Securing Databases: A Comparative Study on the Impact of Implementing SSL on MySQL 8.0.33

Antoni Haikal [1], Heru Wijanarko [2*], Ocha Oktafia [3], Muhammad Hikmah Husnuzon [4], Gunawan [5], Hamdani Arif [6]
[1,3,4,5,6] Cybersecurity Engineering Study Program,
Department of Electrical Engineneering, State Polytechnic of Batam, Batam, Indonesia
[2] Mechatronics Engineering Study Program,
Department of Electrical Engineneering, State Polytechnic of Batam, Batam, Indonesia

[1]antoni@polibatam.ac.id, [2*]wijanarko@polibatam.ac.id, [3]ocha.oktafia11@students.polibatam.ac.id,
[4]muhammad.hikmah.husnuzon@students.polibatam.ac.id, [5]gunawan3@students.polibatam.ac.id, [6]hamdaniarif@polibatam.ac.id

***Abstract*—The increasing reliance on online communication and data storage has made protecting sensitive information a top priority for businesses and individuals. This study investigates the effectiveness of different security protocols in safeguarding databases from external threats. By comparing the performance of databases that use SSL and those that do not, this project provided valuable insights into the importance of securing data in transit and the impact of security protocols on database performance. The results of this research could be helpful for companies and individuals looking to improve the security of their databases and protect sensitive information. These research findings demonstrate that enabling SSL encryption leads to longer average execution times for database operations. This latency can be attributed to the computational workload associated with SSL's encryption and decryption processes. However, the trade-off between performance and security is essential to safeguarding data in transit.**

***Keywords*—Database, MySQL, performance, query, SSL.**

I. INTRODUCTION

Database security has seen significant advancement over the past decades, yet the challenge of securing databases against external threats remains pertinent. According to studies by [1]–[3], databases remain a prime target for cyber attackers due to the value of the information stored. Consequently, the demand for adequate security protocols is more crucial than ever.

A variety of security protocols have been developed and implemented in attempts to safeguard data. Among these, Secure Sockets Layer (SSL) has gained widespread attention. SSL provides an added layer of security, ensuring data integrity and confidentiality during electronic transmission [4]. It establishes an encrypted link between a web server and a browser, ensuring that all data remains private [5]. However, studies have noted potential performance trade-offs associated with implementing SSL, such as the increased computational overhead of web applications [6].

Many researchers have emphasized the importance of data security during transit. The [7]–[9] posted that the security of data in transit is equally, if not more, important than at-rest security. While data encryption at rest can help prevent unauthorized access, ensuring data security during transit is vital to prevent interception attacks.

However, there is a knowledge gap concerning the comparative effectiveness of databases that use SSL versus those that do not. Limited empirical research has been conducted to investigate the impact of SSL on database performance and security, leading to a lack of concrete guidance for businesses and individuals.

This study aims to fill this gap by comparing the performance and security of databases with and without SSL implementation. It will enhance understanding of SSL's role in database security and provide insights into the potential trade-offs that businesses might need to consider when implementing SSL.

II. METHOD

To answer the research question and compare the performance and security of databases that use SSL versus Non SSL, this study adopted an experimental research Design, specifically, a comparison experiment.

A. Data Collection

1) Database Selection:

The study was conducted on two distinct sets of databases those with SSL implementation and Non-SSL. These databases were selected across various industries and sizes to represent real-world scenarios comprehensively. The databases were chosen based on their availability, access permissions, and the ability to manipulate security settings.

2) Performance Measurement:

The performance of these databases was measured using a variety of key performance indicators (KPIs), such as response time, ram used, and CPU used. These KPIs were chosen due to their relevance to daily database operations and their potential impact on business productivity.

B. Data Analysis

The collected data were statistically analyzed to identify trends and differences in performance and security between databases with SSL and those without. This analysis helped determine whether SSL use significantly affects database performance and security.

1) Identification of Hardware and Software Requirements

The hardware and software used in the testing process are identified in this phase. This research utilizes virtualization, explicitly making use of VirtualBox. The specifications of virtual machine components are shown in Table I.

TABLE I
SPECIFICATIONS OF VIRTUAL MACHINE COMPONENTS

Component	Version
Virtual Machines	
Memory	2,00 GB RAM
Processor	2 CPUs
Storage	2 TB (Dynamic)
Host	
Memory	32 GB RAM
Processor	Intel Core i9 Gen 10900
Storage	SSD 474 GB
Software	
VirtualBox	7.0.4
Ubuntu Server	20.04.1
MySQL	8.0.33

2) Dataset Determination

In this case study, the dataset consists of dummy data taken from the Mockaroo website in CSV format for testing on MySQL. The data is then converted to SQL using the Python programming language, allowing it to be input into the database. Experiments will use 10 thousand data records.

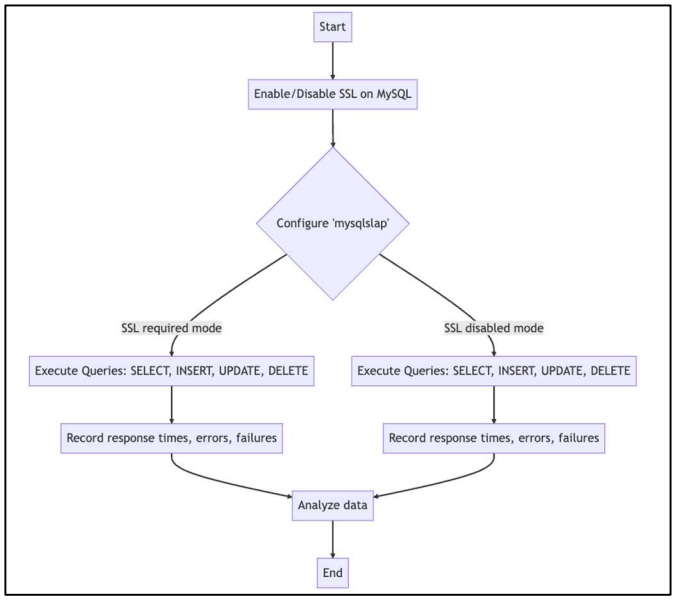


Figure 1. Step of testing the performances.

3) Design of Testing Steps

In this phase, we detail the steps taken to test the performance of both databases, utilizing the 'mysqlslap' utility in both SSL-required mode and disabled mode with additional SELECT, INSERT, UPDATE, and DELETE queries. The step of testing performances is shown in Fig.1.

4) Implementing and Testing

At this stage, testing begins on both sample database models with testing Mysqlslap Utility, displaying direct implementation along with the results in tables and graphs.

- Testing Insert Query Performance in SSL Mode Required and SSL Mode Disable. In this stage, we tried to input 10000 data records into the database. We are using the mysqlslap to do this stage as per the below syntax.

SSL Mode Required (insert)

```
sudo mysqlslap --user=[userdb] --
password=[userdbpassword] --
host=[hostdbvalue] --iterations=5 --
ssl-mode=REQUIRED --number-of-
queries=10000 -create-schema=ssl_test
--no-drop --query="insert.sql"
```

SSL Mode disabled (insert)

```
sudo mysqlslap --user=[userdb] --
password=[userdbpassword] --
host=[hostdbvalue] --iterations=5 --
ssl-mode=disabled --number-of-
queries=10000 -create-schema=ssl_test
--no-drop --query="insert.sql"
```

- Testing Update Query Performance in SSL Mode Required and SSL Mode Disable.
  In this stage, we tried to input 10000 data records into the database. We are using mysqlslap to do this stage as per the below syntax.

SSL Mode Required (update)

```
sudo mysqlslap --user=[userdb] --
password=[userdbpassword] --
host=[hostdbvalue] --iterations=5 --
ssl-mode=REQUIRED --number-of-
queries=10000 -create-schema=ssl_test
--no-drop --query="update.sql"
```

SSL Mode disabled (update)

```
sudo mysqlslap --user=[userdb] --
password=[userdbpassword] --
host=[hostdbvalue] --iterations=5 --
ssl-mode=disabled --number-of-
queries=10000 -create-schema=ssl_test
--no-drop --query="update.sql"
```

- Testing Delete Query Performance in SSL Mode Required and SSL Mode Disable.
  In this stage, we tried to input 10000 data records into the database. We are using mysqlslap to do this stage as per the below syntax.

SSL Mode Required

```
sudo mysqlslap --user=[userdb] --
password=[userdbpassword] --
host=[hostdbvalue] --iterations=5 --
ssl-mode=REQUIRED --number-of-
queries=10000 -create-schema=ssl_test
--no-drop --query="delete.sql"
```

SSL Mode disabled

```
sudo mysqlslap --user=[userdb] --
password=[userdbpassword] --
host=[hostdbvalue] --iterations=5 --
ssl-mode=disabled --number-of-
queries=10000 -create-schema=ssl_test
--no-drop --query="delete.sql"
```

- Testing Select Query Performance in SSL Mode Required and SSL Mode Disable.
  In this stage, we tried to input 10000 data records into a database. We are using mysqlslap to do this stage as per the below syntax.

SSL Mode Required

```
sudo mysqlslap --user=[userdb] --
password=[userdbpassword] --
host=[hostdbvalue] --iterations=5 --
ssl-mode=REQUIRED --number-of-
queries=10000 -create-schema=ssl_test
--no-drop --query="select.sql"
```

SSL Mode disabled

```
sudo mysqlslap --user=[userdb] --
password=[userdbpassword] --
host=[hostdbvalue] --iterations=5 --
ssl-mode=disabled --number-of-
queries=10000 -create-schema=ssl_test
--no-drop --query="select.sql"
```

III. RESULTS AND DISCUSSION

This study examined the impact of using Secure Sockets Layer (SSL) encryption on the performance of MySQL databases. Using the mysqlslap utility, we evaluated the performance of four types of queries (Insert, Update, Delete, and Select) under two different settings: SSL Mode Required and SSL Mode Disabled. The aim was to evaluate the trade-off between enhanced security and potential performance degradation.

Our experiments were conducted using a consistent testing environment and the same volume of data for each type of query. Based on the results obtained from the CPU usage, RAM utilization, and response time, it was evident that using SSL encryption affects database performance.

As online threats continue to evolve and proliferate, ensuring data security in transit has become a critical concern for organizations. While SSL encryption may introduce some overhead to database operations, this study illustrates that the cost is justifiable considering the high level of protection it provides against data breaches and theft. Therefore, businesses and individuals should use SSL, or similar security measures, when transmitting sensitive data. Table II provides the data CPU usage comparing SSL required and SSL disabled.

In the provided Table II, we are focusing on the '%system' column, which signifies the CPU utilization by the system for operational management. Our analysis shows that without SSL, the highest value lies at the 30-second mark with a %system value of 19.71%. However, with SSL, the highest value also lies at the 30-second mark but with a %system value of 25.78%.

In addition, the average CPU usage also shows a disparity between conditions with and without SSL. For non-SSL conditions, the average CPU usage (%system) stands at 14.11%, whereas, with SSL, the average CPU usage (%system) is 17.53%.

From this analysis, we can infer that the usage of SSL in this system significantly impacts the rise in CPU utilization. This aligns with the general understanding that SSL encryption and decryption require additional computational resources. Despite this, the increase in resource usage is seen as a necessary cost for improving the security level of data transactions, which is a critical requirement in many applications. Therefore, security and performance must be considered when designing and managing systems.

Based on table III, it can be seen that the maximum value of %system in the insert query process that does not use SSL is at 8.26% at 25 seconds. While the maximum value for %system that uses SSL is at 9.24% at 20 seconds. average on the process of inserting SSL and No SSL. For insert processes that do not use SSL, the average value obtained is 5.95% and for insert processes that use SSL, the average value obtained is 6.40%.

Based on this analysis, it proves that there is an additional use of %system resources when using SSL. Even though these results are not significant, this proves that there is an influence on the use of SSL in the insert query.The analysis of the table above provides an insightful depiction of the variations in system resource usage under the SSL and non-SSL scenarios during the execution of 'Insert' database operations. Each record in the table symbolizes the system resource consumption at different points in time while running the 'Insert' operation.

Focusing on the '%system' column, which represents the percentage of CPU time spent on system-level operations, we can note a few striking observations. In the case of operations without SSL, the CPU usage peaks at 8.26% at the 25-second mark. This indicates the highest level of system resources allocated towards managing the 'Insert' query within the span of the operation. It is important to note that these results depict the system's response under an environment without the added overhead of encryption and decryption, inherent to SSL. On the other hand, when the same operation is performed with SSL, we observe a slightly higher peak system usage of 9.24% at the 20-second mark.

TABLE II
CPU USAGE RESULT DURING SELECT TEST (SSL AND NO SSL)

Times (s)	%user		%nice		%system		%iowait		%steal		%idle	
	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL
5	0	22.64	0	0	0	7.02	0	0	0	0	100	70.34
10	30.24	27.86	0	0	13.46	22.35	0.11	0	0	0	56.20	49.79
15	39.38	17.14	0	0	18.15	29.11	0	0	0	0	42.46	53.75
20	39.07	38.71	0	0	18.58	15.05	0	0	0	0	42.36	46.24
25	44.87	28.30	0	0	12.07	21.64	0	0.1	0	0	43.06	49.95
30	35.53	22.98	0	0	19.71	25.78	0	0	0	0	44.76	51.24
35	43.48	37.22	0.11	0	13.57	16.33	0.11	0	0	0	42.74	46.45
40	41.45	39.16	0	0	16.79	17.26	0	0	0	0	41.76	43.58
45	41.85	40.76	0	0	15.56	14.97	0	0	0	0	42.60	44.27
50	43.24	39.56	0	0	14.27	15.72	0.11	0	0	0	42.38	44.73
55	45.53	40.76	0	0	11.3	15.95	0	0	0	0	43.16	43.29
60	43.85	39.89	0	0	12.73	15.45	0	0	0	0	43.42	44.66
65	39.96	40.13	0	0	18.19	16.12	0	0	0	0	41.85	43.76
70	31.16	40.38	0	0	13.16	14.83	0.11	0	0	0	55.58	44.79
75	36.96	40.44	0.01	0	14.07	15.44	0.03	0	0	0	48.92	44.12

TABLE III
CPU USAGE RESULT DURING INSERT TEST (SSL AND NO SSL)

Times (s)	%user		%nice		%system		%iowait		%steal		%idle	
	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL
5	0.20	0.10	0	0	0.30	0.10	0.40	NULL	0	0	99.09	99.60
10	0.51	2.06	0	0	2.87	7.84	17.11	40.56	0	0	79.51	49.54
15	1.16	1.55	0	0	6.53	7.65	39.47	38.88	0	0	52.84	51.91
20	1.67	1.52	0	0	6.79	9.24	40.13	38.38	0	0	51.41	50.86
25	2.58	2.15	0.10	0	8.26	8.40	39.46	38.63	0	0	49.59	50.82
30	1.55	1.58	0	0	7.56	6.20	40.00	39.39	0	0	50.88	52.84
35	2.20	1.48	0	0	7.23	5.40	39.48	39.62	0	0	51.10	53.50
40	2.16	1.49	0	0	8.02	6.40	38.68	33.54	0	0	51.13	58.57

This increase can likely be attributed to the extra computational work associated with the SSL's encryption and decryption processes, effectively demanding more from the system resources.

Examining the averages, we see a subtle increase from the non-SSL 'Insert' operations to those with SSL, moving from 5.95% to 6.40% respectively. This underlines the slight additional overhead that comes with using SSL.

Despite the marginally increased demand on system resources with SSL, the difference may not significantly impact the overall performance of the system. However, the increased security provided by SSL during data transmission cannot be overlooked.

Based on table IV, it can be seen that at 5-15 seconds the value of the %system update process that does not use SSL has a lower value compared to the %system update process that uses SSL. Whereas the maximum value for the %system update process that does not use SSL is 3.38% at 45 seconds and for the %system update process that uses SSL is 1.9 at 15 seconds. For the average value, the update process does not use SSL has a value of 2.18% while the update process using SSL has an average value of 1.26%.

Based on the analysis, this is very different from the other three processes because the %system update process that does not use SSL has a higher value than the %system process that uses SSL. This may be caused by many things and this requires further research. Analyzing the '%system' column, we observe that in the initial 15 seconds, the Update Query process with SSL enabled records higher CPU utilization at the system level compared to the process without SSL. This can be attributed to the additional computational overhead induced by SSL due to encryption and decryption operations. However, interestingly, post the 20-second mark, the '%system' values for the process without SSL consistently supersede the process with SSL enabled. The maximum CPU utilization at the system level for the No SSL process peaks at 3.38% at the 45-second mark, while for the SSL process, it reaches a high of 1.9% at the 15-second mark.

TABLE IV
CPU USAGE RESULT DURING UPDATE TEST (SSL AND NO SSL)

Times (s)	%user		%nice		%system		%iowait		%steal		%idle	
	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL
5	0.10	18.90	0.10	0	0.10	0.92	0	1.12	0	0	99.70	79.06
10	37.62	44.57	0	0	1.25	1.79	0.21	2.32	0	0	60.92	51.32
15	44.13	44.08	0	0	1.59	1.90	1.06	2.54	0	0	53.23	51.48
20	39.18	44.88	0	0	3.86	1.27	1.55	2.22	0	0	55.41	51.64
25	40.37	45.35	0	0	2.97	0.95	2.09	2.33	0	0	54.57	51.37
30	39.53	45.86	0.11	0	3.01	0.84	2.00	2.31	0	0	55.35	51.00
35	38.72	44.48	0	0	3.37	1.49	2.13	2.23	0	0	55.78	51.80
40	39.73	44.70	0	0	2.90	1.38	2.12	2.22	0	0	55.25	51.69
45	40.50	44.40	0	0	3.38	1.59	1.86	2.33	0	0	54.26	51.69
50	44.29	44.66	0	0	1.80	1.48	2.33	2.22	0	0	51.59	51.64
55	45.31	44.47	0	0	1.26	1.37	2.42	2.42	0	0	51.00	51.74
60	44.90	43.97	0	0	1.68	1.17	2.21	2.35	0	0	51.21	52.51
65	28.82	0.10	0	0	1.14	0.20	1.77	0.10	0	0	68.26	99.60
70	36.97	39.06	0.02	0	2.15	1.25	1.66	2.04	0	0	59.21	57.64

TABLE V
CPU USAGE RESULT DURING DELETE TEST (SSL AND NO SSL)

Times (s)	%user		%nice		%system		%iowait		%steal		%idle	
	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL	No SSL	SSL
5	12.53	8.39	0	0	4.52	2.97	11.91	7.47	0	0	71.05	81.17
10	26.28	26.29	0	0	7.96	7.47	24.61	24.50	0	0	41.15	41.75
15	25.42	26.42	0	0	9.17	7.44	24.79	24.63	0	0	40.62	41.51
20	25.55	25.71	0	0	7.78	8.25	26.81	25.18	0	0	39.85	40.86
25	24.26	24.52	0	0	8.51	7.22	25.00	24.42	0	0	42.23	43.84
30	22.36	23.15	0.11	0	9.07	8.35	25.21	24.63	0	0	43.25	43.87
35	22.62	22.78	0	0	7.50	8.90	27.97	26.69	0	0	41.91	41.63
40	21.68	23.93	0	0	8.08	8.01	29.44	29.55	0	0	40.81	38.50
45	20.38	21.16	0	0	8.39	7.63	32.59	29.11	0	0	38.64	42.11
50	17.52	18.30	0	0	9.02	8.94	33.23	31.38	0	0	40.23	41.38
55	12.03	14.04	0	0	8.06	9.50	35.21	34.56	0	0	44.70	41.90
60	9.43	8.23	0	0	6.50	10.58	36.19	35.58	0	0	47.89	45.62
65	4.53	6.33	0	0	6.63	6.97	34.63	37.55	0	0	54.21	49.14
70	18.85	19.19	0.01	0	7.78	7.85	28.19	27.24	0	0	45.17	45.72

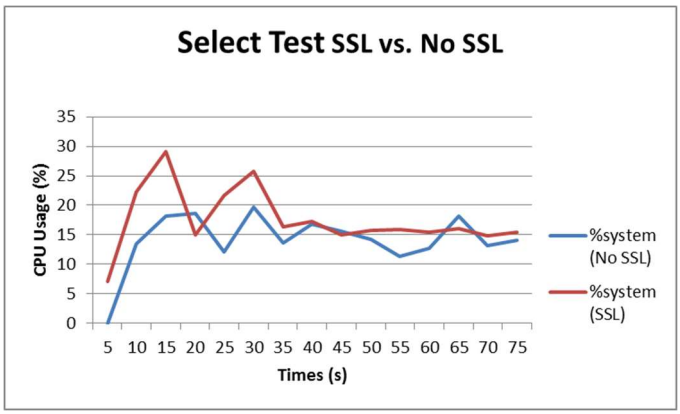
The average CPU utilization at the system level is recorded at 2.18% for the No SSL process and 1.26% for the SSL process.

Based on table V, it can be seen that the maximum value of %system in the delete query process that does not use SSL is at 9.17% at 15 seconds. While the maximum value at %system that uses SSL is at 10.58% at 60 seconds. For the average value - average on the process of deleting SSL and No SSL. For the delete process that does not use SSL, the average value obtained is 7.78% and the delete process that uses SSL, the average value obtained is 7.86%.

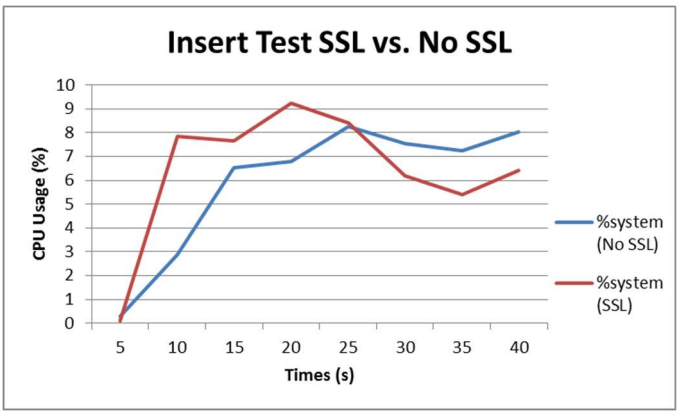
Based on these results, it was found that implementing SSL in the delete process can affect the needs of the %system process on the CPU. Even so, the average value shows that there is not too much a significant difference.

TABLE VI
RESPONSE TIME IN QUERY (SELECT, UPDATE, DELETE, INSERT)

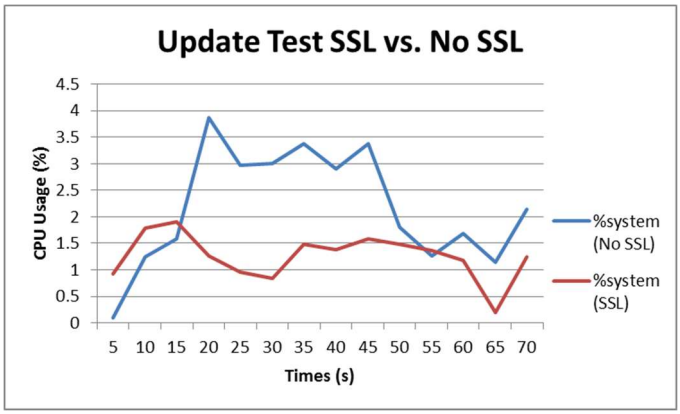
Query Mode	Average Response Time (s)	
	SSL Disable	SSL Required
INSERT	31.487	31.924
SELECT	61.164	68.137
UPDATE	55.381	56.788
DELETE	61.744	62.129



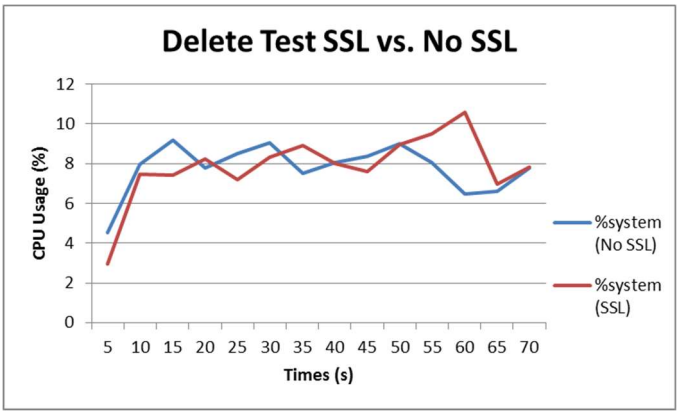
(a) Select Test



(b) Insert Test



(b) Update Test



(d) Delete Test

Figure 2. Comparison %system CPU Usage (a) Select, (b) Insert, (c) Update, (d) Delete Test

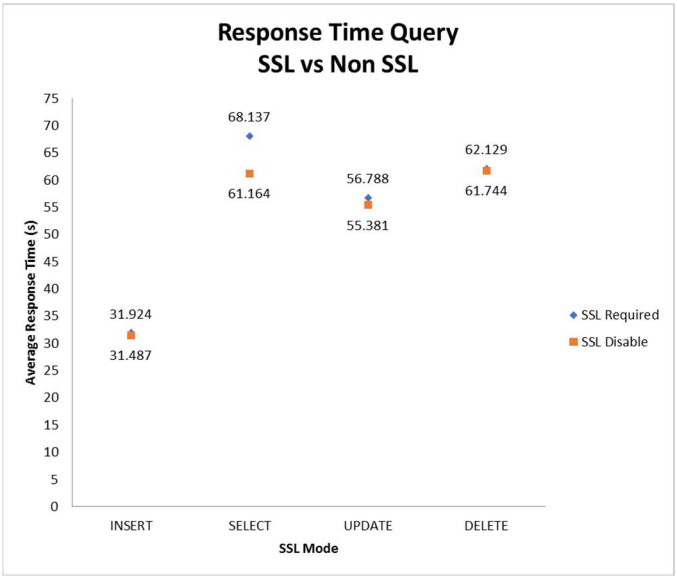


Figure 3. Comparison of SSL vs non-SSL of response time query.

Table VI and Fig. 3 compare the average time taken to perform various SQL queries with and without Secure Sockets Layer (SSL) enabled.

- INSERT: On average, performing an INSERT operation takes approximately 31.487 seconds without SSL. However, when SSL is enabled, this operation takes slightly longer, averaging around 31.924 seconds. This implies a slight increase in latency due to the overhead introduced by SSL, which can be attributed to the

additional encryption and decryption processes required for securing data transmission.

- SELECT: The difference in time for executing a SELECT operation is more pronounced. Without SSL, the average execution time is approximately 61.164 seconds. With SSL enabled, this operation takes an average of 68.137 seconds. The increase in latency can be due to the additional time required to secure data transmissions with SSL.
- UPDATE: For the UPDATE operation, the average time taken without SSL is around 55.381 seconds, while with SSL, the average time increases to 56.788 seconds. Like the other operations, this increase can be associated with the added overhead of SSL.
- DELETE: The DELETE operation shows a similar pattern. Without SSL, the operation takes an average time of 61.744 seconds. However, when SSL is turned on, the average time taken slightly increases to 62.129 seconds.

Overall, this data suggests that enabling SSL does lead to an increase in the average time taken to perform database operations. This increase in latency is likely due to the computational overhead associated with SSL's encryption and decryption processes. While this overhead may impact performance, it also significantly enhances the security of data transactions, an aspect critical in many application scenarios. Consequently, system designers and administrators must consider this trade-off between performance and security.

IV. CONCLUSION

In conclusion, as reliance on online data storage and communication continues to grow, so does the importance of securing this data. This research provides valuable insights into how SSL encryption. At the same time, it may affect performance, which is a critical component in maintaining the confidentiality, integrity, and availability of data in MySQL databases and enabling SSL results in longer average execution times for database operations. This latency is probably caused by the additional computational workload required for SSL's encryption and decryption procedures. Further research is recommended to investigate performance impacts under different network conditions using larger datasets and alternative encryption methods. It would also be interesting to evaluate the performance impacts of SSL on other types of databases or under different workloads.

ACKNOWLEDGEMENTS

We are thankful to Cyber Security Engineering Study Program, Politeknik Negeri Batam, for supporting the data capture using the CSOC lab computational platform.

REFERENCES

[1] E. Rescorla, "SSL and TLS: Design and Building Secure System." Addison Wesley Professional, USA, 2001.

[2] M. Gargiulo, "Council Post: Data Security Threats: What You Need To Know," Forbes. https://www.forbes.com/sites/forbestechcouncil/2022/05/16/data-security-threats-what-you-need-to-know/ (accessed Jun. 06, 2023).

[3] "Insights from 2023 Cyberthreat Defense Report," BrightTALK. https://www.brighttalk.com/service/player/en-US/theme/default/channel/12349/webcast/583264/standalone?commid=583264&reactPlayer=true&b=38861&embedUrl=https://www.imperva.com/learn/application-security/cyber-attack/ (accessed Jun. 06, 2023).

[4] B. Schneier, Applied Cryptography: Protocols, Algorithms and Source Code in C. John Wiley & Sons, 2017.

[5] D. Wagner and B. Schneier, "Analysis of the SSL 3.0 protocol," in Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2, in WOEC'96. USA: USENIX Association, Nov. 1996, p. 4.

[6] V. Beltran, J. Guitart, D. Carrera, J. Torres, E. Ayguade, and J. Labarta, Performance Impact of Using SSL on Dynamic Web Applications. 2004.

[7] B. Schneier, Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World. W. W. Norton & Company, 2015.

[8] A. Hoog, "Chapter 5 - Android device, data, and app security," in Android Forensics, A. Hoog, Ed., Boston: Syngress, 2011, pp. 159–194.

[9] C. Wolfe, "Securing Data in Transit using Two Channel Communication," Theses and Dissertations, Mar. 2018, [Online]. Available: https://scholar.afit.edu/etd/1828