International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Security Analysis of Authentication and Database on the Dock Management System Website against SQL Injection Attacks Using Penetration Testing Methods

Lutfi Febrianto[1]
[1]Department of Business Information Systems, Master Program in Information Systems Management, Gunadarma University, Jalan Kenari I, Kenari, Kec. Senen, City of Central Jakarta, DKI Jakarta, Indonesia- 10430
Email Address: [1]lutfi.febrianto @ gmail.com

Abstract**—** *The Dock Management System (DMS) is an application system designed to track the movement of trucks within the yard of a warehouse or manufacturing facility. There must be a good security system for this DMS application, so the DMS security system needs to be tested for its level of security. In web-based applications that use databases as data storage, one approach is to use SQL Injection, an exploitation technique. This research was using Structure Query Language (SQL) Injection to analyze the security level of a DMS Site and the application of Anti SQL Injection to prevent SQL Injection attacks. The test results show that SQL Injection can attack DMS using the specific syntax and the attack can be prevented using Anti SQL Injection by preventing the use of single quotation marks by the algorithm.*

Keywords**—** *DMS; SQL Injection; Management System; Havij.*

I. INTRODUCTION

DMS (Dock Management System) is a web-based information system for warehousing management that can help employees of company complete tasks such as receiving, putaway (storage), and picking).

They would enter a password where the password is private and confidential when someone logs in. Security problems are also a very critical problem since the internet is a public network that is linked to each other in a network and if the password entered by the user is not encrypted before being transmitted through the network to the server, it would be very dangerous. That's where sniffers can track user data or passwords.

In this study, the research goals to be accomplished are to get a set of SQL syntax that works to carry out attacks on an information system and to analyze the security system of DMS applications against SQL injection attacks.

II. THEORETICAL FRAMEWORK

The theories discussed in this chapter are those which support the understanding of the problem formulated in Chapter I.

A. Dock Management System

The Dock Management System (DMS) or warehousing management system functions as an application for data collection in the storage of production goods or production, resulting in a certain amount and time span, which is then distributed on request to the intended venue.

- Make it easier for warehouse managers to provide information to the production or delivery planning department on the availability of an item such that the availability of goods stays at a safe level.
- To facilitate the storage, selection and measurement of stock items, the placement of goods is determined by the system.
- Reducing the lead time from the tasks of storage and delivery of goods, as well as understanding the status of incoming and outgoing drivers.
- The availability of a warehouse gate makes it easier to evaluate to establish a more effective strategy for warehouse use.

B. Penetration Testing

Penetration testing is to test the security of a computer network or system by simulating an attack on a computer network or system. So later, you'll see security gaps that can be filled. Penetration testing has a variety of forms, tools, and methods. The penetration testing method for web-based applications are as follows:

1. *Passive Penetration Testing*

What is done in this case is to map and test the controls in the web application, login and setup, so that it is possible to map the target system.

2. *Active Penetration Testing*

Carrying out active system security testing activities by manipulating input, accessing and testing existing vulnerabilities.

3. *Aggressive Penetration Testing*

Exploiting vulnerabilities, reverse engineering software and systems, embedding backdoors, downloading code, trying to take over information on the server.

C. Structure Query Language Injection

By manipulating SQL syntax, SQL injection is a hacking technique. The SQL injection is performed by entering the commands used through the URL in the database. For instance, suppose there is a web-based source code application, as shown in Figure 1.

```
$SQL = select * from login where username = '$username'
and password = '$password' ;
```

Fig. 1. Example of source code for a web-based application.

Lutfi Febrianto, "Security Analysis of Authentication and Database on the Dock Management System Website Against SQL Injection Attacks Using Penetration Testing Methods," *International Research Journal of Advanced Engineering and Science*, Volume 6, Issue 1, pp. 199-204, 2021.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

With the GET or POST data transmission methods, queries like this can be injected by entering the string "or" = "on the loginform, so the results will be like this in Figure 2.

```
$SQL = select * from login where username = 'user' or "="
and password= 'pass' or "=" ;
```

Fig. 2. An example of using SQL Injection.

The selection results will always be Valid for this SQL Injection syntax so that it can penetrate the application.

SQL Injection is essentially a way to exploit security holes that exist at the level of the database or' layer' and its applications. It shows this security weakness when an attacker enters a string value and other example characters found in the SQL command. It is said to be a "injection" since the attacking operation is carried out to bypass the logical access rights filter on the intended website or computer system by "entering" a special string (character set).

```
' or ' 1 ' = ' 1          ' or 0 = 0 --
" or " 1 " = " 1          " or 0 = 0 --
' ) or ( ' 1 ' = '1        or 0=0 --
" ) or ( " 1 " = " 1      ' or 0 = 0 #
" #                       " or 0 = 0 #
```

Fig. 3. SQL String Injection Collection.

D. Havij

Havij is an application for "Automated SQL Injection" that can help perform penetration testing and exploit a web page's weaknesses in SQL Injection. This application can take advantage of vulnerabilities of insecure web applications, perform fingerprints of the back-end database, retrieve user and user password hashes, retrieve database data, execute SQL commands, and even access the underlying file system and execute operating system commands. In Figure 4, an example of Havij's representation is shown.

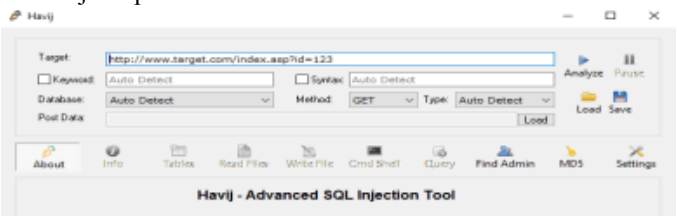

Fig. 4. Havij Display.

III. METHODS

The method used in this research is the research process of Penetration Testing. The research method of Penetration Testing was chosen because it focused specifically on the object to be analyzed in this analysis, namely the injection of databases on a website. This study begins with the injection of a compromised website to get sensitive information on the website. In this study, the stages to be carried out are:

A. Passive Penetration Testing

Mapping and testing the controls in the web application, login and configuration, so that the target system can be mapped.

B. Active Penetration Testing

Carrying out active system security testing activities by manipulating input, accessing and testing existing vulnerabilities.

C. Aggressive Penetration Testing

Exploiting vulnerabilities, reverse engineering software and systems, embedding backdoors, downloading code, trying to take over information on the server. In order to do this research we used:

TABLE I. Notebook Specifications (Client Computer).

S. No.	Label	Description
1	Platform	Notebook ASUS TUF FX504
2	Prosesor	Intel Core i5 8300H Processor
3	Sistem	Windows 10 Home
4	Memori	8 GB DDR4 2666MHz SDRAM
5	Grafis	NVIDIA GeForce GTX 1060
6	Storage	1TB SATA HDD & 128GB SSD
7	Networking	Integrated Wi-Fi 5 (802.11 ac (2x2))
8	Interface	USB 3.0 port(s), LAN jack, HDMI

TABLE II. Mobile Specifications (Hotspot).

S. No.	Label	Description
1	Platform	Handphone Redmi Note 6 Pro
2	Prosesor	Prosesor Qualcomm Snapdragon 636
3	Sistem	Android Pie
4	Memori	4 GB RAM LPDDR4X
5	Grafis	GPU Adreno 509
6	Storage	64 GB ROM eMMC 5.1
7	Networking	Wi-Fi Direct, Wi-Fi, 802.11 a/b/g/n/ac
8	Interface	Audio Jack, Speaker/Mikrofon, Port USB

IV. RESULT AND DISCUSSION

This test has 2 different scenarios, namely manual SQL Injection testing and automatic SQL Injection testing. The explanation are:

A. SQL Injection Manual Scenario

Testing will be conducted three times in this manual SQL Injection test scenario, including the first test of the DMS web will be injected into the URL, the second test of the DMS web will be injected into the login form, and the third test is a repeat of the first and second tests, however DMS has been provided with security.

1. *Injection on URL*

In general, the method used to inject in URLs is to use single quotes (`) at the very back of the address in the URL. The modified DMS page is shown in Figure 5.

```
https://dms.azhaclient.com/dms/login`
```

Fig. 5. An unmodified address.

After that a message will appear as Figure 6.

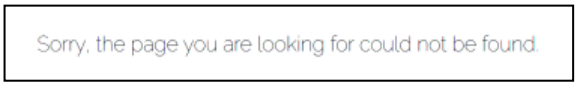

Fig. 6. An unmodified address response.

This implies that the web is a vulnerability if an error happens, so the initial suspicion is that the DMS application

Lutfi Febrianto, "Security Analysis of Authentication and Database on the Dock Management System Website Against SQL Injection Attacks Using Penetration Testing Methods," *International Research Journal of Advanced Engineering and Science*, Volume 6, Issue 1, pp. 199-204, 2021.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

has a weakness in its URL. Then, computing the database tables in the DMS is the next step. Based on the Indonesian hacker forum, the method used is to alter the URL address with the command ORDER BY, as follows. Figure 7 shows the modified DMS page.

https://dms.azhaclient.com/dms/login+order+by+1

Fig. 7. An unmodified address

After that a message will appear as Figure 8.

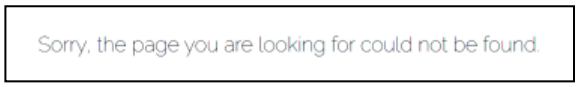

Fig. 8. An unmodified address response.

This informs that the DMS application at the address http://dms.azhaclient.com/dms/login does not have database operations, so in other words the URL of the DMS application cannot be injected because it is safe.

2. *Injection in Login Form*

Before testing, please note that DMS is a web-based application that has 2 types of users, namely security and admin.

TABLE III. Table username on DMS web

S. No.	Username	Group Name
1	security-1	Security
2	ruma-admin-1	Admin Ruma

In the previous chapter, we explained a bit about the SQL Injection Syntax as shown in Figure 3. This syntax will be entered on the DMS web username form along with the username that is already known, so if you want to use this syntax effectively, you must at least know the username. It shows an example of the input of the SQL injection string to the username in Figure 9.

Fig. 9. SQL injection process using security-1 syntax 'or' 1 '=' 1.

TABLE IV. The results of testing the DMS web application username and SQL Injection syntax

S. No.	username	group	syntax	output
1	security-1	Security	‘or‘1’=‘1	succeed
2	security-1	Security	‘ #	succeed
3	security-1	Security	‘ or 0=0 #	succeed
4	ruma-admin-1	Admin	‘or‘1’=‘1	succeed
5	ruma-admin-1	Admin	‘ #	succeed
6	ruma-admin-1	Admin	‘ or 0=0 #	succeed

Information:

Username : Username used
Group : The username used belongs to security or admin?
Syntax : The syntax used to inject in the login form
Result : Does it work means that the DMS web can be penetrated.
Web Status : If successfully injected, the DMS display will appear whether the display is for security or the display for admin?

Based on the test results table above, we can see it that of the 10 SQL Injection syntax used, only 3 syntax were successful, namely syntax ‘or `1’ =` 1; ‘or 0 = 0 #; and ‘#. From these 3 syntaxes, all usernames used together with these 3 syntaxes are successfully used to inject the web. In DMS, the query algorithm used to check username and password is as shown in Figure 10 below.

SELECT a.userid as userid, a.idgroup as idgroup, a.password as password, b.nama as agroup from tbuser a, tb_group b where a.userid = ".$_POSTruserl." and a.password = ".$_POSTrpassl." and a.idgroup=b.idgroup and sign=0"

Fig. 10. The query algorithm used to check the DMS web username and password.

In the question section above, there is a section where the username and password are checked as shown in Figure 10. When the username value is entered in the source code “_POSTruserl.” and the password entered the source code “_POSTrpassl.”

Each user and password as stated must be TRUE, since it uses the AND feature, where if one user or passwords is FALSE, the result of the query will be FALSE. If the result of the question is FALSE, the DMS system can read that the user is not registered in the DMS database and has no right to access the DMS.

On a successful SQL Injection test, the syntax used interferes with the AND function. If the wrong syntax that successfully injects above is entered into the username form, then it will manipulate the query as shown in Figure 11.

where a.userid = ‘security-1’ or ‘1’=’1’ and a.password = ‘’

Fig. 11. The query manipulated by 'or' 1 '=' 1

So that the value of the query is shown in Figure 12.

TRUE or **TRUE** and **FALSE** = **TRUE**

Fig. 12. The manipulated query value

3. *Provides Anti SQL Injection Security*

Web DMS, after being given Anti-SQL Injection security in the login form, apparently prevented the SQL Injection process from occurring when the login process occurred. This is because of the Anti SQL Injection algorithm that prohibits single quotation marks (`) from being used in the username form.

TABLE V. The test results of the DMS web application and SQL Injection syntax after being secured

S. No.	username	group	syntax	output
1	security-1	Security	‘or‘1’=‘1	failed
2	security-1	Security	‘ #	failed
3	security-1	Security	‘ or 0=0 #	failed
4	ruma-admin-1	Admin	‘or‘1’=‘1	failed
5	ruma-admin-1	Admin	‘ #	failed
6	ruma-admin-1	Admin	‘ or 0=0 #	failed

Lutfi Febrianto, “Security Analysis of Authentication and Database on the Dock Management System Website Against SQL Injection Attacks Using Penetration Testing Methods,” *International Research Journal of Advanced Engineering and Science*, Volume 6, Issue 1, pp. 199-204, 2021.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

It was previously clarified that a syntax with a single quotation mark (‘) character or string is used by the SQL Injection process because it is this one quotation mark that can manipulate queries that search web usernames and passwords. quotation mark is used during the login process as a username. See the algorithm in Figure 13 for more information.

```
$userinject = $_POST[‘user’];
// anti injection code

    for ($i = 0; $i < strlen($userinject); $i++) {
    /* check one by one the string elements */

        if($userinject{$i} == “’”){
        /* whether the I-character is a single quotation mark */

            $error = “Login failed username/password not match,
            (‘) is forbidden”;
            header(“Location:
            {{url}}/dms/login”.urlencode($error));
            /*redirect to the login form page with an error message
     */

            exit(0);
        }
} // end anti injection code
```

Fig. 13. Anti SQL Injection Algorithm

Based on the above algorithm, that Anti SQL Injection is used to check the characters entered the form of a username. The algorithm will check the characters entered one by one when the username character is entered into the username form. If a single quote character (‘) is found during checking, then the login operation will not run and will return to the form page.

B. Automated SQL Injection Scenarios

The SQL Injection process uses Havij software as the SQL Injection tool in an automated SQL Injection scenario. In stark contrast to the SQL Injection manual process, where the client only needs to enter the software, the target web address.

On the DMS site and on the dummy web, there are 2 types of testing, namely testing tools. Use SQL Injection if the web DMS is safe and cannot be hacked. Web dummy is a web test that uses SQL injection tools to assess hacking websites.

1. *Web DMS testing*

The SQL Injection process is carried out by first opening the Havij software as shown in Figure 14 below.

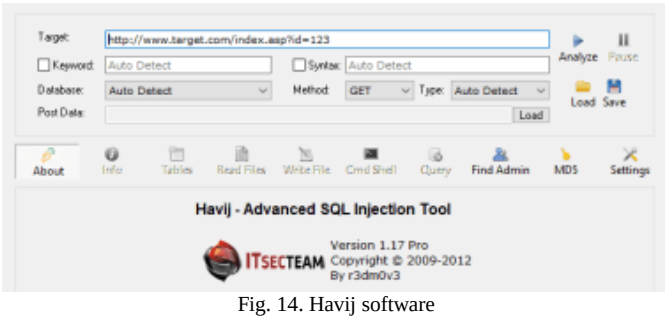

Fig. 14. Havij software

Then you enter the web address that is the SQL Injection target. The web address of the DMS is http://dms.azhaclient.com/ where the page is redirected to the http://dms.azhaclient.com/dms/login page. As in Figure 15, the http://dms.azhaclient.com/dms/login page will be entered into the target form of the Havij software.

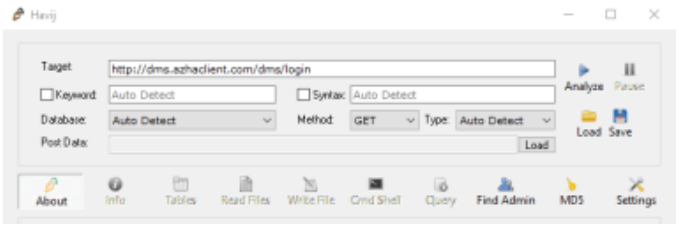

Fig. 15. DMS address entered in Havij

Then after that look at the analysis produced by Havij in injecting the DMS web as in Figure 16.

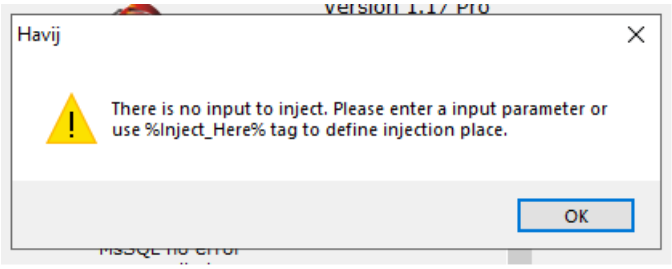

Fig. 16. No input parameter error

It turned out that there was an error when the DMS web wanted to be injected, based on the web address entered the target, the address did not have any input variables or parameters which caused Havij to not be able to inject the DMS web. However, this can be overcome by writing “% Inject Here%” at the back of the target web address, so that the address entered Havij is http://dms.azhaclient.com/dms/login%Inject_Here%. Then we can see the results of Havij in injecting the DMS web, where the results are as in Figure 17 below.

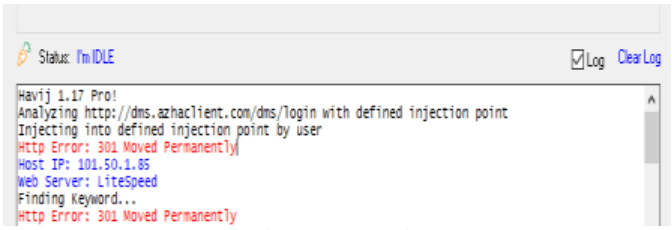

Fig. 17. Error Havij failed to successfully inject web DMS

Based on the results above, the SQL Injection process failed. In this result, Havij questioned the existence of the input parameters used and also the methods used. This will be discussed in the analysis section.

2. *Web dummy testing*

Now what will be tested is the dummy web, where the dummy web has a homepage as shown in Figure 18.

Lutfi Febrianto, “Security Analysis of Authentication and Database on the Dock Management System Website Against SQL Injection Attacks Using Penetration Testing Methods,” *International Research Journal of Advanced Engineering and Science*, Volume 6, Issue 1, pp. 199-204, 2021.

International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Fig. 18. Web dummy's homepage

Testing on this dummy web uses the same method as the DMS web. The dummy web address is http: //localhost/diary/index.php where the address has a login form with a URL request http: //localhost/diary/login.php? Username = lutfi.

After the address is entered, the results of the SQL Injection are as shown in Figure 19 below.

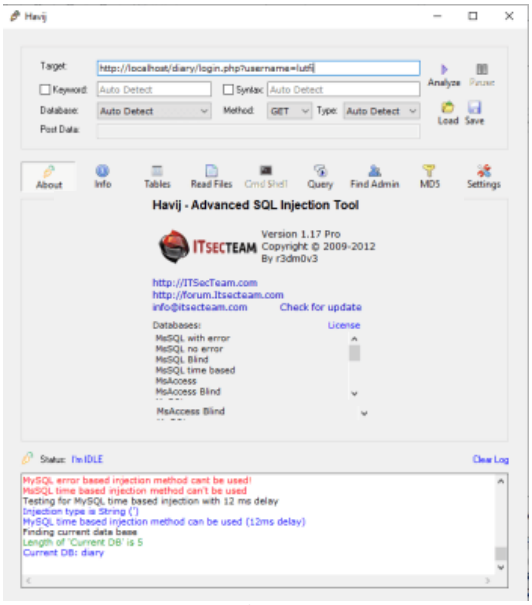

Fig. 19. Havij successfully injected the local web

The SQL Injection process was successful, where the database was called "diary". Havij software can view the data in the database that has been successfully obtained. As an example in Figure 20, this dummy web database called "diary" has 15 tables.

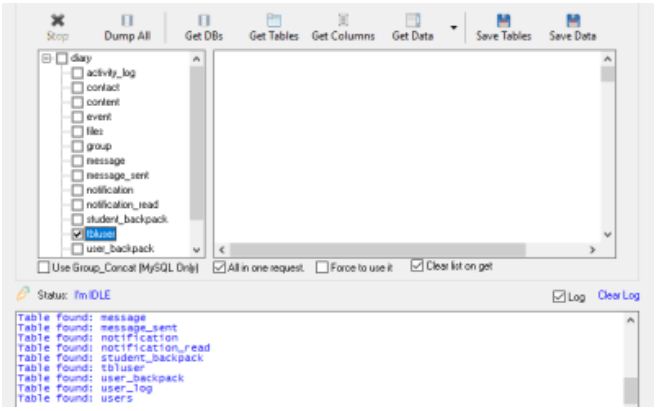

Fig. 20. Diary database table

From this table, the most important table is the "tbluser" table which contains user data from the dummy web. When the table is opened, the results will have columns as in Figure 21.

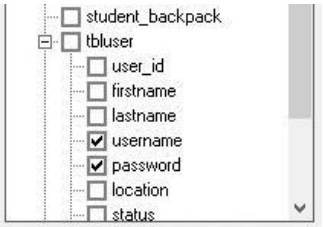

Fig. 21. Column in the "tbluser table"

Based on the "tbluser" column above, there are the most important data, namely "username" and "password". When the data is opened, the results are as in Figure 22.

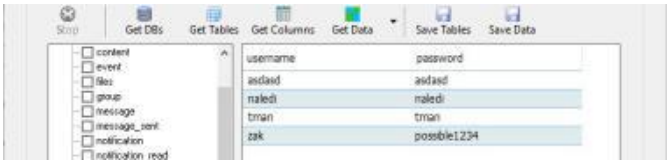

Fig. 22. Data "username" and "password"

Based on the data above, the usernames and passwords registered in the web dummy are "asdasd, asdasd"; "naledi, naledi"; "tman, tman"; and "zak, possible1234". So that based on the tests that have been done, it can be concluded that it proves this web dummy to be injectable using SQL Injection via the Havij software.

3. *Automated test analysis*

The SQL injection process carried out using Havij software has certain conditions, namely the target web address. The target address must have a format like this http://www.target.com/index.asp?id=123. The input here means the variable and value of the address, for example, id = 123 where id is the variable and 123 is the value of the id variable.

Web DMS cannot be injected because the form of DMS address that is targeted differs from the form of address that can be injected by Havij. The target DMS web address has no variables and values because it is hidden by the DMS web itself, because DMS itself uses the POST method. As explained in the manual SQL injection analysis section, DMS uses the POST method, which is different from the GET method. The POST method hides the variables that will be sent to the web server, while the GET method does not hide these variables and is visible in the URL of the web address. By default, the target format on Havij is an address that uses the GET method, so that the address must have a variable and value like http: //localhost/diary/login.php? Username = lutfi, this causes a web dummy to be successfully injected.

Havij can also inject web addresses using the POST method. But if you use the POST method, the attacker must enter the post data containing the variables and values of the web address manually as shown in Figure 23.

Lutfi Febrianto, "Security Analysis of Authentication and Database on the Dock Management System Website Against SQL Injection Attacks Using Penetration Testing Methods," *International Research Journal of Advanced Engineering and Science*, Volume 6, Issue 1, pp. 199-204, 2021.

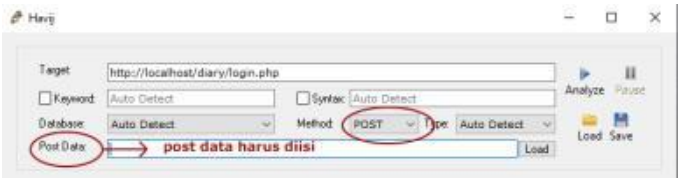

Fig. 23. SQL Injection using POST method on Havij

The problem is that, whether the attacker has a way to see the original web source code on the web server or just by guessing certain variables and values, the attacker does not know the variables and values that must be entered in the post data sector. This is the first explanation why Havij cannot inject the DMS web.

In the dummy web test, when the dummy address is injected it displays an error message 301 moved permanently or permanently moved, so the original address http://dms.azhaclient.com/dms/login was moved to https://dms.azhaclient.com/ dms / login. The problem is that havij does not allow clients to scan or run SQL Injection queries on sites using SSL (https: //). This is the second reason that explains that web DMS cannot be injected using havij, where havij cannot inject addresses using https: // SSL because data sent using https: // is secured through the Transport Layer Security (TLS) protocol, which provides three layers of protection. The keys are: Encryption, Data Integrity, and Authentication.

V. CONCLUSION

The Web Dock Management System (DMS) is an application system designed to monitor truck's movement in a manufacturing facility or warehouse yard. Before giving Anti SQL Injection) the web can be injected into the user id form on the login form page using the syntax 'or' 1 '=' 1; "Or 0 = 0 #; and #. After being given Anti SQL Injection Web security, it cannot be injected again using manual methods or entering queries in the url or using Havij's Automated SQL Injection software, because the Anti SQL Injection algorithm prohibits and prevents the use of the single quotation mark string (`) used to inject the web DMS on the form user id and uses DMS using the MD5 hash algorithm which functions to randomize the password before entering the database query.

REFERENCES

[1] P. R. Utami. "Analisis Forensik Jaringan Studi Kasus Serangan SQL Injection pada Server Universitas Gadjah Mada," *Indonesian Journal of Computing and Cybernetics Systems*, vol. 6, issue 2, pp. 101-112, 2012.

[2] Dewi, E. Kusuma., S.N. Azhari. "Analisis Keamanan Sistem Perangkat Lunak," *Universitas Gajah Mada*, ISSN: 1907-5022, 2012.

[3] K. Sanskriti. "Detection and Prevention of SQL Injection Attacks in Web Applications," ISSN: 2393-9915, 2016.

[4] K. Yuliana., S. Jacky., W. Shinta. "Analisis Sistem Manajemen Pergudangan pada CV. Sulawesi Pratama Manado," *Universitas Sam Ratulangi Manado*, vol. 5, issue 2, ISSN: 2303-1174, 2017.

[5] Nazwita., R. Siti. "Analisis Sistem Keamanan Web Server dan Database Server Menggunakan Suricata," *UIN Sultan Syarif Kasim Riau*, ISSN: 2579-5406, 2017.

[6] Y. Utami., A. Nugroho., A.F. Wijaya "Perencanaan Strategis Sistem Informasi Dan Teknologi Informasi Pada Dinas Perindustrian Dan Tenaga Kerja Kota Salatiga," *Universitas Kristen Satya Wacana*, vol. 5, issue 3, ISSN: 2528-6579, 2017.

[7] K. D. Marisa, "Analisis Keamanan Sistem Login," *Universitas Mulawarman*, vol. 6, No 2, 2011.

[8] B.A. Nugroho, "Analisis Keamanan Jaringan Pada Fasilitas Internet(Wifi) Terhadap Serangan Packet Sniffing," *Universitas Muhammadiyah Surakarta*, 2012.