The Scientific Journal of Cihan University – Sulaimaniya **PP: 219-229**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

Prevent the SQL Injection base on Session using the static ID of retrieving the URL

Brzu T. Mohammed[1], Ardallan H. Awlla[2], Sherko H. Murad[3], Hawar H. Yaba[4]

[1,3,4] Computer Science Department, Kurdistan Technical Institute, Sulaimani, Iraq
[2]Department of Computer Science, Cihan University -Sulaimaniya, Sulaymaniya, Iraq

Email: brzu.tahir@kti.edu.iq[1], ardalana.husain@sulicihan.edu.krd[2], sherko.murad@kti.edu.iq[3], hawar.yaba@ kti.edu.iq[4]

Abstract:

Today, the rise in cyber threats has underscored the vulnerability of web applications, making website security a continuous challenge. Structured Query Language (SQL) injection attacks are among the top ten security vulnerabilities recognized by the Open Web Application Security Project (OWASP). Structured Query Language injection is still the most typical vulnerability and the most critical security threat due to the diversity of forms and dramatic changes that it could lead to, including financial losses, data leaks, and serious database corruption that could paralyze a site. One vulnerability in a web application is sending sensitive data through the Uniform Resource Locator (URL) query string. Therefore, the Uniform Resource Locator query string can be a trap for Structured Query Language injection attacks to steal user data. This paper proposes an solution based on a session using the static identifier of retrieving the Uniform Resource Locator to prevent Structured Query Language injection vulnerabilities.

Keywords: SQL injection, SQL injection attack, Query parameters, Session, Database Security

الملخص:

اليوم، أصبحت الزيادة في التهديدات الإلكترونية توضح ضعف التطبيقات الويب، مما يجعل أمان المواقع تحديًا مستمرًا. تُعد هجمات SQL injection واحدة من أهم عشر ثغرات أمنية تعترف بها منظمة أمن تطبيقات الويب المفتوحة (OWASP) لا تزال SQL Injection هي الثغرة الأكثر شيوعًا والتهديد الأمني الأكثر خطورة بسبب تنوع الأشكال والتغييرات الدراماتيكية التي يمكن أن تؤدي إليها، بما في ذلك الخسائر المالية، وتسرب البيانات، وتلف قواعد البيانات الخطير الذي يمكن أن يشل الموقع. إحدى الثغرات في تطبيق الويب هي إرسال البيانات الحساسة عبر Uniform Resource Locator (URL) Query String لذلك، يمكن أن تكون URL Query String فخًا لهجمات SQL Injection لسرقة بيانات المستخدمين.هذا البحث تقترح حلاً يعتمد على استخدام معرف ثابت لاسترجاع URL عبر session لمنع ثغرات SQL Injection.

الكلمات المفتاحية: حقن SQL، هجوم حقن SQL، معلمات الاستعلام، الجلسة، أمان قاعدة البيانات.

DOI: http://dx.doi.org/10.25098/8.1.32

The Scientific Journal of Cihan University – Sulaimaniya **PP: 219-229**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

پوختە:

ئەمڕۆ زیادبوونی هەڕەشە ئەلیکترۆنییەکان لاوازی بەرنامەکانی وێب نیشان دەدات، ئەمەش وایکردووە ئاسایشی ماڵپەڕەکان ببێتە تەحەدایەکی بەردەوام. هێرشی SQL Injection یەکێکە لە دە لاوازییە ئەمنییەکانی سەرەکی کە لەلایەن ڕێکخراوی ئاسایشی بەرنامەی وێبی کراوە (OWASP) ناسراوەتەوە. SQL Injection وەك باوترین لاوازی و جددیترین هەڕەشەی ئاسایش دەمێنێتەوە بەهۆی جۆراوجۆری فۆرم و گۆڕانکارییە گەورەکان کە دەتوانێت ببێتە هۆی، لەوانەش زیانەکانی دارایی، دزەپێکردنی زانیاری و خراپبوونی جددی بنکەی زانیارییەکان کە دەتوانێت ماڵپەڕەکە ئیفلیج بکات. یەکێك لە لاوازییەکانی ناو بەرنامەی وێب ناردنی زانیارییە هەستیارەکانە لە ڕێگەی Uniform Resource Locator (URL) Query String لە بەرئەوە URL Query String دەتوانێت ببێتە تەڵە بۆ هێرشی SQL Injection بۆ دزینی زانیاری بەکارهێنەران ئەم توێژینەوەیە چارەسەرێك پێشنیار دەکات پشت بە بەکارهێنانی ناسێنەرێکی جێگیر دەبەستێت بۆ وەرگرتنەوەی URL لە ڕێگەی Session بۆ ڕێگریکردن لە لاوازییەکانیSQL Injection.

1. Introduction

Today's technology-driven world heavily relies on website services for a variety of activities, such as online shopping, banking, and socializing. However, websites that use databases to store sensitive information [1], such as financial data, biometric data, and passwords, are prime targets for hackers [2].

SQL injection attacks are a significant internet security threat. In 2017, a Russian hacker named "Rasputin" exploited this vulnerability to access data from 60+ institutions in the United Kingdom and the United States, as confirmed by the Federal Bureau of Investigation (FBI) and the Department of Homeland Security (DHS). Similarly, in 2018, the Cisco Prime License Manager suffered from a SQL injection flaw, enabling attackers to manipulate database information. These incidents highlight the widespread impact and serious consequences of SQL injection vulnerabilities.

Despite ongoing efforts to enhance internet security, the dangers to the internet persist and escalate due to the ever-growing global population of internet users [3].

As per OWASP, a SQL injection attack occurs by injecting a SQL query through client input data into a program. Successfully executing this attack allows access to sensitive database information, enabling actions like data insertion, modification, and deletion. Additionally, it grants control over database management tasks, such as shutting down the database management system and restoring the contents of certain files that exist in the database management file system Fig.1.

In practice, many database websites face the threat of SQL injection attacks because regular users and SQL injection attacks are not separated from each other when accessing a system. Attackers can execute SQL injections conveniently through query strings or web forms, making their actions harder to detect. Despite the existing web application firewall's reliance on feature matching algorithms, it may fall short in safeguarding against all SQL injection variants.

Developers commonly utilize the URL query string as a primary means to transmit data across web application pages. Unfortunately, some developers inadvertently send sensitive information through these URLs, inadvertently creating vulnerabilities that attackers exploit through SQL injection techniques.

DOI: http://dx.doi.org/10.25098/8.1.32

The Scientific Journal of Cihan University – Sulaimaniya **PP: 219-229**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

To prevent SQL injection attacks, it is crucial for website developers to follow secure coding practices, such as using parameterized queries, input validation, and proper error handling. While web application firewalls can provide some protection, they are not foolproof and may not catch all types of attacks. This paper suggests an approach: substituting the query string with a session-based mechanism as a preventive measure against SQL injection attacks. This method aims to bolster security by altering the way data is handled and transmitted within sessions, potentially mitigating vulnerabilities posed by query strings susceptible to exploitation.

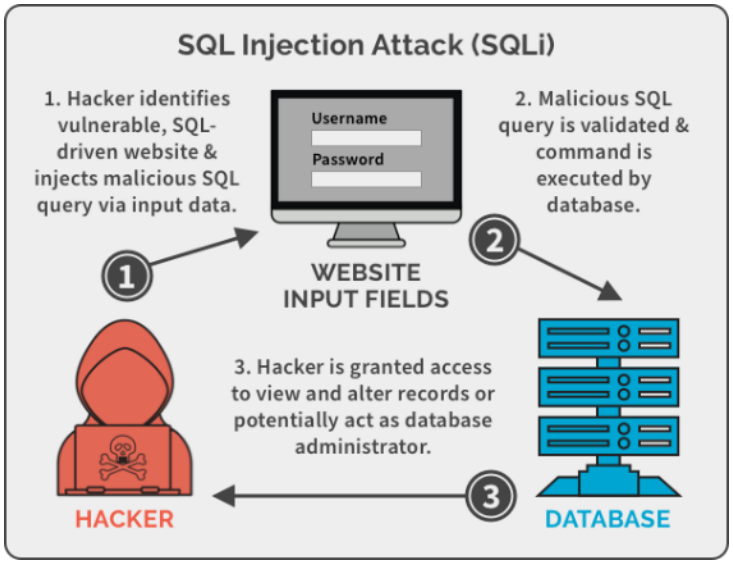

Fig.1. SQL-injection-attack

2. Literature review

In Tang, et al. [4] SQL injection a popular web attack, is a difficult problem for network security, that has resulted in annual financial losses of millions of dollars in addition the disclosure of a considerable quantity of users' personal information. This research proposed a high accuracy SQL injection detection solution based on neural network, Consequently, this paper's security achievement is 99.9 percent. Li, et al [5] research identifying SQL injection in the field of network security is a challenge problem, while traditional machine learning-based methods are difficult to handle multiple features and redundant features, whereas deep learning methods contain multiple hyper-parameters and are prone to over-fitting. As a result, this paper proposed an adaptive deep forest model-based SQL injection detection approach, this research compares the suggested strategy to traditional machine learning techniques in terms of accuracy, precision, recall, and f1-score. The outcomes demonstrate that the strategy was more effective.

Hasan, et.al. [6] extracted a number of features from SQL query and analyzed them to see if they are injected with harmful commands. The suggested system sits between both the database application and the database management system, examining the flow of queries and determining whether or not they should be passed to the database.

The article evaluated the performance of 23 various classifiers. The top five models based on the accuracy are chosen from among them to build the suggested system. The suggested method has been

DOI: http://dx.doi.org/10.25098/8.1.32

thoroughly evaluated, and the findings show that Ensemble Boosted and Bagged Trees classifiers are the most accurate with 93.8 percent, the results show that the approach is successful with 99,23%

Xie, et.al. [7] Generally matching is the most common method for detecting SQL injection. This kind of regular technique has a high rate of detection and accuracy, but it is incapable of detecting new threats. It is certain that new bypass techniques will came out to escape rules such as URL multi-encoding. This paper proposed Elastic-Pooling CNN which is automatically determines SQL Injection characteristics and that makes it Identifies SQL injection in web applications efficiently, the proposed method has a 99.9320% accuracy rate. It can detect new threats and is more difficult to defeat because of the irregular matching features. Alenezi, et.al. [8] provided a comprehensive overview of existing methods for preventing SQL injection threat. This study demonstrated that there is no one solution that can fully guard against SQL injection attacks; consequently, more research is necessary to combine several static and runtime techniques to achieve the best possible security with the least amount of computational power.

Abikoye, et.al. [9] The Knuth-Morris-Pratt clustering algorithm is used to identify and prevent certain threats. for the purpose of detect such malicious code, the technique was used to compare the user's input string with the stored pattern of the injection string. The PHP scripting language and the Apache XAMPP Server were used in the implementation. Various testing cases of SQL injection, cross-site scripting (XSS), and encoded injection attacks were used to assess the technique's security. The test results demonstrated that the technique was capable of detecting and preventing attacks, logging the attack item in the database, blocking a system using its Mac Address to thwart subsequent assaults, and sending a blocked message.

Jang, et.al.[10] proposed code generation method for detecting the embedded SQL injection vulnerability in the C/C++ host programming language. The suggested technique has the advantage of being beneficial in increasing software security monitoring, allowing us to design successful remediation methods to ensure security applications and reduce errors. Furthermore, this paper methodology demonstrated that successful retrofitting strategies can be designed to provide security in legacy applications while also removing well-known attackers. In addition, this paper offered a simple case study to demonstrate how SQL injection may be detected in embedded SQL.

Falor.et al. [11] investigated the various strategies for detecting and blocking SQL injection attacks. All sorts of SQLi attack queries, and also queries used to target specific databases, were included in a systematic dataset. Then they analyzed and compared the performance of five various classification models. They discovered that CNN has the best outcomes.

According to Nanhay, et.al.[12] there were a few ways to prevent SQL injection, including minimizing privileges, implementing consistent coding standards, and SQL server firewalling. Decreasing privileges means prioritizing security, and appropriate steps must be taken during the development stage. Implementing consistent coding standards requires developers to establish some coding policies to ensure that input validations checks are performed on the server, making it more secure. SQL firewalls are necessary to ensure that only trusted clients can be contacted. The firewall should reject all untrusted traffic.

DOI: http://dx.doi.org/10.25098/8.1.32

The Scientific Journal of Cihan University – Sulaimaniya **PP: 219-229**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

In Vamshi, et.al [13] there were three prevention methodologies identified the first method is referred to as processing inputs. SQL injection is carried out using keywords such as 'FROM', 'WHERE', and 'SELECT'. This problem can be solved if the keywords are not accepted in the input fields. The second method is to manage permissions so that only people with database authorization can access the data.

In Krit, et.al. [14,] they discussed the vulnerability of SQL injection and proposed a framework known as "PhpMinerl" for SQL injection. In addition, a novel method for detecting SQL injection attacks based on removing the SQL query attributes values is presented. They had devised a method for removing SQL query attributes. Nonetheless, before detecting the SQL injection, this method cannot justify the SQL syntax. Furthermore, this paper discusses Microsoft Azure Machine Language, a cloud-based predictive service that offers fully managed model predictive analytics and predictive models.

In Raja, et.al [15] the DUD approach was used to detect SQL injection. The DUD approach is a post-processing approach that is based on query classification. This approach is entirely dependent on the user, who must be defined prior to the algorithm's execution. This DUD approach is then improved by comparing the run time of SQL statements with the sanitizers to verify the attacks.

In Rhythm, et.al [16] are discussed more SQL injection prevention techniques, such as black box testing, Black box testing improves the testing system that has been infiltrated by the use of machine learning approaches.

3. Methodology

Table 1. Qualitative assessment of the relative risk levels.

Threat	Potential Impact	Risk Level (0-100%)
SQL Injection	Unauthorized data access and manipulation	80
Cross-Site Scripting	Theft of session data, disclosure of data	60
Cross-Site Request Forgery	Unauthorized actions on behalf of users	50
Data Breach	Unauthorized access to sensitive data	90
Phishing Attacks	Disclosure of login credentials	60
Man-in-the-Middle Attacks	Eavesdropping on sensitive data	50
Malware Attacks	Theft of sensitive information, disruption	85

According to OWASP the table 1. Presents a qualitative assessment of the relative risk levels associated with various security threats. Each threat is evaluated based on its potential impact and assigned a risk level on a scale from 0% to 100%, where higher percentages indicate a higher perceived risk. The risks are characterized in terms of potential consequences, such as unauthorized data access, session theft, and unauthorized actions.

While these risk levels are subjective and illustrative, they provide a comparative perspective on the potential severity of each threat. the risk level for SQL Injection is given as 80%. This indicates a relatively high perceived risk associated with SQL Injection compared to the other threats listed in the table.

DOI: http://dx.doi.org/10.25098/8.1.32

The Scientific Journal of Cihan University – Sulaimaniya **PP: 219-229**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

3.1. Query String

A query string is a part of a URL that contains data used by web applications to interact with servers. It usually follows a question mark (?) in the URL and consists of key-value pairs separated by ampersands (&). For example, in the URL "https://example.com/search?q=hello&page=1":

- The query string starts after the question mark?
- q=hello is a key-value pair where q is the key and hello is the associated value.
- page=1 is another key-value pair.

Query strings often facilitate passing information between different pages on a website. They can contain various types of data, such as search terms, user preferences, session identifiers, and more. However, sensitive information should not be included in query strings as they are visible in the URL and can potentially be intercepted or exposed.

3.2. Session

Session management in web applications is crucial for maintaining user authentication and tracking their activity. session rely on HTTP protocol [17], as HTTP is stateless, it doesn't retain user data between requests. To overcome this, servers generate unique session IDs for each user, which are transmitted between the client's browser and the server. These IDs are stored by the server and serve as identifiers for individual user sessions [18]. There are three primary methods to maintain sessions:

- Cookies,
- HTML hidden form fields, and
- URL arguments.

For instance, when a user interacts with a web application, such as clicking a URL link, a URL query string is formed. This query string contains parameters that request specific information from the server. The server uses these parameters to retrieve and provide the requested data, allowing the web application to maintain user sessions and serve information based on these parameters Fig.3.

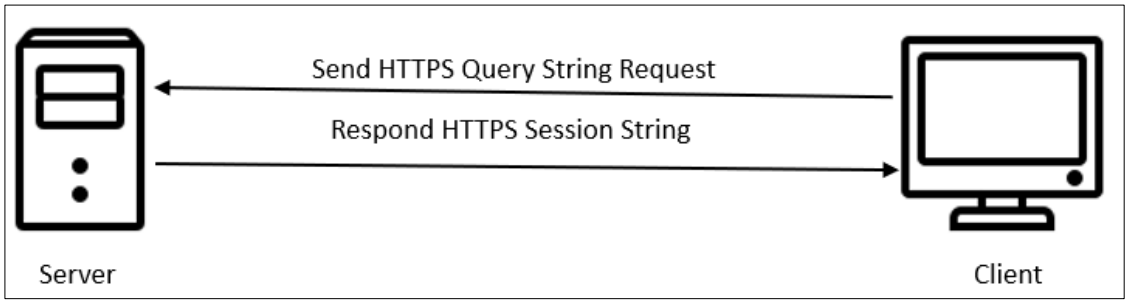

Fig.3. Client and server side

DOI: http://dx.doi.org/10.25098/8.1.32

The Scientific Journal of Cihan University – Sulaimaniya **PP: 219-229**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

When a query string is used to fetch data from a database, hackers can exploit this by manipulating the query string value (ID) to launch attacks. To showcase this vulnerability, a website was developed and hosted on a local server under the domain http://localhost/news/example.php?id=1. Testing this website for susceptibility to SQL injection attacks using the SQL Injection Map tool. It demonstrated that the website is prone to such attacks, as depicted in Fig.4. This underscores the importance of implementing robust security measures to safeguard against SQL injection threats.

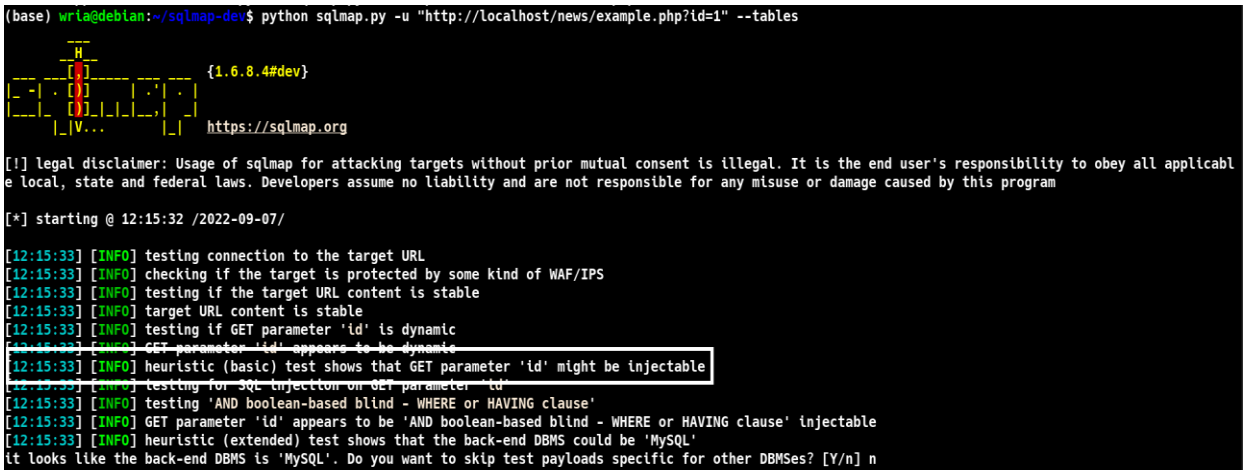

Fig.4. SQL Map Injection Attacking Queries

To addressing this security vulnerability involved a two-step approach. Initially, the query ID was extracted from the URL query string before retrieving data from the server's database. This step aimed to decouple direct database access from the ID provided in the URL, mitigating potential vulnerabilities.

Second, the extracted ID was replaced with a predefined value, ensuring database queries relied on a controlled and secure identifier.

The implementation of these steps, detailed in pseudocode and depicted in Fig. 5, aimed to showcase a practical solution to mitigate the risk of SQL injection by securing the query string handling process.

DOI: http://dx.doi.org/10.25098/8.1.32

The Scientific Journal of Cihan University – Sulaimaniya **PP: 219-229**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

```
Proposed Pseudocode:
Class main
Begin
  initialize array characters(i,j);
  initialize string querystring = URL;
  initialize string getID = null;

  if (Request.queryString("ID") != null)
  Begin
    getID = Request.queryString("ID");
    getID = characters(1, j);
    session[getID];
  End
  Else
  Begin
    Return error page 404;
  End
End
```

Fig.5. Proposed Pseudocode

4. Discussion and Results

The study used the SQL map tool to reattempt an attack on the prototype website after the implementation of suggested security measures. The subsequent reattack revealed the website's resilience, effectively proving that the suggested approach successfully safeguards against SQL injection hacking. Figure 6 illustrates that upon integrating this solution into the web application, the parameters become indiscernible, rendering it impossible for hackers to access any data.

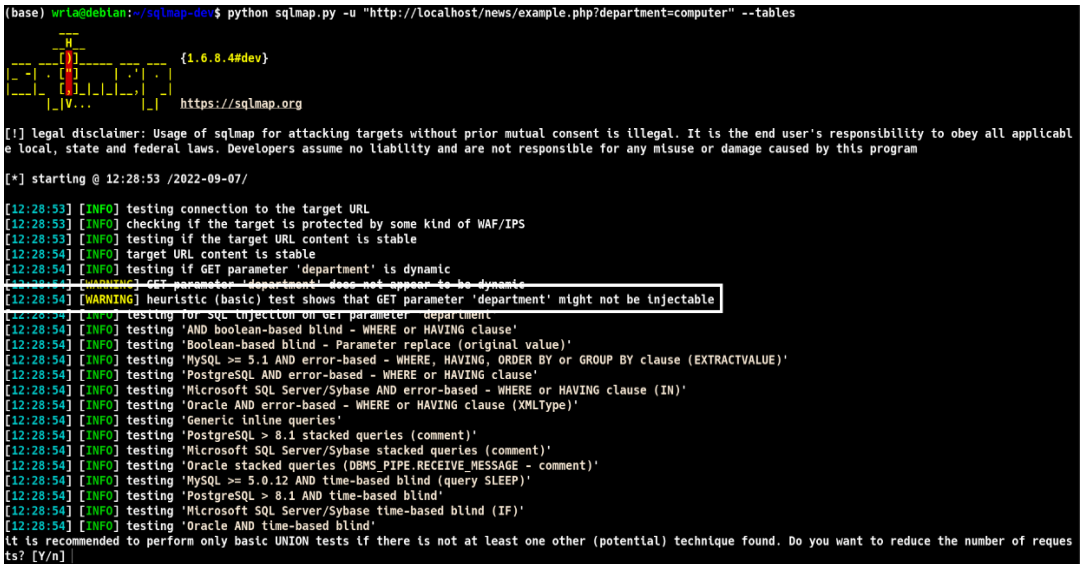

Fig.5. SQL Map Injection Attacking Queries

DOI: http://dx.doi.org/10.25098/8.1.32

The Scientific Journal of Cihan University – Sulaimaniya **PP: 219-229**
Volume (8), Issue (1), June 2024
ISSN 2520-7377 (Online), ISSN 2520-5102 (Print)

Fig. 6 visually illustrates the substitution of the URL query string's ID with a highly secure alternative, not stored in the server's database. This prevents SQL injection attacks. Even if hackers acquire this secure ID, it won't access any database info via the URL query string. The absence of this static ID in the database removes the vulnerability of the URL query string to attacks.

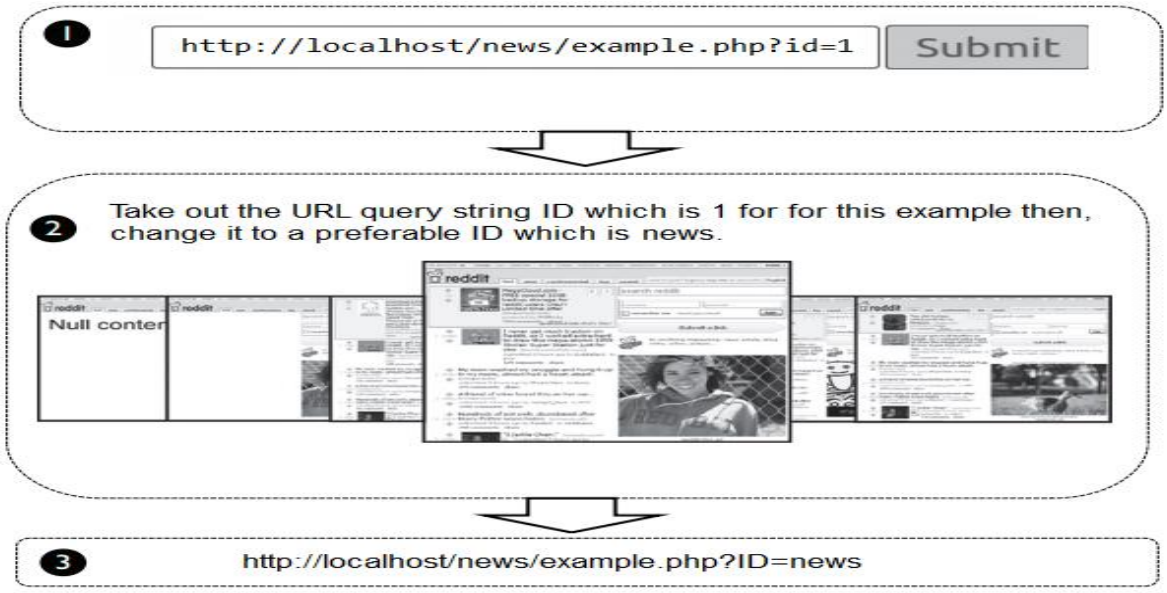

Fig.6. Result of detection for test

5. Conclusion

Developers commonly employ URL query strings to transmit sensitive parameters for database data retrieval in web applications. However, storing parameters directly in the database poses a security vulnerability. Hackers exploit this vulnerability through SQL injection attacks, extracting information from the database.

This research introduces a preventive approach by converting the URL query string value into a more secure form. This secure query string serves as a session parameter, distinct from direct storage in the database. Subsequently, this session parameter facilitates data retrieval from the web application database. With this method, hackers are unable to access database content because the highly secure parameter isn't stored in the database; it solely functions as a key for the session from the query string.

DOI: http://dx.doi.org/10.25098/8.1.32

References

[1] M. C. Jaeger, M. C. Vieira, and C. A. Tacla, "Web services standards: An overview," Journal of Information Systems Engineering & Management, vol. 3, no. 3, pp. 1-11, 2018.

[2] Zhiquan Lai, Yongjun Shen and Guidong Zhang, "A security risk assessment method of website based on threat analysis combined with AHP and entropy weight," 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016, pp. 481-484, doi: 10.1109/ICSESS.2016.7883113.

[3] Yin, C., Awlla, A. H., Yin, Z., & Wang, J. 2015.Botnet detection based on genetic neural network. International Journal of Security and Its Applications, 9(11): 97-104.

[4] Peng Tang, Weidong Qiu, Zheng Huang, Huijuan Lian, Guozhen Liu, Detection of SQL injection based on artificial neural network, Knowledge-Based Systems, Volume 190, 2020, 105528.

[5] Q. Li, W. Li, J. Wang and M. Cheng, "A SQL Injection Detection Method Based on Adaptive Deep Forest," in IEEE Access, vol. 7, pp. 145385-145394, 2019, doi: 10.1109/ACCESS.2019.2944951.

[6] M. Hasan, Z. Balbahaith and M. Tarique, "Detection of SQL Injection Attacks: A Machine Learning Approach," 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), 2019, pp. 1-6, doi: 10.1109/ICECTA48151.2019.8959617.

[7] X. Xie, C. Ren, Y. Fu, J. Xu and J. Guo, "SQL Injection Detection for Web Applications Based on Elastic-Pooling CNN," in IEEE Access, vol. 7, pp. 151475-151481, 2019, doi: 10.1109/ACCESS.2019.2947527.

[8] Mamdouh Alenezi, Muhammad Nadeem, Raja Asif, "SQL injection attacks countermeasures assessments", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 21, No. 2, February 2021, pp. 1121-1131.

[9] Abikoye, O.C., Abubakar, A., Dokoro, A.H. et al. A novel technique to prevent SQL injection and cross-site scripting attacks using Knuth-Morris-Pratt string match algorithm. EURASIP J. on Info. Security 2020, 14 (2020).

[10] JANG Young-Su. Detection of SQL Injection Vulnerability in Embedded SQL. IEICE Transactions on Information and Systems, IEICE TRANS. INF. & SYST., VOL.E103–D, NO.5 MAY 2020.

[11] Falor, A., Hirani, M., Vedant, H., Mehta, P., Krishnan, D. (2022). A Deep Learning Approach for Detection of SQL Injection Attacks Using Convolutional Neural Networks. In: Gupta, D., Polkowski, Z., Khanna, A., Bhattacharyya, S., Castillo, O. (eds) Proceedings of Data Analytics and Management. Lecture Notes on Data Engineering and Communications Technologies, vol 91. Springer, Singapore.

[12] S. Nanhay, D. Mohit, R.S. Raw, and K. Suresh, "SQL Injection: Types, Methodology, Attack Queries and Prevention", in 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016, p. 2872 – 2876.

[13] K.G. Vamshi, V. Trinadh, S. Soundabaya, and A. Omar, "Advanced Automated SQL Injection Attacks and Defensive Mechanisms", in Annual Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA), 2016, p. 1-6.

DOI: http://dx.doi.org/10.25098/8.1.32

[14] K. Krit and S. Chitsutha, "Machine Learning for SQL Injection Prevention on Server- Side Scripting", in International Computer Science and Engineering Conference (ICSEC), 2016, p. 1-6.

[15] P.K. Raja and Z. Bing, "Enhanced Approach to Detection of SQL Injection Attack", in 15th IEEE International Conference on Machine Learning and Applications (ICMLA), 2016, p. 466 – 469.

[16] D. Rhythm and G. Himanshu, "SQL Filtering: An Effective Technique to prevent SQL Injection Attack", in International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2016, p. 312 – 317.

[17] Kolšek, Mitja. "Session fixation vulnerability in web-based applications", "Acros Security" Available online: http://www.acrossecurity.com/papers/session_fixation.pdf,2002 (Accessed 14 Sep 2022).

[18] IETF, RFC2616: Hypertext Transfer Protocol -- HTTP/1.1, Available online: https://tools.ietf.org/html/rfc2616 (Accessed 14 Sep 2022).

[19] OWASP, Session Management Cheat Sheet, Available online: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md (Accessed 14 Sep 2022).

DOI: http://dx.doi.org/10.25098/8.1.32