**, *82*, 13–30. [CrossRef]