ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 8, Issue 8, August 2021

Web Application Security Scanning using Machine Learning

[1] Dr. Harmeet Kaur Khanuja, [2] Pranav Gadekar, [3] Samruddhi Kulkarni, [4] Shalaka Kulkarni, [5] Shruti More

[1][2][3][4][5] B Dept. of Computer Engineering, MMM's College of Engineering, Chennai, Tamil Nadu, India

***Abstract*---- Web and web-based technologies have gained popularity in recent times. The security-sensitive information and functionalities of web applications can be extracted easily. Web applications are the most common source of sensitive data, so they are more vulnerable to a large number of web-based attacks. Incorrect input validation is one of the primary reasons for vulnerabilities to take place.Though these vulnerabilities are simple in nature and usually easy to mitigate, developers are unaware of security implications of these issues. This results in more vulnerable web applications on the Internet. If these vulnerabilities remain present in the web application, then it might have some severe impacts on confidentiality of user data.**

We implemented a system which crawls the entire web application to collect all referenced URLs and scan those URLs for the most frequent vulnerabilities like SQL Injection and Cross Site Scripting. A comprehensive report for sub types of SQL injection like Error-based, Union and Boolean SQL injection along with Cross Site Scripting, is presented to users. Each of the aforementioned reports consists of URLs vulnerable to SQL Injection or Cross Site Scripting attacks.

***Keywords*— SQL Injection, Cross Site Scripting, Web Application Testing, Security Scanner, Exploitation, Code Injection, Web Security, Machine Learning, Artificial Intelligence**

I. INTRODUCTION

As of January 2020, there have been over 1.74 billion websites on the web. On an average hackers attack after every 39 seconds, that is 2,244 times a day. This gives us the idea that many websites on the Internet are vulnerable to different attacks. [1] As of the end of 2019, 42% of publicly facing websites are prone to SQL Injection and 19% to Cross Site Scripting attacks. A security researcher has earned a $25,000 bug bounty after finding a Cross Site Scripting (XSS) vulnerability in one of the most popular social media sites 'Facebook'. Another such attack, in August 2019, was on the famous coffee chain 'Starbucks' web services that created a way to access their critical database through the SQL Injection Vulnerability. [2] From this discussion, we can conclude that security has a major role to play while developing websites. Unfortunately, web developers are not aware of these security aspects resulting in more vulnerable websites. Some of the most commonly occurring ones being SQL injection and Cross Site Scripting. So we have developed a system that will find these vulnerabilities in given web applications and report them to the user of the system.

We have designed a web application that accepts the target URL from the user. Then it passes the accepted URL to a Web crawler that crawls the given URL and collects all the referenced URLs. Then it scans all collected URLs and it tests different payloads to detect the vulnerabilities using machine learning. Finally, a report is generated which contains the detected vulnerabilities.

II. RELATED WORK

- Machine Learning for Web Vulnerability Detection: The Case of Cross-Site Request Forgery published within the year 2020 by Stefano Calzavara, Mauro Conti, Riccardo Focardi, Alvise Rabitti, Gabriele Tolomei. Its main advantage is that it offers a language-sceptic vulnerability detection perspective, which hides the complexity of scripting languages as it offers a compatible interface to a large range of web applications. [3]
- An efficient algorithm and tool for detecting dangerous website vulnerabilities in the year 2020 and written by Hoang Viet Long, Tong Anh Tuan, David Taniar, Nguyen Van Can, Hoang Minh Hue. The given technique has the key feature of detecting attacks involving nested SQL queries and gives fine results. [4]
- Dimitris E. Simos, Jovan Zivanovic, Manuel Leithner proposed Automated Combinatorial Testing for Detecting SQL Vulnerabilities in Web Applications in the year 2019. It shows that our approach can effectively escape defective filtering mechanisms. [5]
- Commix: automating evaluation and exploitation of command injection vulnerabilities in Web applications published within the year 2019 by Anastasios Stasinopoulos, Christoforos Ntantogian and Christos Xenakis. It gives access to a variety of functionalities

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 8, Issue 8, August 2021

that try to cover vast exploitation scenarios such as authentication mechanisms, custom headers, attack vectors developed using programming languages and user enumeration. [6]

- A Distributed Vulnerability Scanning on Machine Learning in the year 2019 by Xiaopeng TIAN, Di TANG. Setting up standardized and quantified data sets for various industries and businesses is of great assistance to increase the testing standard. [7]
- An Automated Composite Scanning Tool with Multiple Vulnerabilities within the year 2019 published by Xun Zhang, Jinxiong Zhao, Fan Yang, Qin Zhang, Zhiru Li, Bo Gong, Yong Zhi, Xuejun Zhang. It assures the automatic detection for executing automatic vulnerability scanning. [8]

III. PROPOSED SYSTEM

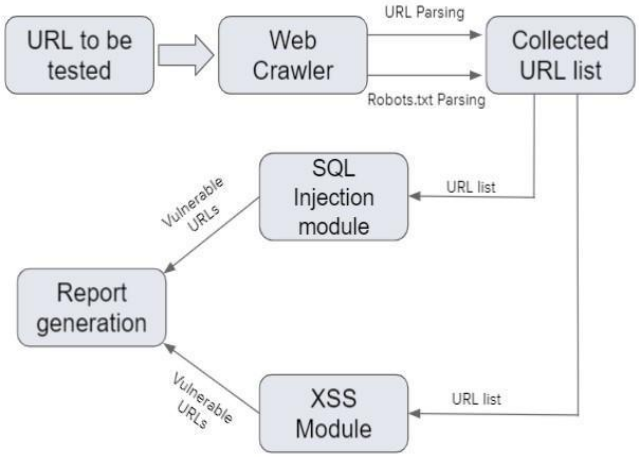

Fig. 1. System Architecture

The figure given above (Fig. 1. System Architecture), represents the architecture of the proposed system. The proposed system has four modules, those are, Web Crawler, SQL
Injection detection, Cross Site Scripting detection and Report generation. These modules are described in a later part of the paper.

We have developed a web application of the proposed system. The web application has sign-in and sign-up functionalities to log in and add new users to the system respectively. Users have to log in to the system in order to use the system's functionalities.

- After successful login, the user is directed to the web page where the user can provide the URL of the web application to be tested.
- The URL given by the user is passed to the Web crawler module where all referenced URLs are collected recursively from referenced URLs found previously.
- The SQL Injection and Cross Site Scripting modules of the system will get the set of referenced URLs, where URL query parameters, forms and cookies present on that web page will be scanned for vulnerabilities.
- After that, a report will be generated for each of the detected vulnerabilities which contains URL on which vulnerabilities were found and payload used to detect them

A. Project Scope

- The system requires the target URL to be entered by the user.
- If the web application is not having the robots.txt file then the user has to explicitly specify the restricted URLs.
- The system will scan the target application and check if the web application is having any of these vulnerabilities:
  - Reflected SQL Injection
  - Union SQL Injection
  - Boolean SQL Injection
  - Cross Site Scripting
- The report will be generated consisting of endpoint affected, payload used, and generalized remediation.

IV. WEB CRAWLER

As given in Fig. 1, the Web crawler is the first module of the proposed system. The Web Crawler has two sub-modules, that are, Robots.txt parsing and URL parsing.

A. Robots.txt parsing

The web application to be checked for vulnerabilities, may or may not have a robots.txt file. This file basically contains the details of User agents for the web application and disallowed URLs for that User agents. Our system parses the robots.txt file to get both the allowed and disallowed URLs for web applications. The disallowed URLs will not be crawled and tested for vulnerabilities.

- Checks for the presence of the robots.txt file and if present, collect allowed and disallowed URLs.

B. URL Parsing

The URL parser takes input as the URL of the home page or main page and it finds all the referenced URL present on that page. Then it visits all those URLs one by one and collects all the referenced URLs on the page. This procedure continues in a recursive manner.

- All URLs specified within the anchor tag from the

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 8, Issue 8, August 2021

current page are saved in a List.

- Relative URLs (like /admin or #footer) are converted into Absolute URL (like https://example.com/admin or https://example.com#footer)
- URLs which are not in the scope of target application are removed from the list (for example twitter.com or instagram.com)
- Hyperlinks with 'mailto:' or 'javascript:' and those pointing to static file types like images, pdfs, fonts, etc. are also removed.

V. WEB SECURITY VULNERABILITIES

A. SQL Injection

SQL injection attacks are amongst the topmost threats in database-centric web applications and SQL injection vulnerabilities are one of the severe Vulnerabilities. SQL Injection permits the attacker to achieve control over the application's database. [9]

SQL injection can take place in URL, forms, headers or in cookies of any web page. Out of this, our model focuses on URL and forms present in a web page for vulnerability detection.

Depending on the payload used, it can be categorized into 4 major types, namely:

- Reflected or error based SQLi
- Boolean based SQLi
- Union based SQLi Blind SQLi

1) Reflected or error based SQLi:

Reflected or error based SQL injections are the most common type of attack. Error-based SQLi is a SQL Injection technique that relies on exceptions or errors thrown by the server. From the errors received from the server, one can infer the underlying structure of the database.

For detecting error-based SQLi, model performs following steps:

- Reflected SQL injection in URLs:

1) URL is taken as an input.The URL is then checked for the presence of query parameters.
2) If one or more query parameters are present then a special character such as single quote(') or double quote(") is appended to the value of query parameter in URL and it is sent to the server.
3) The contents of server response is then passed to the Machine Learning model to determine whether Reflected SQL injection is possible or not. 4) If no query parameter is found in the URL, then that URL is not vulnerable to Reflected SQL injection.

- Reflected SQL injection in Forms:

1) From the given URL, all the forms are extracted, if any.
2) For every input field present in each form, special characters such as single quote(') or double quote(") are inserted and submitted using the method given in the method attribute of form tag.
3) The contents of the server response is then passed to the Machine Learning model to determine whether Reflected SQL injection is possible or not.

2) Boolean based SQLi:

Boolean based SQL injection is a SQL injection technique that depends on sending an SQL query to the database which results in either TRUE or FALSE, depending on that, the content of HTTP response will change, or it will remain the same.

For detecting Boolean-based SQLi, model performs following steps:

- Boolean based SQL injection in URLs:

1) The list is prepared containing Boolean-based payloads which are grouped in such a manner that each group has payloads, '" or 1=1', '" or 1=1" and ' or 1=1'.
2) URL will be taken as input. Then query parameters are checked in the URL, if any present.
3) Each payload from the list is appended after the value of the query parameter and such a URL is sent to the server.
4) For each payload, contents of server responses are stored in their respective lists.
5) Then we compare these lists with each other and if one of the lists has different contents than the other two, then we can infer that, given query parameter is vulnerable to the respective payload.

- Boolean based SQL injection in forms:

1) The list is prepared containing Boolean-based payloads which are grouped in such a manner that each group has payloads, '" or 1=1', '" or 1=1" and ' or 1=1'.
2) From the given URL, all the forms are extracted, if any.
3) For all input fields present in the form, each payload from the list is inserted and submitted using the method given in the method attribute of the form tag.
4) For each payload, contents of server responses are stored in their respective lists.
5) Then these lists are compared against each other and if one of the lists has different contents than the other two, then we can infer that, given form is vulnerable to the respective payload.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 8, Issue 8, August 2021

3) Union based SQLi:

Union-based SQLi is a SQL injection technique that contains the UNION SQL operator that merges the results of two or more SELECT statements into a single result, then returned as part of the server response.

For detecting Union based SQLi, model performs following steps:

- Union based SQL injection in URLs:

1) The CSV file is prepared which contains payloads specific to Union based injections with labels '1' and '0', where '1' signifies that SQL injection is possible with that payload and '0' signifies that SQL injection is not possible.
2) The input URL is checked for the presence of query parameters.
3) If one or more query parameters are present, then special characters such as Single quote('), Double quote(") and Backtick(') along with their URLencoded versions are appended to the value of each query parameter and such a URL is sent to server.
4) The special characters for which error statement is present in the server response, then payloads with those special characters are labelled as '1' and rest of the payloads are labelled '0'. 5) The machine learning algorithm is trained using CSV file.
5) Testing dataset is prepared and passed to the machine learning model for predictions.
6) The payloads in the dataset classified as '1' are then sent to the server and if contents of base URL response are present in the received response, then SQL injection is achieved.

- Union based SQL injection in forms:

1) The forms are extracted from the input URL, if any present.
2) For the input fields present in the form, special characters such as Single quote('), Double quote(") and Backtick(') along with their URL-encoded versions are inserted and submitted using the method given in the method attribute of the form tag.
3) The special characters for which error statement is present in the server response, then payloads with those special characters are labelled as '1' and rest of the payloads are labelled '0'.
4) The machine learning algorithm is trained using CSV files.
5) Testing dataset is prepared and passed to the machine learning model for predictions.
6) The payloads in the dataset classified as '1' are then sent to the server and if contents of base URL response are present in the response received after submitting the form, then SQL injection is achieved.

B. Cross Site Scripting

Cross Site Scripting is a severe vulnerability that hampers security of a web application. Cross Site Scripting attack is an injection of harmful JavaScript code into the web application by the attacker in the client-side within the user's browser or in the server-side within the database, this JavaScript code is inserted within distrustful input data on the web application [8].

Many applications provide the facility to search for specific content. Whenever the user searches for the required content, the relevant results are displayed on the webpage along with a search keyword entered by the user.

XSS can take place in URL, forms, headers or in cookies of any web page. Out of this, our model focuses on URL, forms and cookies present in a web page for vulnerability detection. For performing XSS we follow the procedure mentioned below:

- Cross Site Scripting in URLs:

1) The CSV file is prepared which contains payloads specific to XSS with labels '1' and '0', where '1' indicates, XSS is possible with that payload and '0' indicates, XSS is not possible.
2) First input URL is taken and checked, if the query is present or not.
3) If query parameters are present, the word like 'l3333t' is given as the value of the query parameter and now it is sent to the server.
4) In the contents of the server response, we search for the same word. If the word is found, that URL is appended to a list. Basically here we are checking if the searched word is reflected in the HTML response or not.
5) From the list of URLs obtained in step 3), the URL is taken at a time, the value of the query parameter is now replaced with each special character, those are specific to Javascript.
6) The characters which are reflected in the server response, labels of those payloads with that specific character are changed to '1' and rest of the payloads are labelled '0'.
7) With this CSV, we train the Machine Learning model.
8) Testing dataset is passed to the machine learning model for predictions. The payloads classified as '1', are the most likely ones for XSS attack.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 8, Issue 8, August 2021

- Cross Site Scripting in Forms:

1) From the given input URL, all the forms are extracted, if any present.
2) In each input field of the form, a specific word like 'l3333t' is inserted and submitted using the method attribute specified in the form tag.
3) The word given as input, i.e., 'l3333t' is searched in server response and if found, that form is appended to a list.
4) For training the machine learning model, we use the same CSV file as used in the machine learning part of URL.
5) Trained machine learning model is then used in prediction of test data.The payloads with label as '1', are the most likely ones for XSS attack.
6) If the list of URLs obtained in step 3) of the URL module is empty, then in the input fields of form, special characters are inserted one by one and submitted to the server.
7) If Special characters are present in the contents of server response,then the labels of those payloads are changed to '1' and rest of the payloads are labelled '0'.
8) Now with this CSV we train the machine learning model.
9) The test CSV file is given to a machine learning model for predictions and the payloads labelled as '1' are most probable ones for XSS attack.

- Cross Site Scripting in Cookies:

1) For the input URL, a dictionary is created which contains the URL along with its cookies.
2) The value for the cookie is a word like 'l3333t' and the input URL is sent to the server along with this cookie.
3) In the response from the server, we find the word 'l3333t' and if it's present we append that cookie to a list.
4) The list of cookies we got from step 3), for each cookie, now the value for the cookie will be special characters and it is submitted to the server with a URL.
5) If the special character is present in the contents of the response from the server, then the payloads containing that special character will be changed to '1' and the rest will be '0'.
6) The machine learning model will be trained on a CSV file and the model test data will be sent.
7) The payloads classified with '1' are most likely to achieve XSS attack.

VI. MACHINE LEARNING ALGORITHM

A. Logistic Regression

Logistic regression is a statistical technique for analyzing a dataset that predicts the probability of an outcome that can only have two values. The goal is to find the best fitting model to describe the relationship between a set of dependent variables and a set of independent variables (predictor or explanatory variable). In logistic regression, the dependent variable is binary in nature, i.e. having two categories. Independent variables can be continuous or binary in nature.

Logistic regression is used in XSS detection and it is implemented in the following manner:

1) An independent variable, X, is defined as a set of XSS payloads along with a set of some ordinary statements.
2) A set of classes, y, with values '0' and '1', where '1' represents payloads that are most likely responsible to trigger XSS attack and '0' represents the ordinary statements.
3) Tokenization technique is used for converting text strings present in X into numeric form.
4) The training dataset, D, is built which contains a pair of (X_i,y_i) where, X_i represents tokenized XSS payload and y_i as its class.
5) Now, this training dataset, D, is used to train the logistic regression classifier.
6) In case of XSS detection, the classifier would be used to identify statements that could be either XSS payloads or the ordinary ones.

B. Naïve Bayes Classifier

Na¨ıve Bayes is a probabilistic classification algorithm based on Bayes Theorem and the Maximum A Posterior hypothesis. Bayes' theorem provides the relationship among the probabilities of 2 events with their conditional probabilities. Na¨ıve Bayes makes an assumption that the effect of an attribute value on a given class is independent of the values of other attributes.

Naïve Bayes classifier is used in detection of Reflected SQL injection, it is implemented in the following manner:

1) An independent variable, X, is defined as a set of generic SQL error statements with ordinary statements.
2) A set of classes, y, with values '0' and '1', where '1' represents statements that are most likely to be SQL error statements and '0' represents the ordinary statements.
3) Count-vectorizer and Tf Idf-transformation is used to convert text strings to numeric form.
4) The training dataset, D, is built which contains a pair of

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 8, Issue 8, August 2021

(X_i, y_i) where, X_i represents tokenized SQL error statements and y_i as its class.

5) This training dataset, D, is used to train Na¨ıve Bayes Classifier.
6) In case of reflected SQLi detection, the classifier would be used to identify statements that could be either SQL error statements or the ordinary ones.

VII. OTHER SPECIFICATIONS

A. *Advantages*

- Provides facility for automated and fast crawling.
- Comprehensive analysis of SQL Injection and XSS vulnerabilities.
- Easy to use GUI.

B. *Limitations*

- Only non-CAPTCHA registrations and logins can be carried out.
- Possible to detect first-order SQL Injection and XSS vulnerabilities.
- For the large Web application, stack overflow may happen.

C. *Applications*

With the proposed system, vulnerabilities present in the Web application can be detected.

VIII. RESULTS

Proposed model was successful in detecting SQL injection and Cross site scripting vulnerabilities in the particular realworld web applications.

For testing purposes, we used http://testphp.vulnweb.com/. This website is made intentionally vulnerable for testing purposes. It was chosen because it is built in PHP and PHP still continues to be the most common language used for web application development. As this website has known vulnerabilities, our goal was to find all of them and also try to find those vulnerabilities which are not yet known.

Following table summarizes the results of testing the model on various websites and vulnerable labs:

Web Application (to be tested)	URLs found	Pages vulnerable to SQLi	Pages vulnerable to XSS
http://testphp.vulnweb.com/	35	18	2
https://<redacted>.web-securityacademy.net/ (SQL Injection labs)	27	22	-
https://<redacted>.web-securityacademy.net/ (XSS labs)	30	-	12

IX. CONCLUSION AND FUTURE SCOPE

We have found the common vulnerabilities present in the web application, such as Error based SQLi, Union based SQLi, Boolean based SQLi and Cross Site Scripting. We proposed a system that will crawl the entire web application, scan different types of vulnerabilities, and generate a report specifying an overview of the detected vulnerabilities. Instead of conventional programming, we have used Machine learning algorithms for detecting the vulnerabilities.

There is a scope of improvement in some modules of the developed system. More vulnerabilities can be incorporated to further increase the scope of scanning. By adapting some techniques like file management, the limitation of large web application crawling can be eliminated. Furthermore, new features can be added to make the analysis of reports more understandable.

X. ACKNOWLEDGMENT

We would like to express sincere gratitude towards our respected project coordinator Prof. Shubhada Mone, Department of Computer Engineering, for guiding and supporting us for successful completion. We thank our guide for providing us a helping hand whenever needed and clearing our doubts. We thank all the staff members for their mentoring and valuable advice. With respect and gratitude, we would like to thank all the people, who have helped us directly or indirectly.

REFERENCES

[1] https://portswigger.net/daily-swig.
[2] https://www.websitehostingrating.com/ internet-statistics-facts/.
[3] Stefano Calzavara, Mauro Conti, Riccardo Focardi, Alvise Rabitti, and Gabriele Tolomei. Machine learning for web vulnerability detection: The case of cross-site request forgery. *IEEE Security & Privacy*, 18(3):8–16, 2020.
[4] Hoang Viet Long, Tong Anh Tuan, David Taniar, Nguyen Van Can, Hoang Minh Hue, and Nguyen Thi Kim Son. An efficient algorithm
[5] and tool for detecting dangerous website vulnerabilities. *International Journal of Web and Grid Services*, 16(1):81–104, 2020.

[6] Dimitris E Simos, Jovan Zivanovic, and Manuel Leithner. Automated combinatorial testing for detecting sql vulnerabilities in web applications. In *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*, pages 55–61. IEEE, 2019.

[7] Anastasios Stasinopoulos, Christoforos Ntantogian, and Christos Xenakis. Commix: automating evaluation and exploitation of command injection vulnerabilities in web applications. *International Journal of Information Security*, 18(1):49–72, 2019.

[8] TIAN Xiaopeng and TANG Di. A distributed vulnerability scanning on machine learning. In *2019 6th International Conference on Information Science and Control Engineering (ICISCE)*, pages 32–35. IEEE, 2019.

[9] Xun Zhang, Jinxiong Zhao, Fan Yang, Qin Zhang, Zhiru Li, Bo Gong, Yong Zhi, and Xuejun Zhang. An automated composite scanning tool with multiple vulnerabilities. In *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pages 1060–1064. IEEE, 2019.