Mobile and Forensics
Vol. 6, No. 1, March 2024, pp. 39-50
ISSN: 2714-6685 (online), 2656-6257 (print)
DOI: http://dx.doi.org/10.12928/mf.v6i1.8198

WEBSITE APPLICATION SECURITY VALUE ANALYSIS USING CRAWLING METHOD AGAINST SQL INJECTION ATTACKS

1,*Ari Dimas Yudistiawan, 2Nuril Anwar
1,2Department of Informatics, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
1,*aridimas145@gmail.com, 2nuril.anwar@tif.uad.ac.id
*correspondence email

Abstract

This study focuses on analyzing security vulnerabilities in the Codelatte website, specifically targeting SQL Injection attacks. While the implementation of HTTPS has enhanced user communication and server security, outdated pages lacking WordPress security features remain vulnerable to SQL Injection. The research employs a crawling method to assess security gaps, starting with URL processing, data crawling, and interaction with the web server, followed by output generation in the form of an HTML file. The collected data is used for vulnerability testing via the Acunetix web vulnerability scanner, as well as manual testing and SQL map penetration testing. Findings reveal security weaknesses categorized into informational, low, medium, and high-risk levels. Through crawling, the study identifies vulnerabilities and reduces them to informational, low, and medium levels, highlighting the importance of regularly updating website security. The final report provides recommendations for enhancing the Codelatte website's security to prevent unauthorized database access.

Keywords: Security Gap Analysis, Crawling, Information Security, Website Security, SQL Injection.

INTRODUCTION

The internet as a global communication network can be used as a media and source of the latest information, such as science, technology, entertainment, business, and other sources of information. This ease and convenience cause the internet to always be used and needed. However, behind the ease and convenience provided, it turns out that there is one aspect that is currently still lacking in attention by internet users, namely security. Security is one of the important aspects of the media that connects users to the internet[1].

It is undeniable that technology can have a lot of negative impacts. The development of the internet has made crimes that were previously conventional such as threats, theft, embezzlement, forgery and fraud can now be carried out online through the internet media which has the advantage of minimal risk of being caught by individuals or groups[2]. The negative impacts that are detrimental and developing rapidly have resulted in the idea that no computer and network are truly safe to date[3]. This is evident from the many novice hackers who appear on various internet platforms to commit cybercrime[4].

As technology develops rapidly from time to time, many young hackers have emerged to commit crimes with various new attack methods that they use to launch their actions[5], such as SQL Injection, Directory Traversal Attack, Cross Site Scripting and so on. SQL Injection is one example of the most dangerous method when successfully utilized by hackers or crackers. SQL Injection has long been popular in the world of hacking as one of the web application hacking techniques, although this technique is quite old, this technique is still one of the mainstay techniques because of its nature which can damage the database of a site. One method in the SQL

Article History: *Received June 25, 2023; Revised July 31, 2024; Accepted Agustus 6, 2024*

Injection technique is to input standard commands in SQL such as insert, create, update, union, select, view, along with other standard commands that are familiar to those who have studied SQL in depth or those who are just learning[6].

SQL Injection is a method of attack against vulnerabilities that occur when an attacker has the ability to influence the Structured Query Language (SQL) Query that passes through an application to a back-end database that is able to manipulate what will be forwarded to the database[7]. Attackers take advantage of the syntax and capabilities of SQL itself, as well as its strength and flexibility to support the operation and functionality of the system available to the database[8]. SQL Injection is caused by the absence of a filter on the SQL language so that attackers can exploit the vulnerability with the aim of gaining access to the SQL-based database system and infiltrating it[9]. This study conducted a trial of finding SQL Injection security holes on 5 websites, including uad.ac.id, codelatte.id, pt-jayapura.go.id, mk.mercubuana-yogya.ac.id, jateng.polri.go.id. Of the five websites, three of them found SQL Injection security holes in them, the three websites are codelatte.id, pt-jayapura.go.id, mk.mercubuana-yogya.ac.id, and pt-jayapura.go.id. However, the repair permit was only sent to the codelatte.id website and pt-jayapura.go.id, and the codelatte.id website allowed repairs on its website, therefore the codelatte.id website was chosen as the object of this study.

Codelatte is an organization that focuses on web development services (web developer) and penetration testing[10]. Codelatte has a website as a means of advertising, branding, and increasing existence[11]. The website was targeted using a protocol that is tasked with sending data from the web server to the browser using HTTPS (Hypertext Transfer Protocol Secure) on each of its subdomains. After doing a Lookup, it was discovered that the website uses PHP and Wordpress as its Content Management System (CMS)[12]. WordPress is the CMS (Content Management System) with the most users currently in website development. The convenience and security system provided by WordPress as a service provider are very satisfying to its users, so that the majority of its users are reluctant to turn to other CMS[13].

The implementation of WordPress is inversely proportional to the implementation of HTTPS, the use of Wordpress is not applied to each subdomain owned by the Codelatte website[14]. The remaining old website files in the file manager cause gaps to be found in other subdomains of the website, for example, after crawling and scanning which functions to find any security gaps in the website, it was found that there was an open SQL Injection gap in the Codelatte web sub-domain[15].

Crawling not only results in the exposure of a website weakness, crawling can also be used by pentesters to find the location of weaknesses and fix them so that the security level of a website increases[16], the acunetix tool has an engine to perform manual crawling if the user does not want to use automatic testing on the acunetix web vulnerability scanner, this tool is very useful for both parties, both from the attacker and the pentester side to find site weaknesses to infiltrate the system or fix them[17]. Dirsearch is a tool that relies on the bruteforce method and checking the http respond code in folders that are often used on the website in the crawling process[18]. By utilizing this dirsearch tool, someone can easily find out the directory path, search for sensitive data, search for hidden directories, and so on.

Attackers who successfully find and exploit the SQL Injection gap in the sub-domain of the site and if they have succeeded in stealing access to the database, then the attacker will have the freedom to manipulate the data contained on the website. It doesn't stop there, attackers are also able to jump between sub-domains and even to the main domain to install a backdoor which results in the attacker being able to gain full access from within the domain.

Irresponsible things that can be done by attackers from within the website database can certainly pose a risk of leaking organizational Confidentiality, reducing the integrity of information and the availability of data when need becomes uncertain[19]. SQL Injection attacks are an attack method with very dangerous impacts and can be carried out remotely by irresponsible parties and fixing them can help maintain the privacy of the Codelatte website[20], so the author decided to take this theme by using the title "Website Application Security Value Analysis Using Crawling Method Against SQL Injection Attacks"

METHODS

The object of this research is a website engaged in web development and penetration testing, the organization's website is located at codelatte.org, which currently has a main domain at codelatte.id.

A. Proposed Method

Data collection methods in this study include:

1. Observation Method
   Conducting observations on how the software and network used work. Observations are made on accessible websites and conducting experiments on features on each page that have the possibility of security holes. By conducting observations can help to save time in testing the target website.
2. Crawling to find vulnerabilities
   Crawling as an information gathering algorithm, security holes in the codelatte.org website will be exposed and collected based on the low or high order of a vulnerability. This method is carried out using the help of the crawler engine owned by the acunetix web scanner software.

B. Research Stage

The crawling method in this study uses the crawling engine on the acunetix web vulnerability scanner tool to find the location of vulnerabilities on the codelatte.org site which of course has its own scenario. Before entering the research stage, the crawling scenario will be described,

1. Entering URL
   The Codelatte website certainly has a URL address as a tool to identify and access its website. The URL can be used as a first step to start crawling. In this first scenario, the URL is entered into the acunetix tool.
2. URL Processing
   At this stage, the acunetix tool will enter the URL to be processed into its crawling engine. URL processing begins when the website URL has been entered into the input column. The URL is taken to the crawling engine and the system will decide, if the URL has not been crawled then go to the next step, which is to make an http request to the Codelatte website's web server, which will then produce an HTML file output.
   The HTML file output from the web server is used to retrieve and extract other URLs contained in it, then entered the URL data. The URL data then sends the extracted URL to the crawling engine to crawl the new URL from the HTML file extraction results and sends the initial URL to the URL List as a URL that has passed the crawling stage. URL parsing is done with the aim of parsing the URL in the URL list to bring it into the URL testing stage. This URL testing stage has many types of testing processes, especially SQL Injection,
   The results of the URL Testing will be brought back to the initial process, namely at the time of the URL Processor, to produce output in the form of test results. Next, the new URL found during extraction will go through the same stages as the first URL, so the complete scenario flowchart will look like Fig.1.

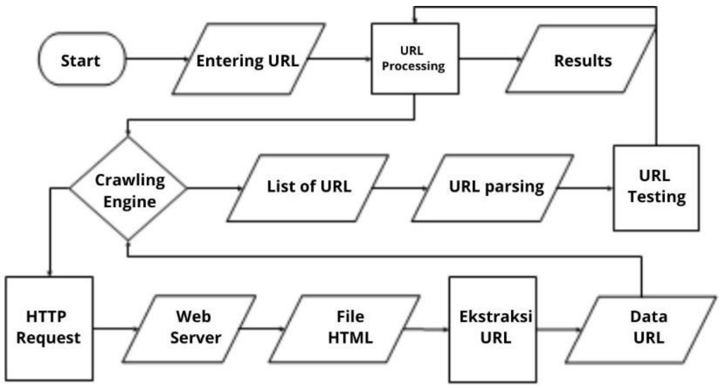


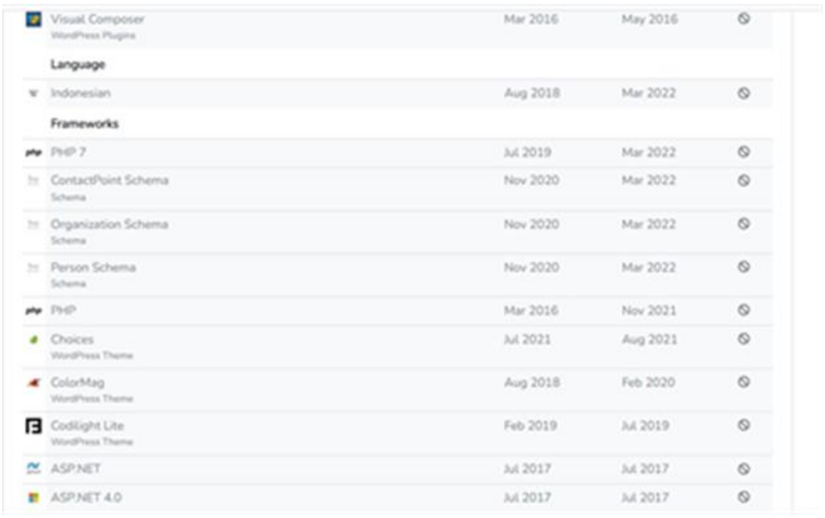
Fig.1 Scenario Flowchart

3. Crawling Results
   The results of the entire URL processing series will be displayed in detail to the interface system, so that the lowest and highest vulnerabilities from the scanning results using the crawling engine owned by the Acunetix web vulnerability scanner tool owned by the Codelatte site can be seen, from the scanning results using the crawling engine owned by the Acunetix web vulnerability scanner tool.

RESULT AND DISCUSSIONS

The process flow carried out to create a report includes Scope, reconnaissance, vulnerability detection, information analysis & planning, penetration testing, Permits and system improvements. This vulnerability detection process includes a crawling method in finding security holes on the Codelatte.org website by utilizing the crawling engine owned by the Acunetix tools.

A. Scope
   Scope is the first step in conducting penetration testing, determining the scope of penetration testing is done to prevent the test from widening during the penetration testing process. The scope of penetration testing used in this study is of course the Codelatte website.
B. Reconnaissance
   Reconnaissance is the next stage when the scope of penetration testing has been determined, Reconnaissance itself means collecting as much information as possible. The information collected is information about the Codelatte website which is then used as material for penetration testing. The collection of information will begin with an examination of the website architecture using built with as in Fig.2.
   Fig.2 shows the website architecture information found by the builtwith-lookup tool. The builtwith tool is a tool that can detect the technology architecture used by a website and display it in the form of a descending description. Based on the search, it was found that Codelatte does not use Laravel as its framework, so there is a possibility of finding a sql injection security hole. Furthermore, after finding out the website architecture information, reconnaissance is carried out manually by directly checking each target web page with the dir search tool. DirSearch is a tool designed to crawl directories and files on a website. After the process is complete, dirsearch will produce a complete directory list output with its URL address.



Visual Composer WordPress Plugins	Mar 2016	May 2016	⊘
Language			
Indonesian	Aug 2018	Mar 2022	⊘
Frameworks			
PHP 7	Jul 2019	Mar 2022	⊘
ContactPoint Schema Schema	Nov 2020	Mar 2022	⊘
Organization Schema Schema	Nov 2020	Mar 2022	⊘
Person Schema Schema	Nov 2020	Mar 2022	⊘
PHP	Mar 2016	Nov 2021	⊘
Choices WordPress Theme	Jul 2021	Aug 2021	⊘
ColorMag WordPress Theme	Aug 2018	Feb 2020	⊘
Codilight Lite WordPress Theme	Feb 2019	Jul 2019	⊘
ASP.NET	Jul 2017	Jul 2017	⊘
ASP.NET 4.0	Jul 2017	Jul 2017	⊘

Fig. 1. Codelatte.org Web Architecture

C. Vulnerability Detection

The vulnerability detection stage or security gap detection in the Codelatte domain aims to find all dangerous security gaps contained in the stack of website pages. Acunetix is a tool for security testing that is accessed through a virtual server. As a useful tool for security testing, acunetix checks various codes, files and entire websites that have the potential to contain vulnerabilities such as SQL injection, xss, and many other vulnerabilities. The crawling results are presented in Figure 3,

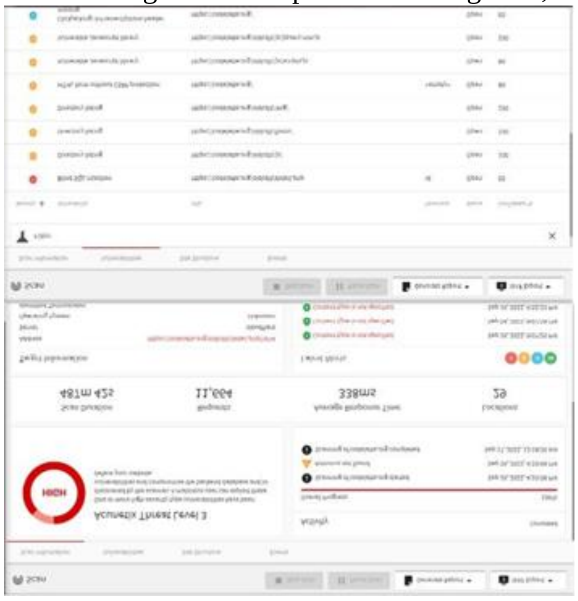


Fig. 2. Acunetix Scan Results

Figure 3 shows that there is one Blind SQL Injection vulnerability on the website, two medium threat level security vulnerabilities, four low threat level security vulnerabilities, and three informational level security vulnerabilities. After finding the location of the vulnerability on the website through the crawling method, the next step is to analyze and plan the information that has just been found at the information analysis & planning stage.

D. Information Analysis and Planning

Vulnerability detection in the previous stage resulted in several security gaps on the codelatte.org website, so that in the information analysis & planning stage, it will choose which security gaps will be exploited in penetration testing, among these security gaps, the security gaps used for testing are listed in Table 1.

Table 1. Selected Security Vulnerabilities

No	Type of Vulnerability used
1.	SQL Injection
2.	Directory listing

The security gaps in table 4.1 were selected based on the information analysis carried out where Sql Injection was the main focus for conducting penetration testing because the damage when injecting SQL could include all website data needed to gain access to the system, while the directory listing gap became support for the running of the SQL injection.

E. Penetration Testing

Penetration testing is done to validate or prove that the security gaps found are true, so that the validated security gaps are then used as targets for improvement. The testing is done in stages as follows:

1. SQL Injection

   Vulnerabilities found from crawling results at the vulnerability detection stage are then used as targets, to validate or prove the existence of the security gap. Identification of SQL Injection through the URL on the codelatte.org domain, is done by adding a string (') at the end of the URL. This identification will produce findings in the form of security gap characteristics that can be used for SQL Injection testing, the URL address that will be used for testing is, http://codelatte.org/oldcdlt/detail.php?id=4, after the string is added, the page shows an error by displaying emptiness.

   The page that has the characteristics of the SQL Injection security gap has been successfully identified, the next step is to conduct a trial by testing using SQL Map version 1.6.8.2#dev. Testing through SQL Map produces results in the form of injections carried out by SQLMap into the database and opening tables in the website database. After opening the tables in the website database, it was found that one of the tables owned by the lush database was the users table that had login access to the codelatte.org website login panel.

   The Lush database displays the Username/password tables when the database is opened. When a dump has been performed and the username/password has been found, administrator access has been obtained. Then a test is carried out to enter the username/password into the admin panel that was found. The admin panel can be seen in Figure 4.

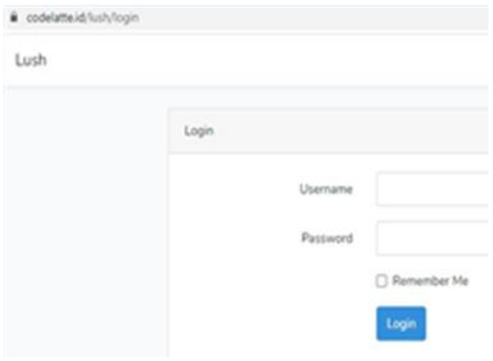


Fig. 3. Login Admin Page Lush

Fig. 5 shows the view of the admin panel of the private application belonging to the Codelatte website. The admin panel dashboard of the private application will open when using the correct username and password, and will display a display Fig. 5.

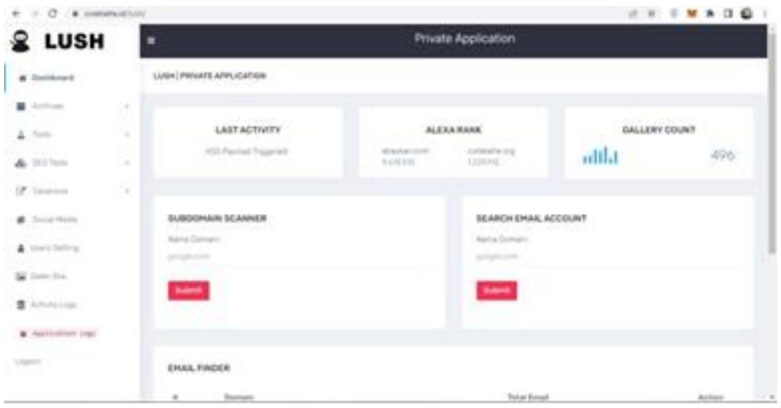


Fig. 4 Index Admin Page Lush

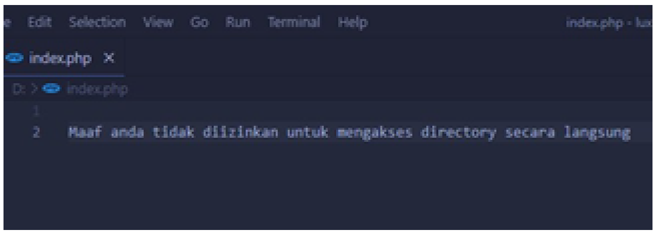
Fig. 5 shows the private application dashboard page where the admin has access to create, use, delete documentation of articles and tools on the website and edit usernames and passwords belonging to users and admins.

2. Listing Directory
   Listing Directory Security Vulnerability is actually a web function that displays a list of all file contents when there is no index in the directory, such as index.php, index.html and default.asp. Usually, a user who opens www.example.org/oldcdlt/js without writing the file name makes the web server process this request and will display the index file located in the directory and the actual website will appear. However, if the index file does not exist as in this case, the web server will return the listing directory or list of contents of the directory. This functionality can be paralleled using the 'ls' command on unix and linux and 'dir' on Windows. 4 medium-type security vulnerabilities were found on the codelate.org website, each of which contains information about, JS, CSS, IMG and Font where the four directories can be accessed and downloaded publicly.

F. System Improvement Permission
The system improvements carried out require permission as a legal force for the exploitation carried out during penetration testing. Therefore, a request for permission for penetration testing and repairs was made via email listed in the contact us column of the Codelatte website. The permission email was sent to the developer of codelatte.org and was responded to with a positive response from the developer in the form of a request to keep in touch while the penetration testing process was running. After sending the permission email, the research continued to the repair stage.

G. Repairing
This stage contains about fixing security holes that were found and have been completed through the penetration testing process on the codelatte.org website. The vulnerabilities include SQL Injection and Directory Listing found on one of the old directories owned by codelatte.org.

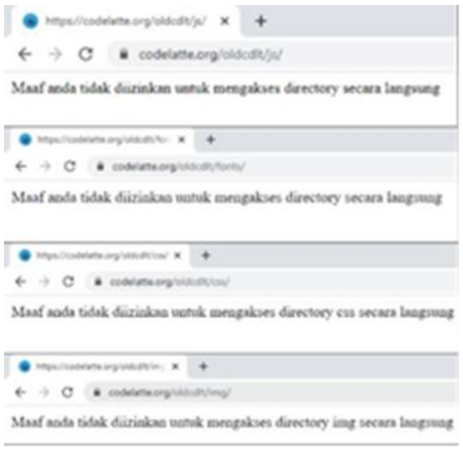
1. Repairing Directory Listing
   Directory listing repair can be done with a simple method so that to strive for effective and efficient repair, the best method will be carried out. The method is to create index.php which is then placed into the directory with a gap in dirlisting. The process of creating an index.php file serves to fill the empty file that is called by the web server when the user asks to open a directory folder without including a specific file. Creating an index.php file is done by opening the code editor and entering plaintext as in Fig. 6.



```
Maaf anda tidak diizinkan untuk mengakses directory secara langsung
```

Fig. 5 Repaired File

Fig. 6 shows the plaintext stored under the name index.php. After that, upload the newly created index.php file into the directory that has the directory listing security hole. Then after the upload is successful, the directory listing display will look like Fig. 7.



codelatte.org/oldcdlt/js/
Maaf anda tidak diizinkan untuk mengakses directory secara langsung

codelatte.org/oldcdlt/fonts/
Maaf anda tidak diizinkan untuk mengakses directory secara langsung

codelatte.org/oldcdlt/css/
Maaf anda tidak diizinkan untuk mengakses directory css secara langsung

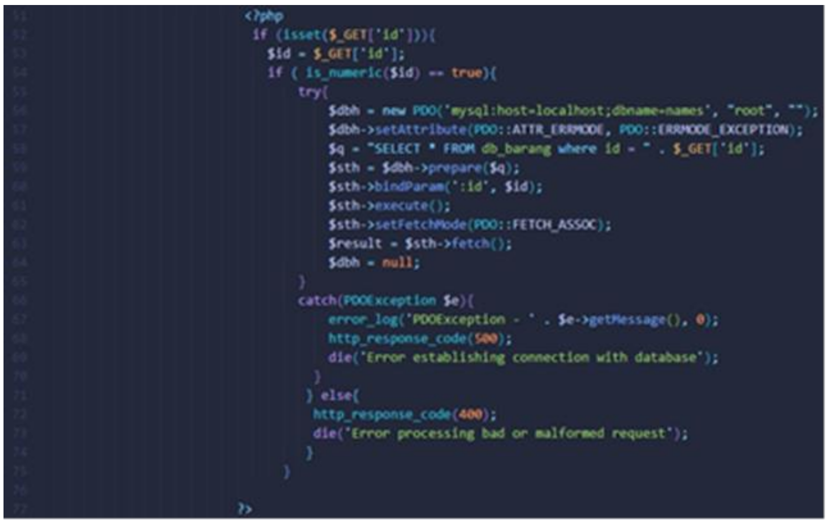
codelatte.org/oldcdlt/img/
Maaf anda tidak diizinkan untuk mengakses directory img secara langsung

Fig. 6 Result Repairing Directory Listing

Fig. 7 shows the contents of the index.php file in the js directory that has a directory listing security hole, so that the display of the file contents list or directory listing is no longer displayed. The prevention is done by adding the code "Option All -Indexes" to Htaccess so that before entering the index.php file, the web server will send an error message to the user interface.

2. Repairing SQL Injection
   The cause of the SQL Injection security gap must be known and understood first before moving on to the repair stage, with the hope that the repairs carried out are effective and on target. To find out, an examination was carried out on the source code that had a weakness gap and it was found that the connection from the website to the database still used the mysqli extension without using a database abstraction layer such as PDO (PHP Data Object). PDO is a database access layer that provides a uniform access method to several databases. The code used in implementing PDO can prevent the bad query process carried out by attackers for SQL Injection cases.

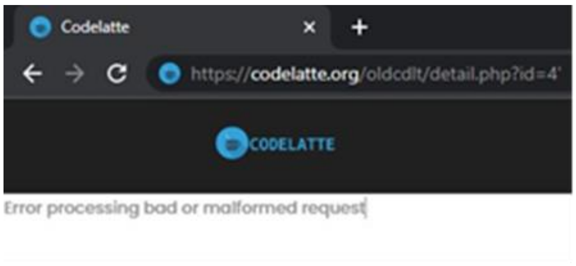
Thus, to overcome the cause of the SQL Injection security gap, PDO was implemented on the Source Code used on the codelatte.org page. PDO is applied to the source code used by replacing the mysqli query connected to the database according to the PDO code as in Fig. 8.



```
51  <?php
52   if (isset($_GET['id'])){
53     $id = $_GET['id'];
54     if ( is_numeric($id) == true){
55         try{
56             $dbh = new PDO('mysql:host=localhost;dbname=names', "root", "");
57             $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
58             $q = "SELECT * FROM db_barang where id = " . $_GET['id'];
59             $sth = $dbh->prepare($q);
60             $sth->bindParam(':id', $id);
61             $sth->execute();
62             $sth->setFetchMode(PDO::FETCH_ASSOC);
63             $result = $sth->fetch();
64             $dbh = null;
65         }
66         catch(PDOException $e){
67             error_log('PDOException - ' . $e->getMessage(), 0);
68             http_response_code(500);
69             die('Error establishing connection with database');
70           }
71         } else{
72          http_response_code(400);
73          die('Error processing bad or malformed request');
74         }
75     }
76
77 ?>
```

Fig. 7 Applying PDO to Source Code

Fig. 8 shows the codelatte.org source code that has been changed by implementing PDO in the database coding.51-78 Line 52 of the code functions to check whether the GET variable 'id' has been installed, line 54 of the code functions as data validation before entering the database. In this case, it checks whether the value of the GET parameter ‘id’ is numeric or not, line 56 is used to setup a connection to the database, line 57 is used to record errors and write them to the log file, line 58 is the code to run a database query and this line is the PDO code that fixes the SQL Injection vulnerability, line 59 is used to prepare the SQL query string, line 60 is used to bind parameters to the statement variable, line 61 is used to execute the statement, line 62 is used to change the fetch mode to FETCH_ASSOC to return an array indexed by column name, line 63 is used to fetch the results, line 64 is used to close the connection to the database, lines 66 - 67 are used to log PDO exceptions to the php system log using the operating system's log engine, lines 68-69 are used to stop execution and return a 500 error display, which is an internal system error, lines 71-73 are executed when the value of the GET parameter ‘id’ is not numeric, stop execution, and return a 'Bad request' HTTP status code (400). When the data has been submitted to PDO as it is now, a check is carried out by inserting the quotation character (‘) at the end of the URL as in Fig. 9.



Error processing bad or malformed request

Fig. 8 PDO Application Results

Figure 9 shows an error when the parameter content is not a number but another character, this has been set in PDO in its code to prevent attackers from trying to penetrate the system database even deeper.
Unfortunately, the PDO implemented is only one line of defense to overcome SQL Injection, the protection provided by PDO does not cover vulnerabilities to other input validation, but it works well in protecting websites from SQL injection. By using PDO, another strategy to maximize protection is to filter the data, meaning not allowing dangerous data or characters to be entered into the query. Then, filtering is carried out on dangerous characters that can display output in the form of the contents of the codelatte.org system database. The filtering carried out is:

- URL Validation Filtering Numeric filterin.
  This URL validation is useful for checking whether the URL is valid or not, with the aim of checking the input entered by the user, whether it is in accordance with the URL writing standard or not.
- Number filtering in the parameter
  The is_numeric filter checks the type of data entered by the user whether it is a number or not. The goal is so that attackers cannot use the quoted string (') to try to enter the system database.
- Html Entities and XSS
  The string filter is also able to protect against cross site scripting (xss) attacks, because the html or javascript script will be changed into a regular string. So that makes the code important and should be applied to this codelatte.org website.

CONCLUSIONS

The conclusion of the research conducted is that crawling carried out using the help of the acunetix crawling engine produces URL addresses and vulnerability diagnoses from low to high levels, so that using it can help find security holes in the codelatte website application. The vulnerability of the crawling results can also be proven so that Penetration testing of the codelatte.org site can be carried out and repairs and prevention can be applied.
The repairs carried out for the directory listing security hole are in the form of creating index.php, while the SQL Injection repairs carried out are by implementing PDO replacing regular queries in the source code of pages that have security holes. Vulnerability has not been exploited by irresponsible parties before so that the damage experienced is limited to open security holes.
Prevention is also carried out to anticipate exploits of security holes that may occur in the future. Directory Listing is prevented by installing the code "Option All -Indexes" in Htaccess so that it will display another page on the user interface. SQL Injection is prevented by adding URL validation, filtering on numbers, and mixed characters such as strings, numeric metacharacters. Thus, the vulnerability contained in the codelatte.org website can be overcome, thereby eliminating the risk of privacy leaks from the website.

REFERENCES

[1] A. K. Jain, S. R. Sahoo, and J. Kaubiyal, ‘Online social networks security and privacy: comprehensive review and analysis’, *Complex Intell. Syst.*, vol. 7, no. 5, pp. 2157–2177, Oct. 2021, doi: 10.1007/s40747-021-00409-7.
[2] S. Cherniavskyi, V. Babanina, I. Vartyletska, and O. Mykytchyk, ‘Peculiarities of The Economic Crimes Committed with the Use of Information Technologies’, *Eur. J. Sustain. Dev.*, vol. 10, no. 1 SE-, p. 420, Feb. 2021, doi: 10.14207/ejsd.2021.v10n1p420.
[3] R. Ande, B. Adebisi, M. Hammoudeh, and J. Saleem, ‘Internet of Things: Evolution and technologies from a security perspective’, *Sustain. Cities Soc.*, vol. 54, p. 101728, Mar. 2020, doi: 10.1016/j.scs.2019.101728.
[4] T. S. Hyslip, ‘Cybercrime-as-a-Service Operations BT - The Palgrave Handbook of

International Cybercrime and Cyberdeviance', T. J. Holt and A. M. Bossler, Eds. Cham: Springer International Publishing, 2020, pp. 815–846.

[5] S. khadim, oday Ali Hassen, and H. Ibrahim, 'A Review on the Mechanism Mitigating and Eliminating Internet Crimes using Modern Technologies: Mitigating Internet crimes using modern technologies', *Wasit J. Comput. Math. Sci.*, vol. 1, no. 3 SE-Computer, pp. 50–68, Sep. 2022, doi: 10.31185/wjcm.48.

[6] M. d. Prayoga, 'Pengertian Dan Komponen Sql Muhammad Denny Prayoga', pp. 1–7, 2018, doi: https://doi.org/10.31219/osf.io/kj43y.

[7] Bangkit Wiguna, W. Adi Prabowo, and R. Ananda, 'Implementasi Web Application Firewall Dalam Mencegah Serangan SQL Injection Pada Website', *Digit. Zo. J. Teknol. Inf. dan Komun.*, vol. 11, no. 2 SE-Articles, pp. 245–256, Nov. 2020, doi: 10.31849/digitalzone.v11i2.4867.

[8] et al. Aliero, Muhammad Saidu, 'Review On Sql Injection Protection Methods and Tools', *J. Teknol. (Sciences Eng.*, vol. 77, no. 13 SE-Science and Engineering, Nov. 2015, doi: 10.11113/jt.v77.6359.

[9] M. A. Z. Risky and Y. Yuhandri, 'Optimalisasi dalam Penetrasi Testing Keamanan Website Menggunakan Teknik SQL Injection dan XSS', *J. Sistim Inf. dan Teknol.*, vol. 3, no. 4 SE-Articles, pp. 215–220, Sep. 2021, doi: 10.37034/jsisfotek.v3i4.68.

[10] F. W. Marrs *et al.*, 'Chemical Descriptors for a Large-Scale Study on Drop-Weight Impact Sensitivity of High Explosives', *J. Chem. Inf. Model.*, vol. 63, no. 3, pp. 753–769, Feb. 2023, doi: 10.1021/acs.jcim.2c01154.

[11] M. J. Cawkwell, A. C. Burch, S. R. Ferreira, N. Lease, and V. W. Manner, 'Atom Equivalent Energies for the Rapid Estimation of the Heat of Formation of Explosive Molecules from Density Functional Tight Binding Theory', *J. Chem. Inf. Model.*, vol. 61, no. 7, pp. 3337–3347, Jul. 2021, doi: 10.1021/acs.jcim.1c00312.

[12] M. Tomiša, M. Milković, and M. Čačić, 'Performance Evaluation of Dynamic and Static WordPress-based Websites', in *2019 23rd International Computer Science and Engineering Conference (ICSEC)*, 2019, pp. 321–324, doi: 10.1109/ICSEC47112.2019.8974709.

[13] R. Azis and S. Yazid, 'Pengujian Kerentanan Website Wordpress Dengan Menggunakan Penetration Testing Untuk Menghasilkan Website Yang Aman', *J. RESTIKOM Ris. Tek. Inform. dan Komput.*, vol. 3, no. 3 SE-Article, Jun. 2022, doi: 10.52005/restikom.v3i3.87.

[14] J. Rodas-Silva, J. A. Galindo, J. García-Gutiérrez, and D. Benavides, 'Selection of Software Product Line Implementation Components Using Recommender Systems: An Application to Wordpress', *IEEE Access*, vol. 7, pp. 69226–69245, 2019, doi: 10.1109/ACCESS.2019.2918469.

[15] I. Sanchez-Rola, D. Balzarotti, C. Kruegel, G. Vigna, and I. Santos, 'Dirty Clicks: A Study of the Usability and Security Implications of Click-related Behaviors on the Web', in *Proceedings of The Web Conference 2020*, Apr. 2020, pp. 395–406, doi: 10.1145/3366423.3380124.

[16] M. Khder, 'Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application', *Int. J. Adv. Soft Comput. its Appl.*, vol. 13, pp. 145–168, Dec. 2021, doi: 10.15849/IJASCA.211128.11.

[17] erick irawadi alwi, H. Herdianti, and F. Umar, 'Analisis Keamanan Website Menggunakan Teknik Footprinting dan Vulnerability Scanning', *INFORMAL Informatics Journal; Vol 5 No 2 INFORMAL - Informatics JournalDO - 10.19184/isj.v5i2.18941* , Aug. 2020, [Online]. Available: https://jurnal.unej.ac.id/index.php/INFORMAL/article/view/18941.

[18] Z. Alizadehsani, 'Proposing to Use Artificial Neural Networks for NoSQL Attack Detection BT - Distributed Computing and Artificial Intelligence, Special Sessions, 17th International Conference', 2021, pp. 247–255.

[19] A. Ramadhani, 'Keamanan Informasi', *Nusant. - J. Inf. Libr. Stud.*, 2018, [Online].

Available: https://api.semanticscholar.org/CorpusID:243906104.
[20] J. Wu, 'Security Risks from Vulnerabilities and Backdoors BT - Cyberspace Mimic Defense: Generalized Robust Control and Endogenous Security', J. Wu, Ed. Cham: Springer International Publishing, 2020, pp. 3–38.