Memory Utilisation of Docker-Based Databases Under SQL Injection: A Comparative Case Study

Ade Dotun Ajasa *, Hassan Chizari *, Abu Alam *

Posted Date: 17 April 2026

doi: 10.20944/preprints202604.1230.v1

Keywords: databases; performance; virtualisation

 doi:10.20944/preprints202604.1230.v1

Article

Memory Utilisation of Docker-Based Databases Under SQL Injection: A Comparative Case Study ‡

Ade Dotun Ajasa *, Hassan Chizari * and Abu Alam *

School of Computing and Engineering, University of Gloucestershire, Cheltenham, GL50 2RH, United Kingdom
* Correspondence: adeajasa@connect.glos.ac.uk (A.D.A.); hchizari@glos.ac.uk (H.C.); aalam@glos.ac.uk (A.A.)
‡ This work is the PhD research of Ade Dotun Ajasa supervised by Hassan Chizari and Abu Alam.

Abstract

Docker is widely used to deploy applications in containerised environments. This study investigates whether the physical memory utilisation of databases deployed in Docker differs from that of equivalent non-Docker deployments during a Structured Query Language injection (SQLi) attack. A quantitative approach was adopted, using Glances to collect system data, JASP 0.18 for descriptive statistics and paired-samples *t* tests, and StatKey to examine mean differences. Two application stacks were evaluated: DVWA (PHP/MariaDB) and Acunetix (MySQL). Within the conditions examined in this study, the Docker-based deployments did not demonstrate improved memory performance when compared with the non-Docker deployments during SQLi. Instead, the results suggest that the Docker-based configurations were associated with higher memory use.

Keywords: databases; performance; virtualisation

1. Introduction

Database performance is commonly evaluated using metrics such as Central Processing Unit (CPU) utilisation and memory usage. Although previous studies have examined the effect of Structured Query Language injection (SQLi) attacks on CPU performance, less attention has been given to memory utilisation, particularly when comparing Docker-based and non-Docker database deployments. In earlier work by Ajasa et al. [1], the analysis focused on CPU performance and identified memory as an area for further investigation. Building on that recommendation, this study examines memory utilisation during SQLi across four deployment conditions: Damn Vulnerable Web Application DVWA without Docker, Acunetix without Docker, Application DVWA with Docker, and Acunetix with Docker. The aim is to determine whether deploying a database in Docker is associated with different memory behaviour during an SQLi attack.

The remainder of the paper is organised as follows. Section 2 presents the research question. Section 3 reviews related work on SQL injection, traditional databases, and Docker containers Section 4 describes Docker architecture and the data-collection process Section 5 the methodology outlines how DVWA and Acunetix were deployed with and without Docker. Section 6 Setup of the Implementation. Section 7 presents the results and statistical analysis using StatKey - Lock5 and JASP 0.18 (Jeffrey's Amazing Statistics Program). Section 8 summarises the main findings and concludes the paper.

2. Research Question

- How would the performance of the physical memory of a database installed in a Docker container compare to that of a database not installed in a Docker container during an SQLi attack?

3. Related Works

The author Gore et al. [2] deployed Docker containers and measured the memory utilisation during their implementation. By monitoring a containers resource metrics of memory utilisation

 doi:10.20944/preprints202604.1230.v1

we can detect if there is a leak in the memory which, could lead to an increase in the usage of the memory [3]. In comparison to the research papers [2,3], the research paper [4] installed various databases into Docker containers measured the data output of the memory usage, once executed resulted in less process from the server and low memory usage [4]. The research paper [5] measured memory usage during their implementation [5]. This research paper in relation to other research papers [2–5] compared the memory usage of databases DVWA non-Docker vs DVWA installed in Docker and Acunetix non-Docker vs Acunetix installed in Docker during an SQLi attack and collected the readings of the data output to be statistically analysed.

3.1. SQL Injection

The research paper Hassanzadeh et al. [6] published in 2021 mentioned Equifax as one of the companies that had its users data breached into in 2017 and according to a research paper [7] which, was also published in 2021 mentioned that Capitol One was a victim of a cyberattack on their database in 2019 yet, the research paper Khan et al. [8] published in 2023 refers to the 2019 data breach in Capital One as a data breach that affected over 100 million individuals who had the security and privacy of their personal information breached, this was the worst data breach since the data breach that occurred in companies such as Marriott and Target Equifax while, in the past decade the number of data hacks has increased [8]. Another key thing to remember, the research papers Liu et al. [9] and Li et al. [10] both published in 2023 said, hackers are receiving worldwide attention stealing confidential data.

Stephen P. Teale Data Centre in California had a massive data breach on 05.04.2002 by a hacker which, resulted in two hundred and sixty-five thousand state employees personal identities been compromised [9] and in an eye clinic in Singapore on 06.08.2021 they had the medical records that belonged to seventy-three thousand patients part of a data breach due to ransomware attacks in contrast, to Dedalus Biologie who were fined €1.5 million on 23.02.2021 by the (European Data Protection Board, 2022), for having a massive breach in their medical data which affected half a million people [10]. With, a cloud space hosting a Database Management System (DBMS) or stand-alone Database Management System hosting users data, we can look at the havoc Structured Query Language injection causes in the cloud space.

Private and valuable user data while, been hosted in a Database Management System have come under heavy attack due to cybersecurity lapses. Structured Query Language injection has and is still been used when hackers or criminals want access to private, valuable data via the Internet, data such as credit card details, personal medical records, national insurance numbers and account names with passwords. Once the Structured Query Language injection attack has been successful all the administrative rights are compromised. The main victims of Structured Query Language injection attack are hospitals and banks [11]. Lastly, a database would become liable to different types of attack if it is based on the Internet [12].

3.2. Docker Containers

The primary purpose of Docker containers is their ability to isolate individual microservices effectively [13]. Docker provides a service called (Docker Hub), the world's largest library of Docker images, which allows users to share or search for Docker images. This service is accessible via the Docker Hub website. Authors of the research paper Alyas et al. [14] supported the findings of the authors of research paper Moysiadis et al. [13] 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 [15]. 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 [3].

 doi:10.20944/preprints202604.1230.v1

4. Docker Architecture

Operating system level virtualisation and hardware system level virtualisation are the two types of virtualisation technology used in a Data Centre environment. In the operating system level virtualisation containers are created by resources been virtualised within the operating system which, is made up of operating systems libraries and their dependencies. Docker containers are a type of operating system virtualisation. But, in a hardware system level virtualisation the servers resources are virtualised into multiple virtual machines by a hypervisor.

To run different types of applications, each virtual machine has its own libraries and operating system. Kernel based virtual machine (KVM) a full virtualisation solution for Linux and A type-1 hypervisor Xen are examples of a hardware level virtualisation [16] (see Figure 1). We used the operating systems Ubuntu 20.04.2 LTS and Oracle (64-Bit) - Parrot OS 4.11 with, Acunetix and DVWA Docker images to create Docker containers. The user inputs commands from the Docker client which acts a primary interface. The Docker Registry represents a server.

4.1. How the Data was Collected

Table 1 shows the knowledge gap of other research papers that have used memory as a metric. We used the variable mem_used (%) in Table 2 as our metric. Table 3 is the descriptions of the main features present in the data that was collected using a software tool called Glances.

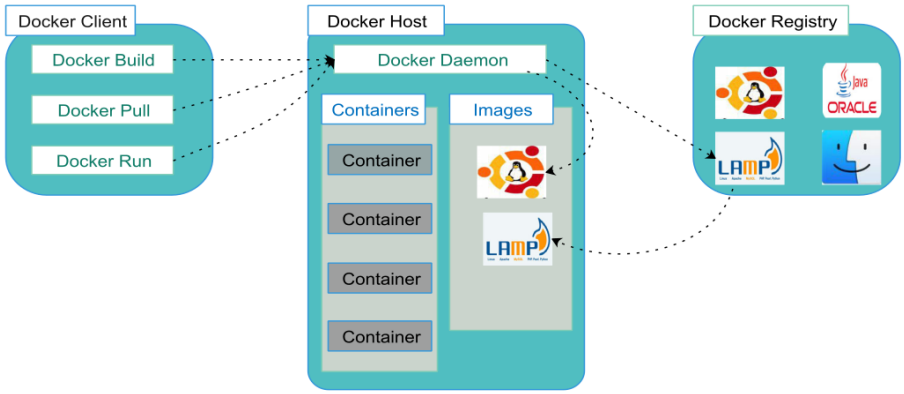

Figure 1. The Architecture Of Docker pratap [16].

Table 1. The knowledge gap of other research papers that have used memory as a metric.

Research Paper	Framework	Metric Measured	Knowledge Gap
Gore et el. [2]	Docker containers	Memory utilisation (%)	Memory performance of Docker vs non-Docker for databases
Zou et el. [3]	Docker container	Memory utilisation	Memory performance of Docker vs non-Docker for databases
Velasquez et el. [4]	Docker containers	Memory usage (mib)	Memory performance of Docker vs non-Docker for databases
Saha et el. [5]	Docker containers	Memory	Memory performance of Docker vs non-Docker for databases

Table 2. System resource monitoring over time.

Time	Uptime (s)	CPU (%)	Mem (%)	Host	OS	IP
25/05/23 12:53	2117	100.0	74.9	PhD-Parrot	Parrot OS 4.11	192.168.0.148
25/05/23 12:53	2120	24.7	74.9	PhD-Parrot	Parrot OS 4.11	192.168.0.148
25/05/23 12:53	2123	10.3	74.9	PhD-Parrot	Parrot OS 4.11	192.168.0.148
25/05/23 12:54	2126	3.6	75	PhD-Parrot	Parrot OS 4.11	192.168.0.148
25/05/23 12:54	2129	5.1	75	PhD-Parrot	Parrot OS 4.11	192.168.0.148

 doi:10.20944/preprints202604.1230.v1

Table 3. Descriptions of the main features present in the data collected using the software tool called Glances.

Interface to be Measured	What it does
mem_used - Mem (%)	the amount of memory used
timestamp	time all the data was recorded
uptime_minutes	time the data was recorded
uptime_seconds	uptime_minutes converted to uptime_seconds
network_enp0s3_interface_name	name of the physical network card
network_enp0s3_rx	records the throughput data
system_hostname	name of the virtual host (example, PhD-Parrot)
network_docker0_rx	network interface of Docker
system_hr_name	name of the virtual host operating system

5. Methodology

5.1. How Acunetix and DVWA were Deployed With and Without Docker

Following Figure 2 we began with the research question then, the type of SQLi we used the was Classic SQLi (In-band SQLi) which, was gathering of information and the SQLi attack was successful using the same communication channel Muscat et al. [17]. A quantitative method was chosen with an experimental design (see Table 4) for the databases Acunetix and DVWA. The control group consisted of the Acunetix database With Docker (WD) vs Acunetix database Non-Docker (ND) and DVWA database With Docker (WD) vs DVWA database Non-Docker (ND) while, the experimental group also consisted of Acunetix database With Docker (WD) vs Acunetix database Non-Docker (ND) and DVWA database With Docker (WD) vs DVWA database Non-Docker (ND). The variable we measured was mem_used (%) and we formed the null hypothesis and alternative hypothesis. The paired or dependent sample t test was chosen to find the significance difference between databases using the software JASP 0.18. StatKey - Lock5 was used to find the performance data and trade off's and also the significance difference in the databases to further examine the hypothesis. Lastly, was the research question answered? Yes then the end or no and go back and rewrite the research question.

Table 4. Experimental design.

	Acunetix	**DVWA**
Applications	Database	Database
Database engine	DBMS MySQL	PHP/MariaDB
Deployment type	Docker containers	Docker containers
Deployment type (Non)	Non-Docker containers	Non-Docker containers
VM resources	VirtualBox	VirtualBox
Operating systems	Ubuntu (64-Bit)	Ubuntu (64-Bit)
Attack script	Linux terminal command	Linux terminal command
Statistical analysis tool	StatKey	StatKey
Sampling interval	1000	1000
Run duration	10	10
Number of runs	10000	10000
Dependent variable	Memory	Memory

 doi:10.20944/preprints202604.1230.v1

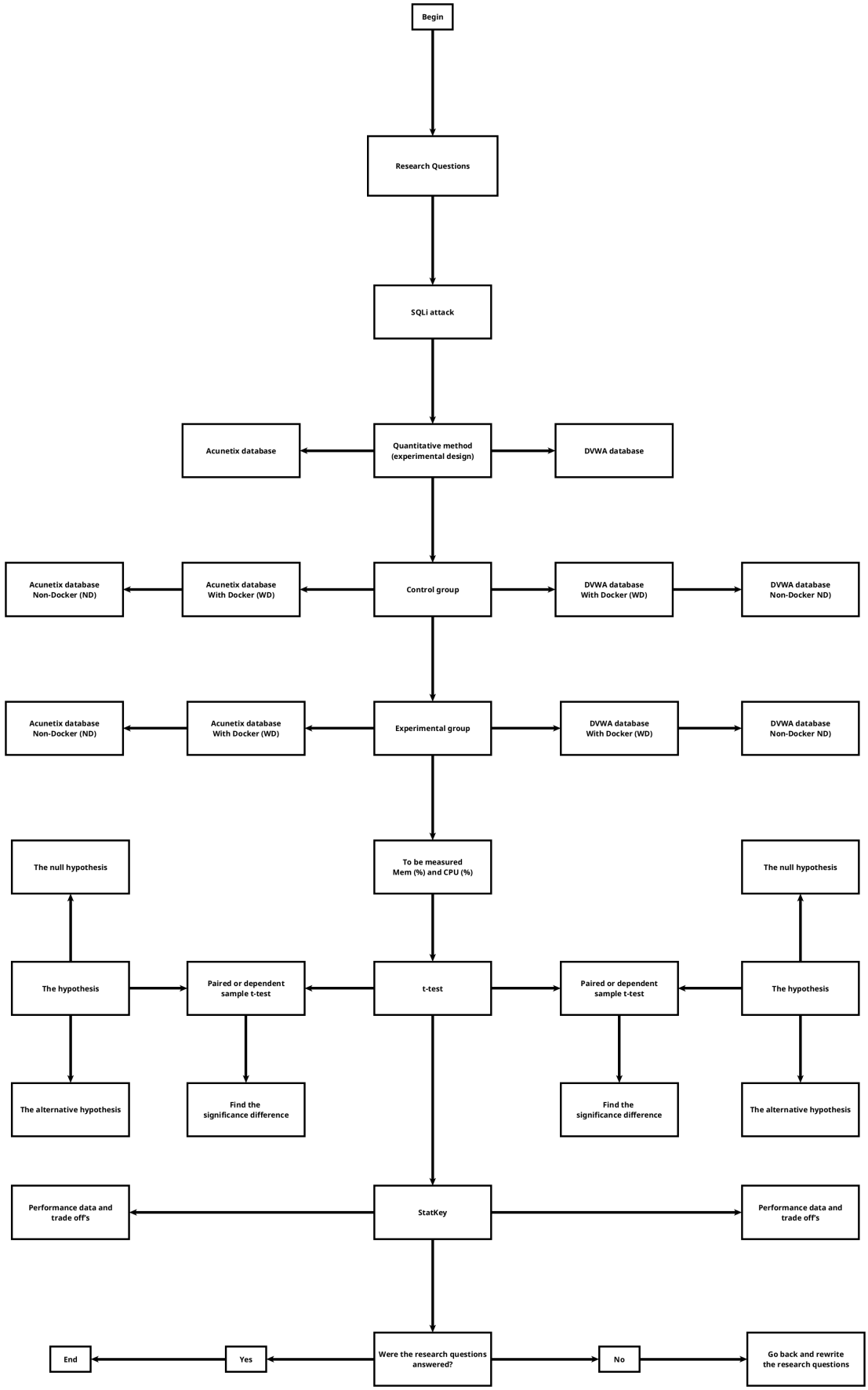

Figure 2. Methodology.

 doi:10.20944/preprints202604.1230.v1

6. Setup of the Implementation

Table 5 shows the specifications of the host computer and virtual machines, Table 6 lists the type of variables used, and Table 7 represents Docker vs. non-Docker databases. The following research papers Gore et al. [2], Zou et al. [3], Velasquez et al. [4] and Saha et al. [5], used memory as a metric. The system monitoring tool we used to collect the data was Glances and JASP 0.18 was used to calculate the descriptive statistics, distribution plots, paired samples t test. DVWA uses PHP/MariaDB database and Acunetix uses DBMS MySQL database. All the conditions were matched plus the deployment mode.

6.1. Validity of DVWA and Acunetix Databases

During their implementation, the following research papers [18,19] had used DVWA and Acunetix was used with the following research papers [20,21].

6.2. Validity of Glances and JASP

Glances was used to collect the data and has been used with the following research papers [22,23]. We used the software JASP 0.18 to analyse the descriptive statistics, calculate the t test, this piece of software has been used with the following research papers [24–32].

6.3. Hypothesis Testing and Finding the Right t Test for the Hypothesis

- H_0 - The null hypothesis - There is no significance difference between the means of the dependent variables - memory usage for both DVWA installed in Docker and DVWA not installed in Docker as well as, Acunetix installed in Docker and Acunetix not installed in Docker.
- H_a - The alternative hypothesis - There is a significance difference between the means of the dependent variables - memory usage for both DVWA installed in Docker and DVWA not installed in Docker as well as, Acunetix installed in Docker and Acunetix not installed in Docker.

$$H_0 : \mu_1 = \mu_2$$
$$H_a : \mu_1 \neq \mu_2$$

Table 5. 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	4GB	VMSVGA	Oracle (64-Bit) - Parrot OS 4.11
VM Database - Docker	Intel i7	2GB	VMSVGA	Ubuntu (64-Bit)
VM Database - No Docker	Intel i7	2GB	VMSVGA	Ubuntu (64-Bit)

Table 6. Type of variables.

Independent variables (cause)	uptime_minutes
Dependant variable (effect)	Mem (%) (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

Table 7. Docker vs. non-Docker databases.

Deployment (Docker vs. non-Docker)	DVWA with Docker vs. DVWA without Docker
Deployment (Docker vs. non-Docker)	Acunetix with Docker vs. Acunetix without Docker
Time	repeated measure
Metrics	mem_used (%)

 doi:10.20944/preprints202604.1230.v1

7. Results

The legend of the statistical analysis is explained in Table 8, paired samples *t* test results for Acunetix - During SQL_attack_WD and During SQL_attack_ND and paired samples *t* test results for DVWA - During SQL_attack_WD and During SQL_attack_ND are in Tables 9 and 10. Table 11 is a summary of the *t* test table of results. Table 12 and 13 represent the descriptive statistics tables, Figures 3 and 4 show the graphs and Figures 5–8 are the time series plots. We used the first 25 (Number of values) data for Acunetix - During SQL_attack_WD and During SQL_attack_ND (Table 12) and the first 45 (Number of values) DVWA - During SQL_attack_WD and During SQL_attack_ND (Table 13). The readings from Tables 12 and 13, Acunetix - During SQL_attack_WD and During SQL_attack_ND (Table 12), $t(24) = 45.26$ exceeds the CV(24) = $\pm$ 2.064 and the p value = <.001 is less than the p value <.05 from, those readings we concluded that there is definitely a significance difference between Acunetix - During SQL_attack_WD and During SQL_attack_ND (Table 12). DVWA - During SQL_attack_WD and During SQL_attack_ND (Table 13), $t(44) = 201.9$ exceeds the CV(44) = $\pm$ 2.021 and the p value = <.001 is less than the p value <.05 from, those readings we concluded that there is definitely a significance difference between DVWA - During SQL_attack_WD and During SQL_attack_ND (Table 13). Table 11 is the summary of Tables 12 and 13. Figures 3 and 4 shows there is more use of memory during the SQLi attack with the use of Docker. Figures 5–8 are the time series plots that show the data points that were collected in a time sequence. Based on the results listed above and the research question (2), Docker-based configurations required more memory and did not show any improved memory performance in contrast to the non-Docker databases.

Table 8. Legend.

ND	Non-Docker
WD	With-Docker
t	Time

Table 9. Paired samples *t* test results for Acunetix - During SQL_attack_WD and During SQL_attack_ND.

Formula	**-**	**Interpretation**
CV(24) = $\pm$ 2.064	-	$t(24) = 45.26$ exceeds the CV
df = n - 1	-	n = 24
p <.05	-	p = <.001 is less than p <.05
95%CI [8.853, 9.699]	-	Does not contain 0
95%CI	-	Confidence Interval
Cohen's *d*	-	9.1 (Effect - large)
CV(24)	-	Critical Value
df	-	Degrees of freedom
n = 25	-	Number of values
p	-	The p value
t	-	The paired samples t test

 doi:10.20944/preprints202604.1230.v1

Table 10. Paired samples *t* test results for DVWA - During SQL_attack_WD and During SQL_attack_ND.

Formula	-	**Interpretation**
CV(44) = ± 2.021	-	*t*(44) = 201.9 exceeds the CV
df = n - 1	-	n = 44
p <.05	-	*p* = <.001 is less than *p* <.05
95%CI [32.28, 32.93]	-	Does not contain 0
95%CI	-	Confidence Interval
Cohen's *d*	-	30.10 (Effect - large)
CV(44)	-	Critical Value
df	-	Degrees of freedom
n = 45	-	Number of values
p	-	The *p* value
t	-	The paired samples *t* test

Table 11. Readings from the paired Samples *t* test results - Tables 12 and 13.

Database	***t*()**	***p***	**95%CI**
Acunetix	*t*(24) = ± 2.064	<.001	[8.853, 9.699]
DVWA	*t*(44) = ± 2.021	<.001	[32.28, 32.93]

Table 12. Descriptive statistics for Acunetix - During SQL_attack_WD and During SQL_attack_ND.

Database	**Acunetix**				**Acunetix**			
No. Cases	**25**				**25**			
Container	**SQL_attack_WD**		**SQL_attack_ND**		**SQL_attack_WD**		**SQL_attack_ND**	
Scenario	Before	During	Before	During	Before	During	Before	During
Valid	-	25	-	-	-	-	-	25
Missing	-	0	-	-	-	-	-	0
Variance	-	1.618	-	-	-	-		0.001
Mean	-	53.68	-	-	-	-	-	25.19
Range	-	4.200	-	-	-	-	-	0.200
Std. Deviation	-	1.272	-	-	-	-	-	0.038
Minimum	-	51.00	-	-	-	-	-	25.00
Maximum	-	55.20	-	-	-	-	-	25.20
Lag 1 Autocorrelation	-	0.938	-	-	-	-	-	-

 doi:10.20944/preprints202604.1230.v1

Table 13. Descriptive statistics for DVWA - During SQL_attack_WD and During SQL_attack_ND.

Database	**DVWA**				**DVWA**			
No. Cases	**45**				**45**			
Container	**SQL_attack_WD**		**SQL_attack_ND**		**SQL_attack_WD**		**SQL_attack_ND**	
Scenario	Before	During	Before	During	Before	During	Before	During
Valid	-	45	-	-	-	-	-	45
Missing	-	0	-	-	-	-	-	0
Variance	-	1.117	-	-	-	-		0.011
Mean	-	75.90	-	-	-	-	-	43.30
Range	-	3.900	-	-	-	-	-	0.600
Std. Deviation	-	1.057	-	-	-	-	-	0.107
Minimum	-	73.60	-	-	-	-	-	43.20
Maximum	-	77.50	-	-	-	-	-	43.80
Lag 1 Autocorrelation	-	0.731	-	-	-	-	-	0.164

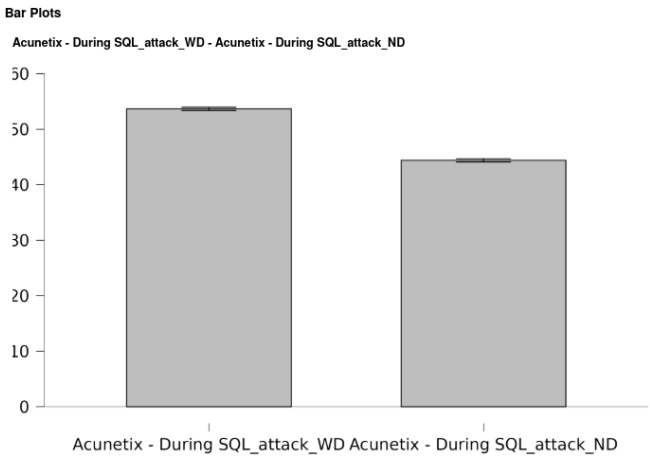

Figure 3. Acunetix distribution plots statistics - during an SQL injection attack with Docker and during an SQL injection attack with non-Docker - graph.

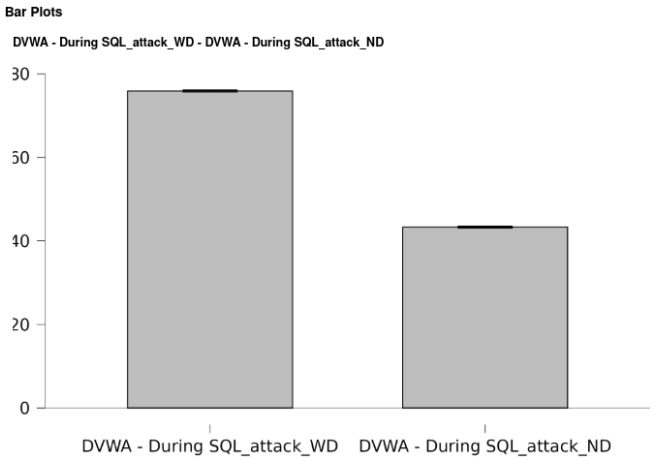

Figure 4. DVWA distribution plots statistics - during an SQL injection attack with Docker and during an SQL injection attack with non-Docker - graph.

 doi:10.20944/preprints202604.1230.v1

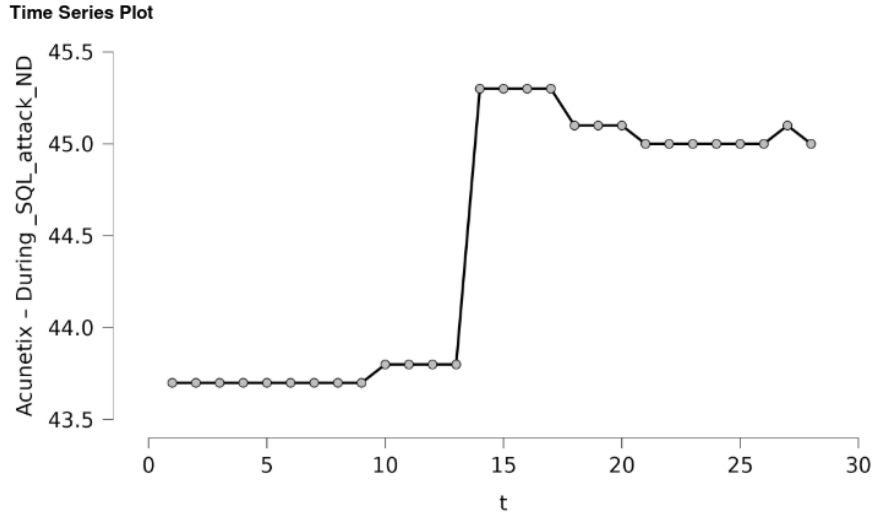

Figure 5. Time series plot of Acunetix during_SQL_attack_ND.

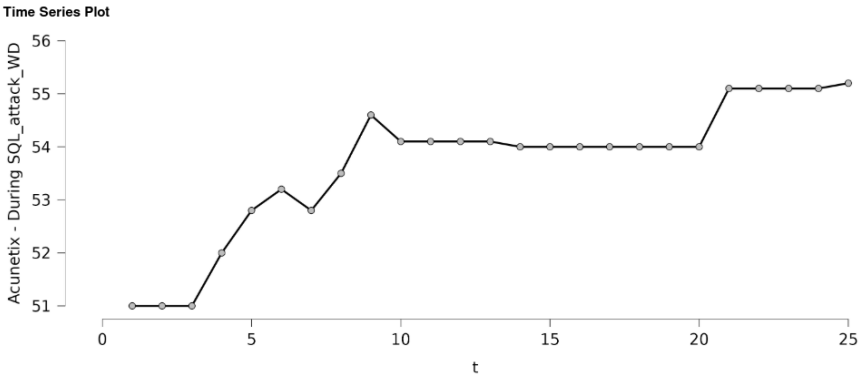

Figure 6. Time series plot of Acunetix before_SQL_attack_WD.

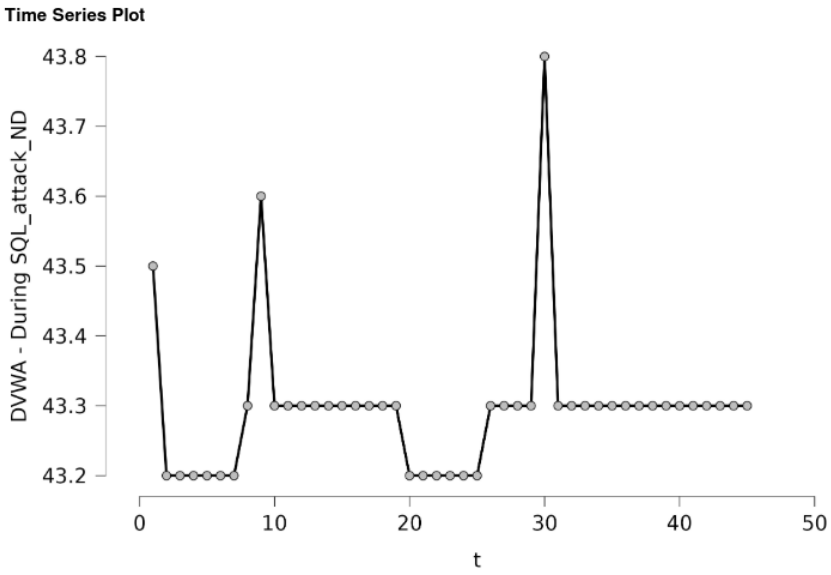

Figure 7. Time series plot of DVWA during_SQL_attack_ND.

 doi:10.20944/preprints202604.1230.v1

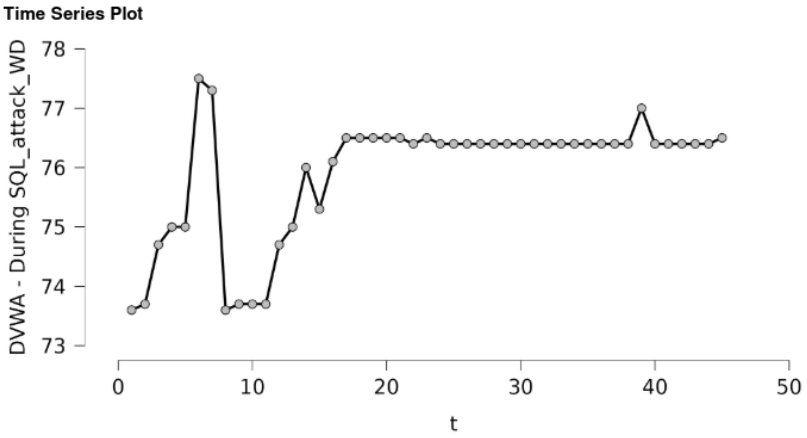

Figure 8. Time series plot of DVWA during_SQL_attack_WD.

7.1. Performance Data and Trade Off's

The software StatKey was used to analyses the data we collected. Table 14, shows the baseline output of the data that was retrieved from the implementation and is coloured in light blue while, the data that was retrieved from the implementation during the SQLi attack is drawn with a light brown colour. Figures 9–12 show the light blue and light brown histogram. Based of Figures 9 and 10 are the baselines of the systems memory once the database is switched on with no other input into the database from the administrator. Figures 11 and 12 were the recordings during the SQLi attack. The baselines of the Figures 9 and 10 reading look alike despite having different data and Figures 11 and 12 shows that the light brown histogram has a slight improvement in performance.

7.2. Acunetix Mem-ND and Mem-WD before the SQLi attack

At first glance the output of Figure 9 resembles the output of Figure 10 but, looking at the summary statistics that, is where we notice the difference. An example, Figure 9 (Acunetix Mem-ND and Mem-WD before the SQLi attack) has the same sample size but, the mean (average) standard deviation (irregular distribution of data around the mean) and the median (middle number) differ. Figure 10 (DVWA Mem-ND and Mem-WD before the SQLi attack) has the same sample size as Figure 9 (Acunetix Mem-ND and Mem-WD before the SQLi) but, the output of the mean, standard deviation and the median of Figure 10 (DVWA Mem-ND and Mem-WD before the SQLi attack) are different.

To summarise, there is a big difference between in the mean, median and standard deviation of Acunetix Mem_WD (mean - 74.931) (median - 74.90) (standard deviation - 0.048) and Acunetix Mem_ND (mean - 25.175) (median - 25.0) (standard deviation - 0.046) both using the same sample size of 16.

7.3. DVWA Mem-ND and Mem-WD before the SQLi attack

The Figure 10 (DVWA Mem-ND and Mem-WD before an SQLi attack) has a big difference in the graph between the mean, median and standard deviation of DVWA Mem_WD (mean - 71.181) (median - 71.20) (standard deviation - 0.040) while, DVWA Mem_ND has a (mean - 28.725) (median - 28.70) (standard deviation - 0.045) both using the same sample size of 16.

7.4. Acunetix Mem-ND and Mem-WD during the SQLi attack

The only similarities between Figure 11 and Figure 12 is that, their sample size was the same. In Figure 11 (Acunetix Mem-ND during the SQLi attack) and (Acunetix Mem-WD during the SQLi attack) on the statistics summary had a big difference between their mean. There is a big difference between in the mean, median and standard deviation of Acunetix Mem_ND (mean - 53.315) (median - 54.00) (standard deviation - 1.165) and Acunetix Mem_WD (mean - 75.230) (median - 75.00) (standard deviation - 1.311) both using the same sample size of 20.

 doi:10.20944/preprints202604.1230.v1

7.5. DVWA Mem-ND and Mem-WD during the SQLi attack

Figure 12 shows there is a big difference between the mean, median and standard deviation of DVWA Mem_WD (mean - 44.250) (median - 43.80) (standard deviation - 0.729) and DVWA Mem_ND (mean - 43.290) (median - 43.30) (standard deviation - 0.708) both using the same sample size of 20. To summarise, relative to the non-Docker configurations the Docker-based configurations required more memory and did not show improved memory performance across the conditions it was tested in.

Table 14. StatKey histogram legend [33].

Light Blue histogram - Memory	Mem-ND (No Docker)
Light Brown histogram - Memory	Mem-WD (With Docker)

7.6. Acunetix and DVWA Memory Readings

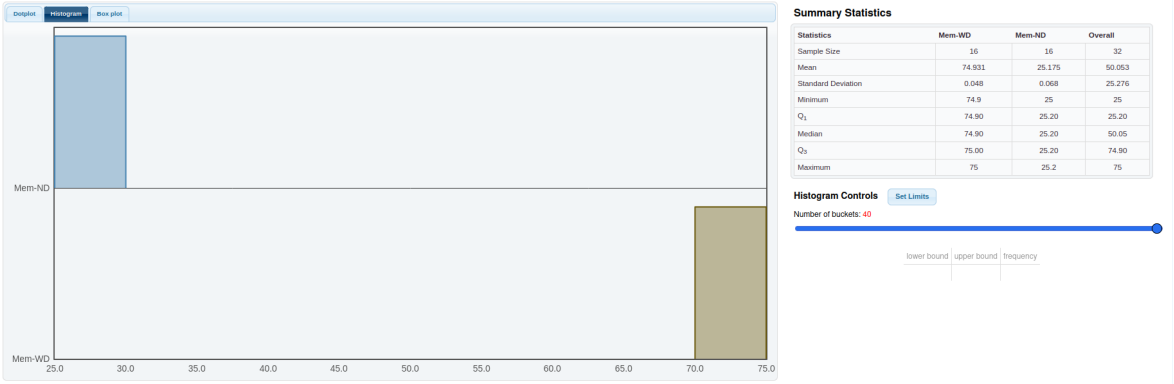

Figure 9. Acunetix Mem-ND and Mem-WD before an SQLi attack.

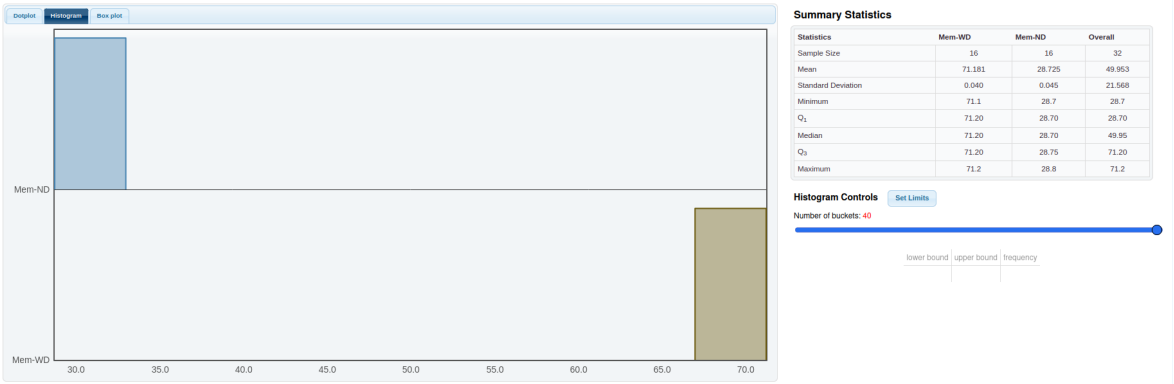

Figure 10. DVWA Mem-ND and Mem-WD before an SQLi attack.

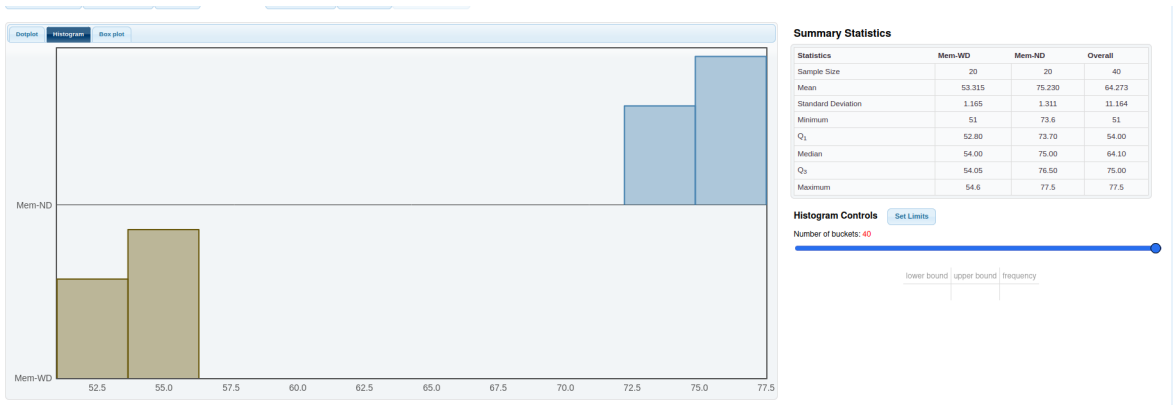

Figure 11. Acunetix Mem-ND and Mem-WD during an SQLi attack.

 doi:10.20944/preprints202604.1230.v1

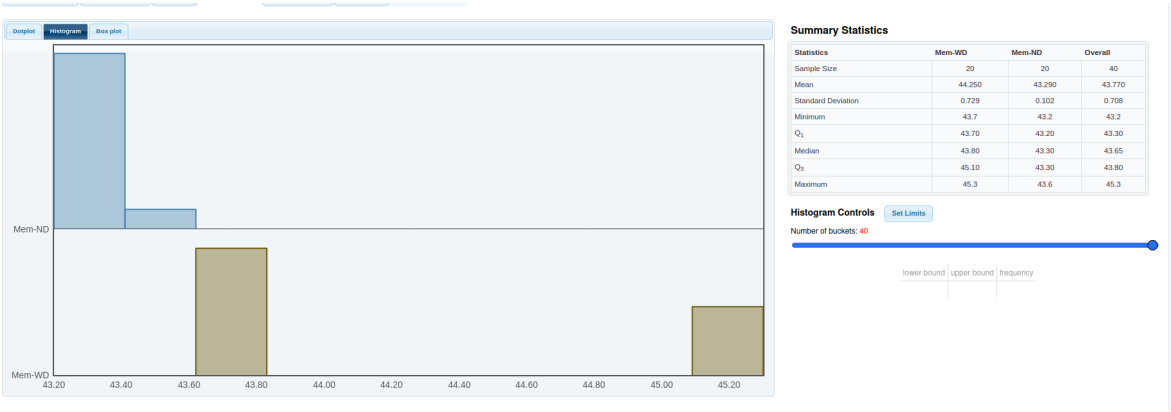

Figure 12. DVWA Mem-ND and Mem-WD during an SQLi attack.

Figures 13–16 and Figures 17–20 showed clearly that there was a significance difference between (Figures 13–16). There was also a significance difference between (Figures 17–20). Table 15 represents the legend of the bell curves Figures 13–20.

Primarily, we chose a two-tailed for the hypothesis testing (see [1]) then, from Figures 13–20 the probability (how likely this occurs) we chose a confidence level of 95% while, the remaining 5% was shared as 2.5% on the left-hand side and 2.5% on the right-hand side of the graph and a sample size of 10000. Figures 13 and 17 both have a slightly left-skewed distribution while, Figure 20 has a right-skewed distribution. Figures 14, 16–19 have a normal or symmetrical distribution. Figure 15 has a negative distribution that is skewed towards the left side.

Table 15. StatKey bell curve legend [33].

Sample size used for Figures 13–20	10000
+ +	Significance difference
- -	Significance difference
+ -	No Significance difference
- +	No Significance difference
Black bars	Represents 95% of the bell curve
Right side red bar	Represents 2.5% of the bell curve
Left side red bar	Represents 2.5% of the bell curve

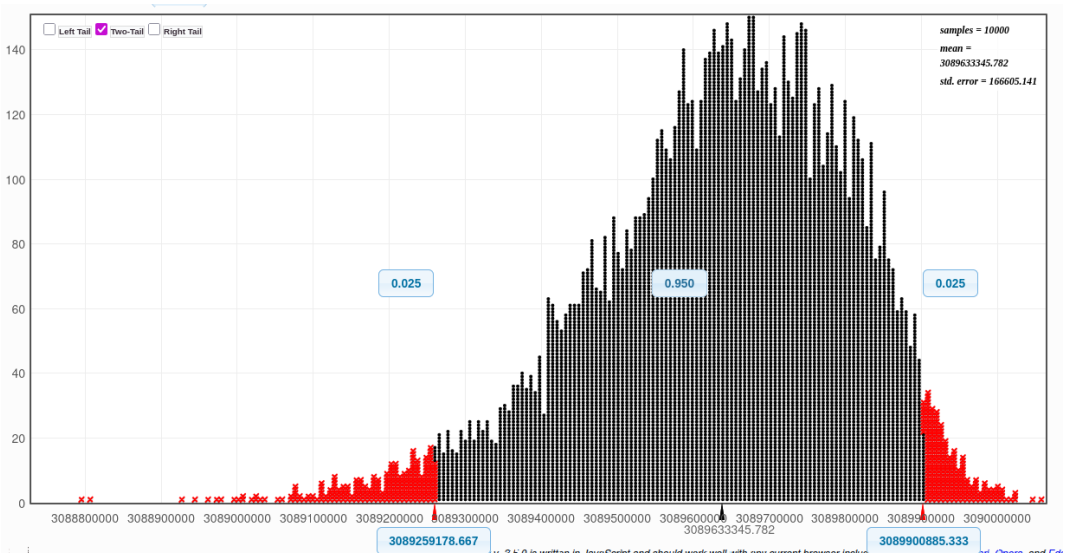

Figure 13. Acunetix with Docker installed before an SQLi attack.

 doi:10.20944/preprints202604.1230.v1

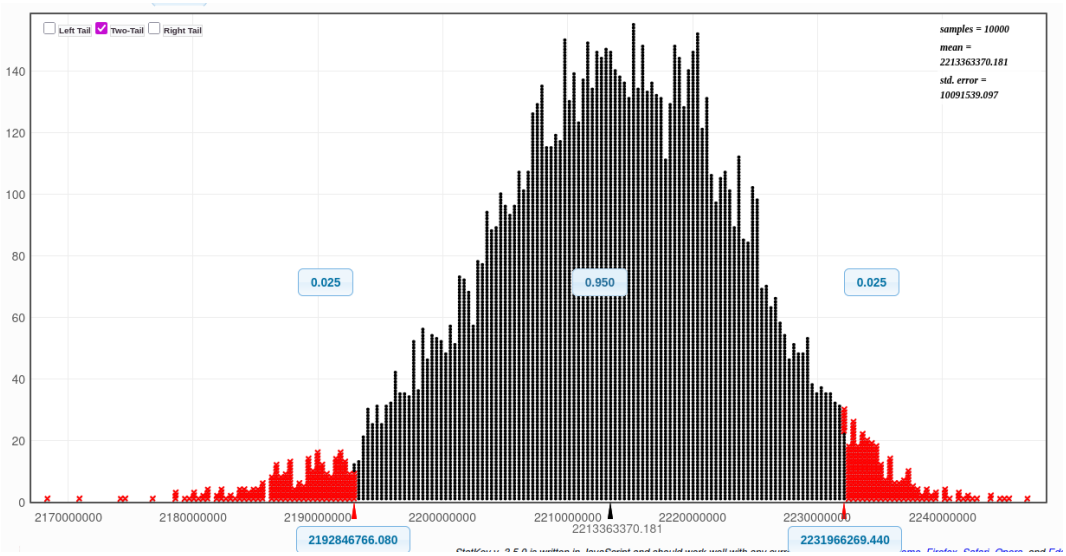

Figure 14. Acunetix with Docker installed during an SQLi attack.

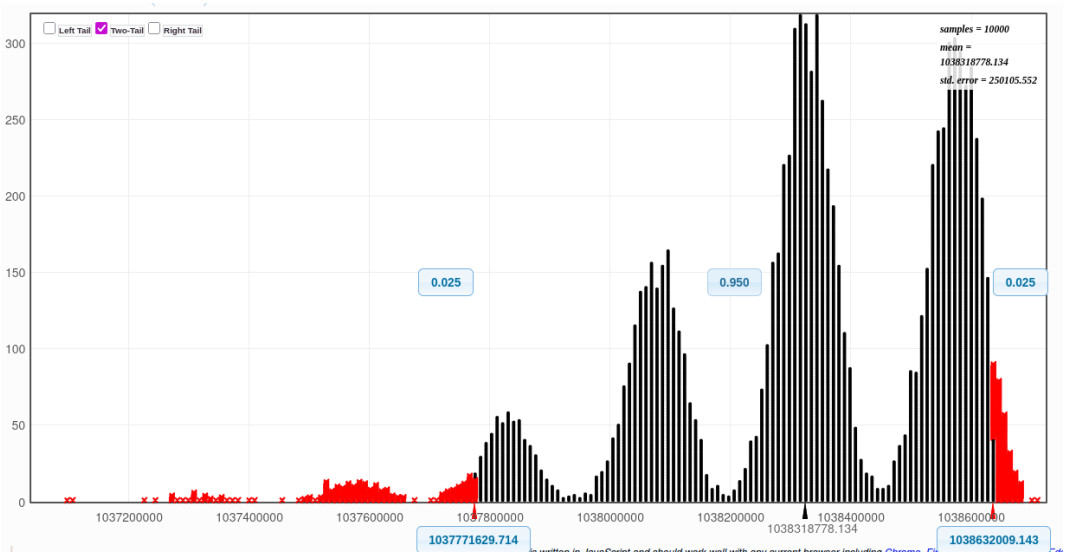

Figure 15. Acunetix with no Docker installed before an SQLi attack.

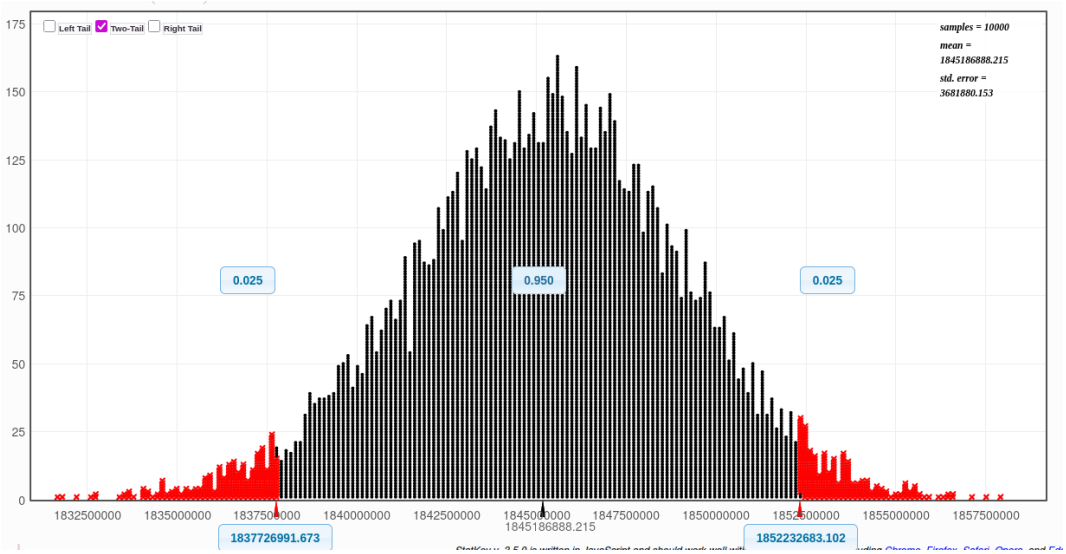

Figure 16. Acunetix with no Docker installed during an SQLi attack.

 doi:10.20944/preprints202604.1230.v1

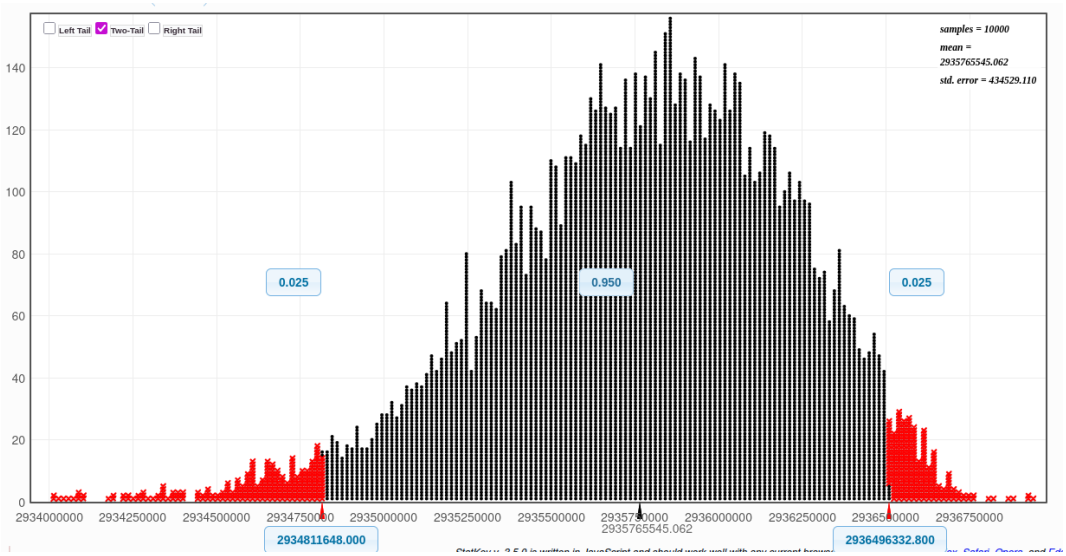

Figure 17. DVWA with Docker installed before an SQLi attack.

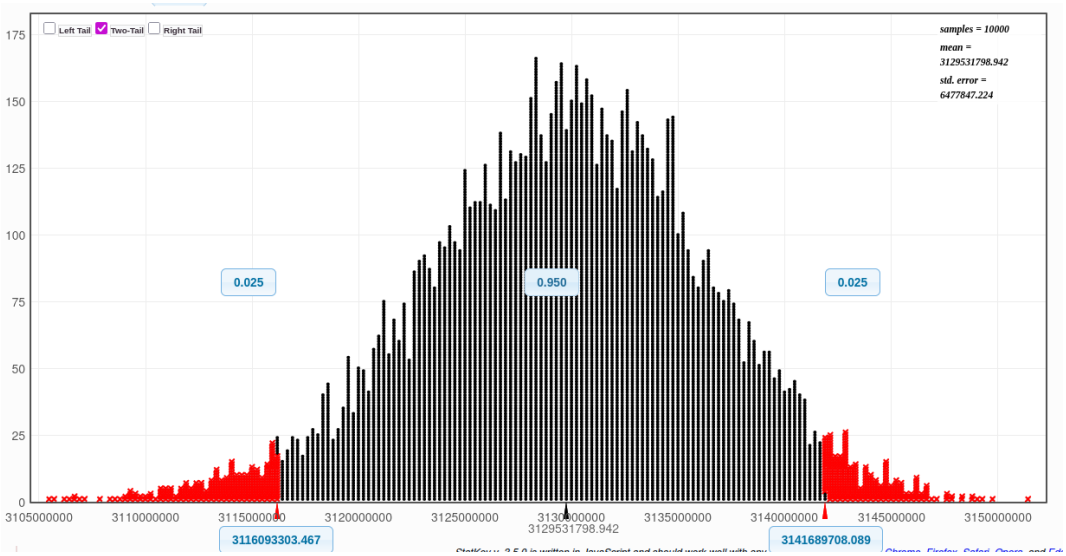

Figure 18. DVWA with Docker installed during an SQLi attack.

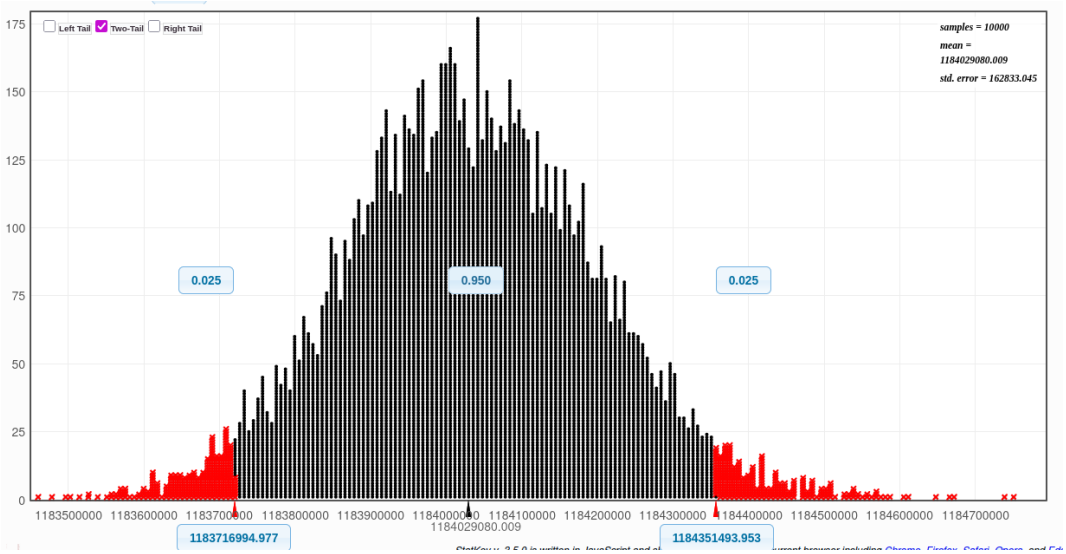

Figure 19. DVWA with no Docker installed before an SQLi attack.

 doi:10.20944/preprints202604.1230.v1

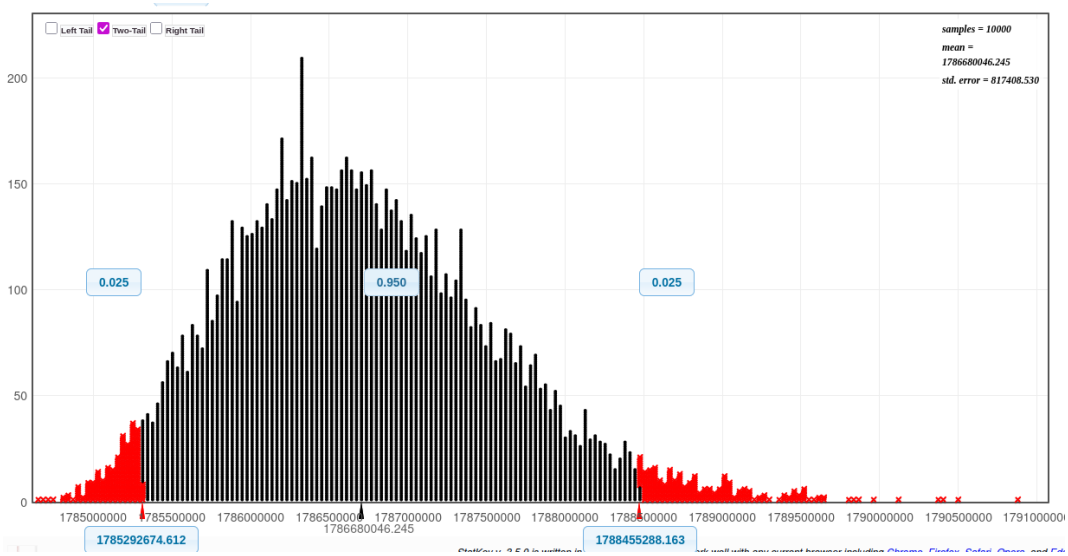

Figure 20. DVWA with no Docker installed during an SQLi attack.

8. Summary of Findings and Conclusion

This study examined whether deploying databases in Docker was associated with different memory utilisation from non-Docker deployment during SQL injection attacks. Across the conditions tested, the Docker-based configurations required more memory and did not show improved memory performance relative to the non-Docker configurations. These findings suggest that, within the experimental setup used in this study, Docker introduced additional memory overhead rather than a memory-efficiency advantage under SQLi conditions. Overall, the results do not support the claim that Docker improved database memory performance in this context. However, the findings should be interpreted within the limits of the current study design. Future work may build on this research by refining the experimental controls, clarifying the measurement procedure, and examining whether similar patterns are observed in other database environments and container configurations.

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 author(s).

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

References

1. 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. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, https://doi.org/10.3390/fi17040156.
2. 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). IEEE, 2020, pp. 1–6. https://doi.org/10.1109/ANTS50601.2020.9342845.
3. Zou, Z.; Xie, Y.; Huang, K.; Xu, G.; Feng, D.; Long, D. A Docker Container Anomaly Monitoring System Based on Optimized Isolation Forest **2022**. *10*, 134–145. Conference Name: IEEE Transactions on Cloud Computing, https://doi.org/10.1109/TCC.2019.2935724.
4. 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), New York, 2018. WOS:000493071300040.
5. Saha, P.; Govindaraju, M.; Marru, S.; Pierce, M. Integrating Apache Airavata with Docker, Marathon, and Mesos **2016**. *28*, 1952–1959. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3708, https://doi.org/10.1002/cpe.3708.

 doi:10.20944/preprints202604.1230.v1

6. Hassanzadeh, Z.; Biddle, R.; Marsen, S. User Perception of Data Breaches. *IEEE Transactions on Professional Communication* **2021**, *64*, 374–389. Conference Name: IEEE Transactions on Professional Communication, https://doi.org/10.1109/TPC.2021.3110545.
7. Neto, N.N.; Madnick, S.; Paula, A.M.G.D.; Borges, N.M. Developing a Global Data Breach Database and the Challenges Encountered. *Journal of Data and Information Quality* **2021**, *13*, 1–33. https://doi.org/10.1145/3439873.
8. Khan, S.; Kabanov, I.; Hua, Y.; Madnick, S. A Systematic Analysis of the Capital One Data Breach: Critical Lessons Learned **2023**. *26*, 1–29. https://doi.org/10.1145/3546068.
9. Liu, J.; Ni, X. Ordeal by innocence in the big-data era: Intended data breach disclosure, unintended real activities manipulation **2023**. *n/a*. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/eufm.12410, https://doi.org/10.1111/eufm.12410.
10. Li, Y.; Mamon, R. Modelling health-data breaches with application to cyber insurance **2023**. *124*, 102963. https://doi.org/10.1016/j.cose.2022.102963.
11. 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, 2019; pp. 145–149. WOS:000569996300024, https://doi.org/10.1109/CSCI49370.2019.00031.
12. 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), 2017, pp. 383–391. ISSN: 2377-2697, https://doi.org/10.1109/NIGERCON.2017.8281910.
13. 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). IEEE, 2022, pp. 1–5. https://doi.org/10.1109/MOCAST54814.2022.9837727.
14. 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 **2022**. https://doi.org/10.32604/iasc.2023.028815.
15. 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 **2021**. *9*, 8820–8834. https://doi.org/10.1109/ACCESS.2021.3049564.
16. Pratap Yadav, M.; Pal, N.; Kumar Yadav, D. A formal approach for Docker container deployment. *Concurrency and Computation: Practice and Experience* **2021**, *33*. https://doi.org/10.1002/cpe.6364.
17. Muscat, I. SQLi part 4: In-band SQLi (Classic SQLi), 2015.
18. 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). IEEE, 2015, pp. 399–402. https://doi.org/10.1109/IDAACS.2015.7340766.
19. 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). IEEE, 2018, pp. 1–6. https://doi.org/10.1109/PDGC.2018.8745996.
20. 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). IEEE, 2017, pp. 1–6. https://doi.org/10.1109/ICCUBEA.2017.8463920.
21. 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). IEEE, 2018, pp. 0198–0202. https://doi.org/10.1109/EIT.2018.8500136.
22. 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). IEEE, 2021, pp. 0883–0889. https://doi.org/10.1109/CCWC51732.2021.9376035.
23. 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). IEEE, 2019, pp. 150–155. https://doi.org/10.1109/ISCAIE.2019.8743734.
24. Houminer-Klepar, N.; Bord, S.; Epel, E.; Baron-Epel, O. Are pregnancy and parity associated with telomere length? A systematic review **2023**. *23*, 733. https://doi.org/10.1186/s12884-023-06011-8.

 doi:10.20944/preprints202604.1230.v1

25. Figueiredo, A.; Raposo, R.; Bem-Haja, P.; Costa, M. Parenting styles and the connection with nature: A look into a nature program **2023**. *4*, 291–305. Number: 3, https://doi.org/10.37291/2717638X.202343288.
26. Phutietsile, G.O.; Fotaki, N.; Jamieson, H.A.; Nishtala, P.S. The association between anticholinergic burden and mobility: a systematic review and meta-analyses **2023**. *23*, 161. https://doi.org/10.1186/s12877-023-03820-6.
27. Pfadt, J.M.; Bergh, D.v.d.; Sijtsma, K.; Wagenmakers, E.J. A tutorial on Bayesian single-test reliability analysis with JASP **2023**. *55*, 1069–1078. https://doi.org/10.3758/s13428-021-01778-0.
28. Nwokenna, E.N.; Sewagegn, A.A.; Falade, T.A. Effect of educational music intervention on college students' aggressive behaviour **2023**. *102*, e32472. https://doi.org/10.1097/MD.0000000000032472.
29. Fute, A.; Sun, B.; Oubibi, M. General Self-Esteem as the Mechanism Through Which Early-Childhood Parental Trust and Support Affect Adolescents' Learning Behavior: A Moderated Mediation Model **2023**. *60*, 00469580231152076. Publisher: SAGE Publications Inc, https://doi.org/10.1177/00469580231152076.
30. Montoya-Hurtado, O.; Gómez-Jaramillo, N.; Bermúdez-Jaimes, G.; Correa-Ortiz, L.; Cañón, S.; Juárez-Vela, R.; Santolalla-Arnedo, I.; Criado-Pérez, L.; Pérez, J.; Sancho-Sánchez, M.C.; et al. Psychometric Properties of the Multidimensional Assessment of Interoceptive Awareness (MAIA) Questionnaire in Colombian University Students **2023**. *12*, 2937. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute, https://doi.org/10.3390/jcm12082937.
31. McMullan, J.C.; Rainey, L.; Morgan, D.; Johnston, L. The effect of COVID-19 on the cervical screening programme within a Northern Irish Health and Social care trust **2023**. *92*, 84–88.
32. Junior, E.; Jesus, P.; Borges, E.M. Whey Protein Analysis Using the Lowry Assay and 96-Well-Plate Digital Images Acquired Using Smartphones **2023**. *100*. https://doi.org/10.1021/acs.jchemed.2c00830.
33. StatKey, 2025.