Detection of Web Cross-Site Scripting (XSS) Attacks

D.Yashwanth Reddy-103 , D.Vignsehwar Reddy-105 , E.Laxmi Priya-106, E.Surya Prakash-107

Guide: Prof. R Sivasubramanian, Hod: Dr.Md.Thayyaba Khatoon

Department of AIML, School of Engineering, MALLA REDDY UNIVERSITY,500100

Abstract: Most applications looking for XSS vulnerabil ities hav e a variety of weaknesses related to the nature of constructing internet applicatio ns. Existing XSS vulnerability packages solely scan public net resources, which negatively influences the safety of internet resources. Threats may be in non-publ ic sections of internet resources that can only be accessed by approved users. The aim of this work is to improve av ailabl e internet functions for prev enting XSS assaults by creating a programme that detects XSS vulnerabilities by completely mapping internet applicatio ns. The innovation of this work lies in its use of environment-friendly algorithms for locating extraordinary XSS vulner abilities in addition to encompassing pre-approved XSS vulnerability scanning in examined internet functions to generate a complete internet r esource map. Using the developed programme to discover XSS vulnerabilities incr eases the effectiveness of internet utility protection. This progr amme also simplifies the use of internet applicatio ns. Even customers unfamiliar with the fundamentals of internet security can use this programme due to its capabil ity to generate a document with suggestions for rectifying detected XSS vulnerabilities.

Keywords: XSS vulnerabilities; XSS; web security; web attacks

1. Introduction

Ensuri ng inform ation security (I S) in com puti ng systems is a priority for organisations that use algorithm s to collect, process, store and transmit inform ation. How ev er, with the widespread use of the internet, many threats to IS have emerged. Most web applications used over the previous decade were static and lacked interactive user interfaces and thereby had no exploitable vulnerabilities [1,2]. As a result, many developers ignored web application security issues at the time. Whilst a large number of dynamic websites that utilise modern technologies to connect users to web applications and enhance their interactions with web resources (e.g., bulletin boards and feedback forms) have been introduced in recent y ears, these i nnov ations hav e v ulnerabilities that allow i ntruders to perf orm com puter attacks, such as SQ L-i njection and cross-site scripti ng (XSS) [3]. With the help of injected code, an intruder can gain unauthorised access to user data, which could allow them to impersonate these users, perform illegal actions on the local computers of users and the network equipment of their companies or change the configurations of their network and software. The lack of proper measures for ensuring IS has resulted in the emergence of computer attacks linked to malicious code execution [4].

According to the Open Web Application Security Project (OWASP), cross-site program- ming is one of the most common types of computer attacks [5]. Around 65% of websites are exposed to XSS vulnerabilities detected in current web applications [6], as shown in Figure 1.

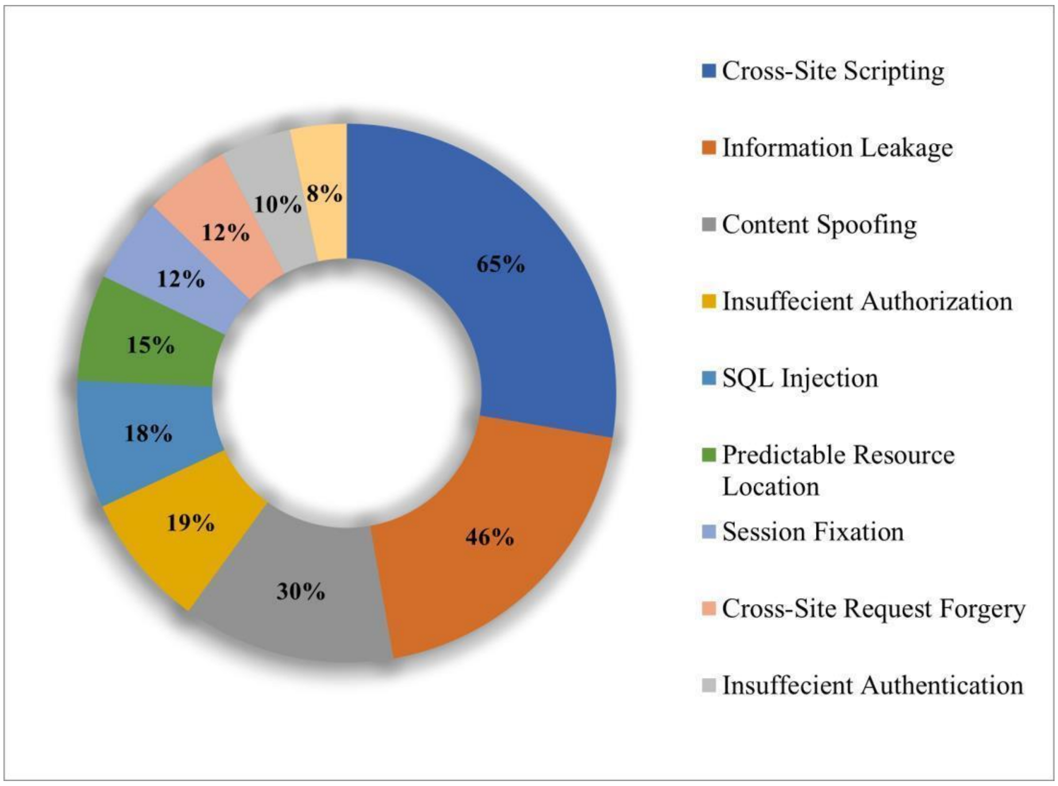


Figure 1. Percentage of websites exposed to various cyber vulnerabilities.

As shown in Figure 1, it is clear that information leakage is the second biggest threat, with 46% of websites exposed to this type of cyber vulnerability.

Recently, poorly developed software programs have been identified as safety threats. The number of viruses to which a system is potentially exposed is directly associated with the size and complexity of the established net packages and servers. Most multifaceted applications have both extensive loopholes and a few already identified weaknesses. In ad- dition, net servers are intrinsically multifaceted applications. Websites are also multifaceted and even intentionally request extra data from users. Cross-site web page scripting (XSS) is one of the riskiest and most-exploited assaults in current times. Almost 65% of web- sites have been identified to have one or more of the XSS vulnerabilities listed in current net packages.

Cross-site attacks, where users inject payloads (malicious code) in the customer section of a website, are the mostcommon network attacks via the web. Weak spots found in poorly encrypted websites are exploited by attackers, using the victim's browser to send malicious text from vulnerable sites. The types of attacks exploiting XSS vulnerabilities are presented in the next section, followed by examplesof XSSvulnerability detection software in Section 3. Then, the evaluation results are discussed in Section 4. Finally the conclusionis presented in Section 5.

2. Types of Attacks Exploiting XSS Vulnerabilities

XSS is a computer attack that involves injecting malicious code into the webpage parameters sent to a web browser. This computer attack is similar to SQL-injection [7] and can be used in various ways [8]. XSS attacks can be classified according to the vector and the method of influence, as shown in Figure 2.

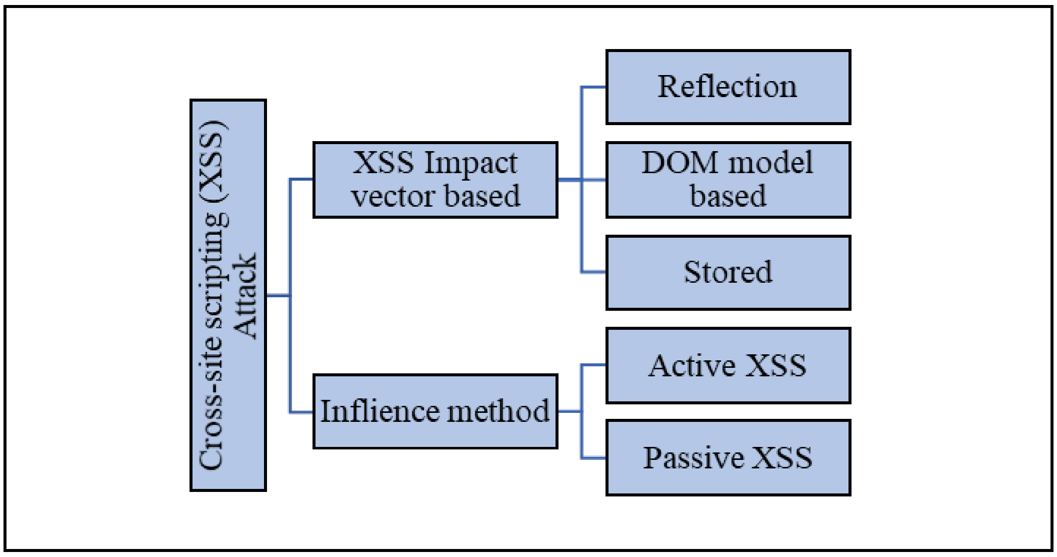


Figure 2. Classification of XSS attacks.

Based on the XSS impact vector, an XSS attack can be classified as follows:

- Active XSS, which does not require additional actions on part of the user in terms ofthe functionality of web applications.
- Passive XSS, which is triggered when a user performs a certain action, such as clickingor hovering the cursor.
- Reflected XSS.
- Stored XSS.
- XSS based on the document object model (DOM).

2.1. *Reflected XSS*

Reflected XSS attacks are the most common type of XSS attack, and the vulnerabilityexploited by this attack is easy to detect. In a reflected XSS attack, the code is sent to theserver and returned to the user within a single HTTP request. The XSS using a reflected XSSvulnerability follows a complex process wherein the malicious code is injected into a URL, this URL is sent to the user, and the user must click on this URL to run the malicious code. However, such complexity does not present an obstacle for attackers. A basic reflectedXSS attack does not aim to steal confidential information. When a user visits a website with a reflected XSS vulnerability, a warning window is displayed after executing thescript code. A link to a webpage with a basic reflected XSS vulnerability resembles thefollowing: http://site.com/<script>alert(\T1\textquotedblleftXSScompletedsuccessfully\T1\textquotedbl right)</scri pt>) . An advanced internet user would find this link strange and would refuse to proceed to the directed webpage. One method for hiding such a malicious link is RFC 1738 encoding [9]. In the following code snippet, after encoding a malicious URL, the script code becomes unrecognisable:

- Link: http://site.com/<script>alert('attack')<script>
- Link after encoding: http:%3b%2a%2asite.com%2a%3xscript%3balert(%27attack%27)% 3xsencod%3b

Attackers combine a reflected XSS vulnerability with other types of attacks to boost the effect of exploiting such a vulnerability. This approach has resulted in a new type of com puter attack called clickjacking, w hich tricks a person into clicki ng a link that is i nvisible or disguised as some other element [10,11]. Clickjacking has the following features:

- Link theft occurs when a user clicks on a malicious link or button generated by an attacker.
- Clickjacking does not need injected scripts, hence facilitating the implementation of the attack.

- This type of attack is based on dynamic HTML.
- This attack is easy to perform.

Clickjacking itself has led to another type of computer attack that combines JavaScript (JS) with HTML Iframe tags. IS specialists call this computer attack cross-frame scripting (XFS), and it loads legitim ate pages to steal user data. An XFS attack is only successful when combined with social engineering. Consider a scenario where the attacker, by using social engineeringmethods, convinces a user to go to a specially crafted page. After visiting this page, m alicious J S codes and Ifram e tags are loaded i nto the w eb browser of the user. W hen a user enters his/her credenti als in an If ram e pointi ng to a l egitim ate site, the m alicious JS code steals his/her keystrokes.

Redirection XSSis another type of XSS attack that exploits the reflected XSSvulnera- bility of web applications. In this type of attack, malicious code is downloaded from an untrusted source and injected into various web application scripts.

In a redi rection XSS attack, data from an unreliabl e source enter a w eb resource in the form of w eb requests. T he receiv ed data are then w edged into the dy namic content of the web application and sent to the user without checking for malicious codes.

The malicious content sent to the web browser of a user often takes the form of JS code, but m ay also include H TM L, Flash or any other code that can be processed by a w eb brow ser. Com puter attacks using stored XSS vul nerabilities are m ore dangerous than those that use reflected XSS vulnerabilities given that the former can be used for a long time.

2.2. *Stored XSS Attack*

Stored XSS attacks are m ai nly used to steal sessi on inf orm ation (e. g., cooki es), redi rect users to a m alicious w eb resource or allow attackers to perf orm m alicious operations on the PC of the user. T wo of the m ost dangerous com puter attacks that expl oit a stored XSS vulnerability in a web application are cookie theft and Trojans.

Cookies are small pieces of data that are stored in the web browser of users and include user data and other inf orm ation that m ay be utilised to recognise a user w henev er s/he accesses a web application. Cookies are generated on the web server side, sent to the web browser of users and stored in the local storage of this browser. The next time a user accesses this w eb applicati on, the sav ed cookies will be sent to his/her w eb serv er to recognise the user and to grant the attacker access to certain w eb resou rces based on the information stored in these cookies [12,13].

Attackers m ay also use an XSS v ulnerability to trick users into dow nloading a Trojan horse for carryi ng out a long-term computer attack. I S speci alists call this com puter attack an XSS-based Trojan horse [14]. To carry out this type of attack, the attacker needs the user to download a Trojan programme by using a stored XSS vulnerability in his/her web application. Malicious code is injected into the area of the web application with a stored XSS vulnerability to download a Trojan programme. Anyone who visits this page will automatically follow a link with a malicious JS code, which will be processed by their browsers and downloaded onto their computers. In this case, XSS-based Trojan horse attacks generally go unnoticed because this type of attack users Iframe to create a child window whose size is set to 0.

2.3. *XSS Attack Based on DOM*

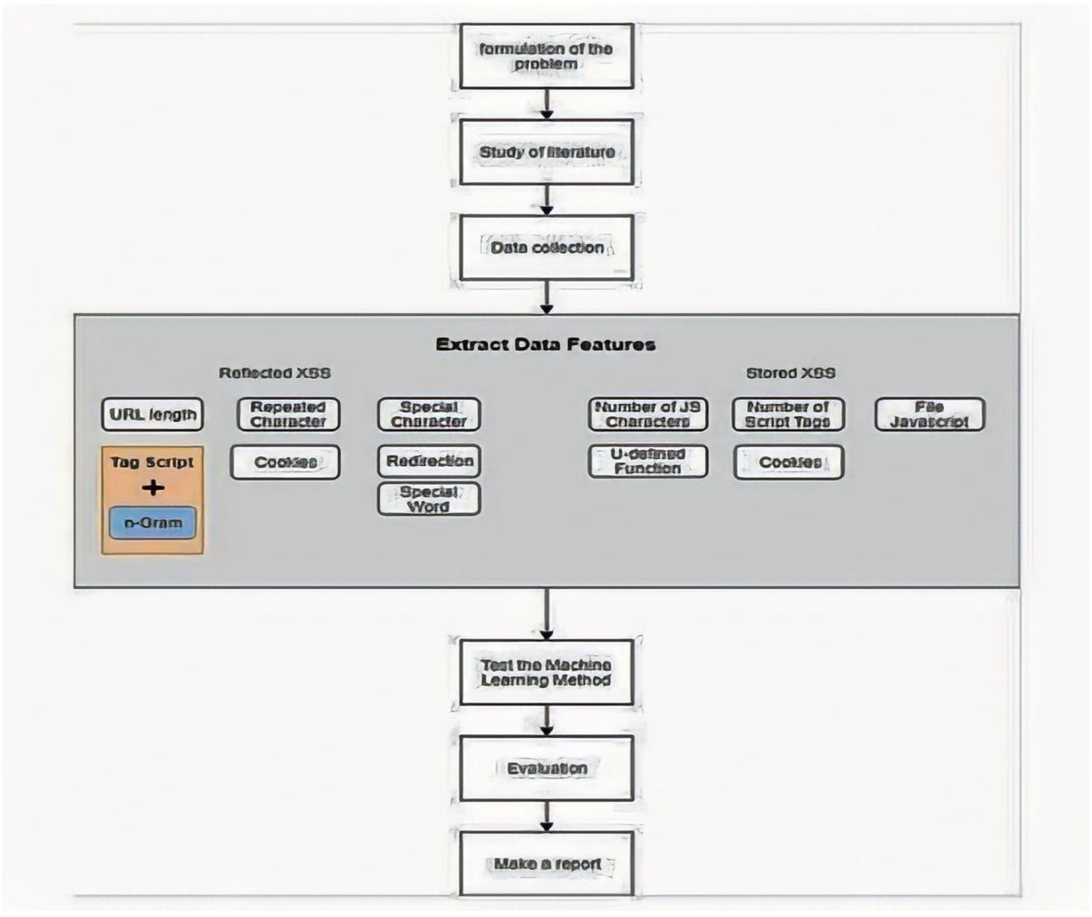
By usi ng a J S script, an XSS attack based on DOM exploits an XSS v ulnerability in the DOM that occurs on the side of the user during data processing. As implied in its name, this type of XSS attack is implemented through DOM, a platform - and language-unbiased program ming interf ace that gives applications and scripts access to the contents of HTML and XML docum ents and m odifies thei r content, shape and execution [15]. With incorrect filtering, the DOM of the attacked site can be modified, and the malicious JS code can be executed in the attacked site.

XSS vulnerabilities can be detected using several methods, including BeEF, Xenotix - XSS, Acunetix, XSpider-MAX-Patrol, Nemesida-Scanner and Wapiti.

XSS attacks based on DOM send a trusted-server-managed script to the user of a web application. T ake, for exam pl e, a J S code that checks the com pl eteness of a f orm bef ore being sent out to the server. The script processes the entered data and inserts them back to the webpage (e.g., via dynamic HTML), hence allowing attackers to 'wedge' a malicious code into a JS script.

In summary, existing information security tools provide a variety of algorithms and programs that can block various information security threats. Most vulnerability tools are heavy and provide redundant features that can slow computer systems and increase resource utilization. Further, there are downsides to simple solutions that are simply "tuned" to find XSS vulnerabilities. The weakness of XSS trackers is related to its ability to write web applications. Most web applications license web resources to increase user ri ghts. T his m eans thatallow ed users can m ore easily access the f eatures of w eb resources than non-privileged users.

ARCHITECTURE



3. XSS Vulnerabilities Detection Software

Modern data protection softw are and equipm ent utilises a wi de range of al gorithms and programmes to forestall data security threats [16]. However, most of this software and equipment has cumbersome operation and redundant performance that can slow the devices of users and increase their resource consum ption. M eanw hile, solutions that can easily detect XSS vulnerabilities have other drawbacks associated with the peculiarities of constructing web applications. Specifically, most web applications authorise internet resources to expand consum er privileges; that is, authorised users are giv en more access to the functionalities of a web resource compared with unauthorised users.

The existing softw are for detecti ng XSS v ulnerabilities, such as the O nline W eb Security Scanner, search for these vulnerabilities only in the open parts of websites that do not requi re access authorisation [17– 21]. Such a draw back is si gnificant consi dering that XSS vulnerabilities may be located in an unsearchable part of a web resource.

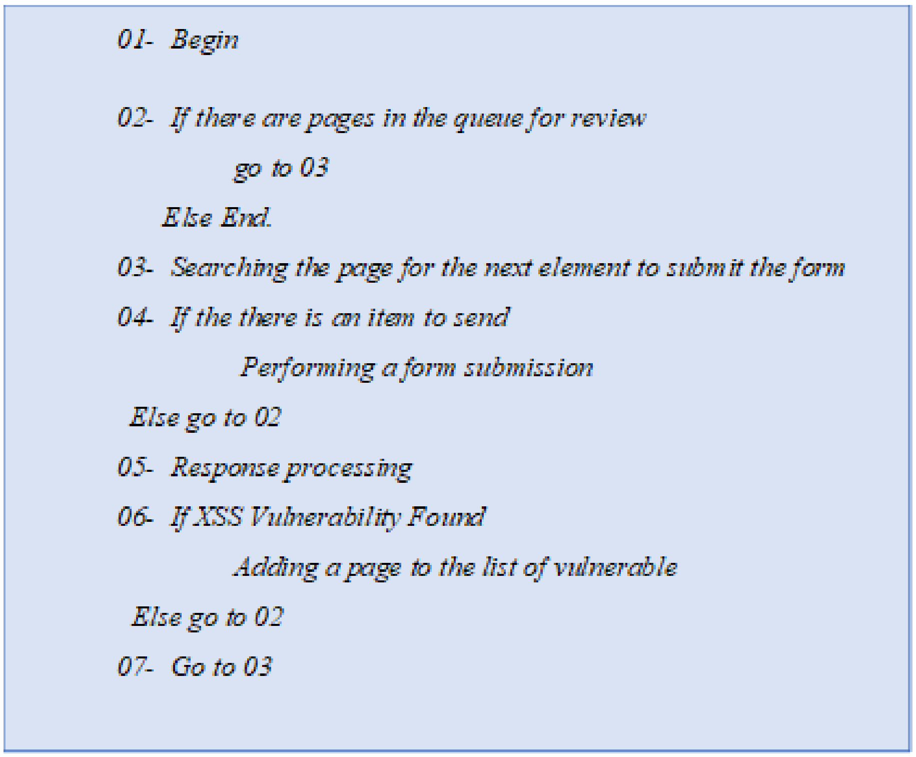
Therefore, applications that search for the XSS vulnerabilities of a web application are urgently needed. To fulfil such demand, this paper designed a programme based on the symmetrical utility of most high-quality algorithms for discovering XSS vulnerabilities. The Delphi programming language was used during programme development. The proposed programme implements the following functions:

- Detection of all types of XSS vulnerabilities (including reflected, stored and DOM-based XSS);
- Pre-authorisation in web applications and cookie storage;
- Compilation of internal URLs in a web application;
- Creating reports on the detected vulnerabilities; and
- Recommending the necessary actions for the detected vulnerabilities.

Figure 3 presents the search algorithm used for reflected XSS vulnerabilities. Another possibility is an attack on insufficiently processed data from the HTTP re-
sponse. M alicious code in the case of reflected XSS is only em bedded in the HT T P response, not stored in the application. In this case, it may occur that malicious code to run it is included in the response.

Given that reflected XSS vulnerabilities are only observed when submitting forms, the above algorithm triggers the submission of these forms usingthe POST method, which involves inserting values in the sent elements and receiving a response in the form of an HTML message. The incoming message is then analysed for vulnerabilities as follows:

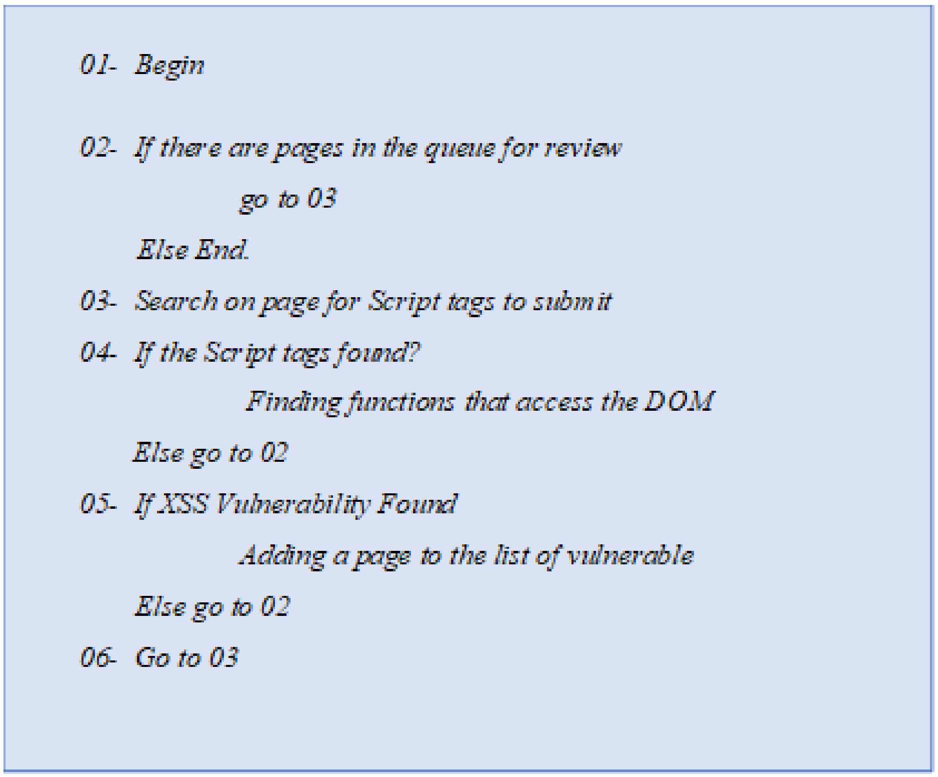
If the inc o ming JS co de sets the value sto red in the document object mo del to a true value (do cument. vulnerab le = true), then the page is mark ed as containing a potential threat o f the corresponding type; otherwise, the page is marked as safe and is not added to the final list of vulnerabilities.



```
01- Begin

02- If there are pages in the queue for review
            go to 03
    Else End.
03- Searching the page for the next element to submit the form
04- If the there is an item to send
            Performing a form submission
  Else go to 02
05- Response processing
06- If XSS Vulnerability Found
            Adding a page to the list of vulnerable
  Else go to 02
07- Go to 03
```

Figure 3. Search algorithm for reflected XSS vulnerabilities.

Figure 4 presents the proposed search algorithm for XSS vulnerabilities based on DOM.



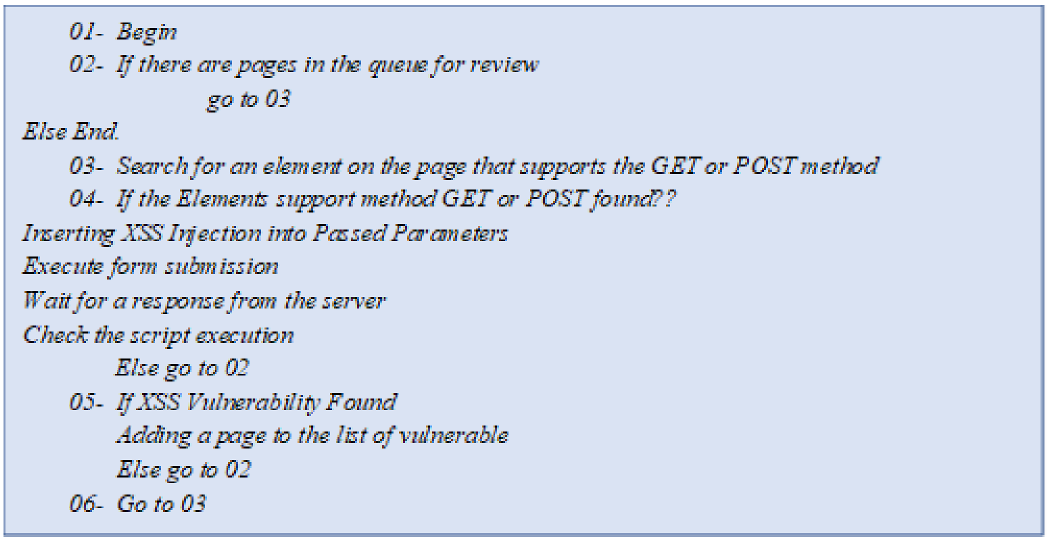
```
01- Begin

02- If there are pages in the queue for review
            go to 03
    Else End.
03- Search on page for Script tags to submit
04- If the Script tags found?
            Finding functions that access the DOM
    Else go to 02
05- If XSS Vulnerability Found
            Adding a page to the list of vulnerable
    Else go to 02
06- Go to 03
```

Figure 4. Search algorithm for XSS vulnerabilities based on DOM.

During execution of the proposed search algorithm for XSS vulnerabilities based on DOM, the al gorithm analyses the current page code including the hi dden scri pts in H TML tags. Then, the software searches this information for scripts that are used for calls to DOM methods. Calls to DOM methods are similar to the following:

- Write net HTML;
- Steer amendm ent of document models (dynamic HTML events are included); and
- Steer explicit execution of scripts.

Figure 5 shows the proposed algorithm used to search for stored XSS v ulnerabilities. Stored XSS occurs when a web application receives data from an untrusted source and then incl udes this data in its later HTT P responses. These m alicious data or scri pts can be placed, for example, in a comment next to a post, in a message forum, in a visitor's log or in other pl aces that are visi ble to other visitors. T his al gorithm has unique characteristics giv en that, unlike reflected XSS v ulnerabilities, stored XSS v ulnerabilities are the result of saving scripts in a database. This operation must be carri ed out using a preliminary PO ST request to prev ent m alicious codes f rom modifying the database. T his al gorithm is similar to that used for detecti ng reflected XSS v ulnerabilities, except the form er requi res the user to submit the form and wait for a response from the server.



```
    01-  Begin
    02-  If there are pages in the queue for review
                   go to 03
Else End.
    03-  Search for an element on the page that supports the GET or POST method
    04-  If the Elements support method GET or POST found??
Inserting XSS Injection into Passed Parameters
Execute form submission
Wait for a response from the server
Check the script execution
         Else go to 02
    05-  If XSS Vulnerability Found
         Adding a page to the list of vulnerable
         Else go to 02
    06-  Go to 03
```

Figure 5. Search algorithm for stored XSS vulnerabilities.

After the script is executed, the form and XSS injection are stored in a special structurealong with other information pertaini ng to the detected vulnerability. After the script is executed, the search is terminated because when the form is submitted, the script located inthe web application database will be executed. Therefor e, the scan must be perform ed again after the vulnerability is fixed. Figure 6 presents the algorithmof the developed program m e.

In summ ary, as a v ery urgent mission, softw are that searches for XSS v ulnerabilities by analysing the full map of the w eb page, includi ng the hidden scri pts, is dev eloped. The proposed software is based on consistent application of the most effective algorithmsto detect various types of XSS vulnerabilities, as presented in this section.

```
01- Begin

02- Program configuration
03- Entering a list of pages to search
04- Search for potential XSS vulnerabilities
    Search for "Reflected XSS"
    Search "XSS based on the DOM model"
    Search for "Stored XSS"
05- If XSS Vulnerability Found
        Preparing and generating a report
        Output of the report and final recommendations
        End
    Else go to 05
```

Figure 6. Algorithm of the programme developed for detecting XSS vulnerabilities.

4. Evaluation Results

The assessm ent criteri a w ere based on standards proposed by softw are security or- ganizations and researchers to help develop the basic tools used to assess web browser vul nerabilities. Open W eb Application Security O utlook (O W ASP) provides an up- to- date assessment of ten critical security threats that threaten internet application security today.

The evaluation experiment of cross-site scripting (XSS) attacks is based on an XSS impact vector and includes reflected, stored and DOM-based XSS.

The measurem ents for this experim ent are vulnerability detection rate based on the
number of detected vulnerabilities, average time spent finding vulnerabilities and total time spent in the search process. Si nce all the scanners in our study w ere eligi ble for PCI compliance, they were mandated to test for each vulnerability category of the Top Ten Open W eb Application Security Project (OW ASP). W e also exami ned each scanner's scan profile customization features to get a better idea of their target vulnerability categori es.

This study focuses on three types of cross-site scripts: XSS-type-1, XSS-type-2 and XSS- type-3. XSS-type-1 is comprised of samples of reflected-XSS scripts, executed via a <script> tag. XSS-type-2 is comprised of stored-XSS vulnerabilities, where untrusted user input is written to a database and then executes a script when read from the database. XSS- type-3 includes hidden scripts in HTML tags that make up reflected and stored XSS using non-standard tags and keywords, such as <style> and prompt().

In this study, data w ere entered into the programm e configuration section, if necessary. This was also searched for specific XSS vulnerabilities to save time. After the search was completed and if records in the internal structure matched the detected threats and their descriptions, a report on these vulnerabilities was generated. Table 1 presents an example of the generated report.

This report provides w eb dev elopers with com pl ete i nform ation about the w eb applica - tion being tested and proposes some recommendations for fixing each detected vulnerabil- ity, thereby greatly simplifying their work. The issuance of recommendations is considered an advantage of the developed programme over existing vulnerability-detection software.

Accuracy: 0.9766252739225713
Precision: 0.9882596685082873
Recall : 0.9682002706359946

Table 1. Sample of a generated report.

No	Registration Date	Title	URL	XSS Type	Verify
1	22.01.2022 15:04:46		http://localhost:7081/basic/web/index.php?r=request%2Fview&id=33	Yes	Yes
2	22.01.2022 15:04:47		http://1ocalhost:7081/basic/web/index.php?r=request%2Fupdate&	Yes	Yes
3	22.01.2022 15:04:47		http://localhost:7081/basic/web/index.php?r=request%2Fdel ete&id=33	Yes	Yes
4	22.01.2022 15:04:46	ID	http://localhost:7081/basic/w eb/index.php?r=request%2Findex&sort=id	Yes	Yes
			Discovered vulnerabilities		
	1		Image injection by embedding the javascript protocol		
	2		Complete absence of filtering on the server		
			Total found on page: 2		
5	22.01.2022 15:04:41	Auto-writing	http://localhost:7081/.	Yes	Yes
6	22.01.2022 15:04:41	Exit (admin)	http://localhost:7081/basic/web/index.php?r=site%2Flogout	Yes	Yes
7	22.01.2022 15:04:41	Main	http://localhost:7081/basic/web/	Yes	Yes
8	22.01.2022 15:04:46	Add a request	http://localhost:7081/basic/web/index.php?r=request%2Fcreate	Yes	Yes
9	22.01.2022 15:04:41	Applications	http://localhost:7081/basic/web/index.php?r=request%2Findex	Yes	Yes
10	22.01.2022 15:04:54	Applications	http://localhost:7081/basic/web/index.php?r=request%2Findex&sort=-id	Yes	Yes

Using the advanced software program to discover XSS vulnerabilities will be a boon to the effectiveness of shielding Web applications. In practice, the advanced application simplifies the checking of internet applications. Due to the capability of the report, technol- ogy with tips f or the remov al of detected XSS v ulnerabilities m akes it f easi ble to apply this system by customers who do not understand the fundamentals of records security.

To ev aluate the dev eloped programm e, its m ain characteristics w ere com pared with those of similar solutions, namely, BeEF, Xenotix-XSS, Acunetix, XSpider-MAX-Patrol, Nemesida-Scanner and Wapiti. The comparison results are presented in Table 2.

Table 2. Comparison of programmes for detecting XSS vulnerabilities.

Characteristics	BeEF	enotix- XSS	Acunetix	XSpider-MAX-Patrol	emesida-Scanner	Wapiti	Proposed Programme
Authorization in system	No	No	No	No	No	No	Yes
Stored XSS Search	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reflected XSS Search DOM	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Search XSS	Yes	No	Yes	No	Yes	No	Yes
Availability of GUI	Yes	Yes	Yes	Yes	Yes	No	No
Issue of recommendations	Yes	No	Yes	Yes	No	No	Yes

The developed program and the aforementioned solutions were tested on the Open Web Application Security Project (OWASP) Juice Shop, which is an internet resource that was specifically designed for training and testing software for detecting XSS vulnerabilities.

Juice Shop is a w eb application with a huge am ount of vul nerabilities and is activ ely supported by the worldwide information security community. The OWASP consortium also owns the project and distributes it under free licenses. It can be used in security training, demos, CTF competitions and security testing.

Juice Shop incl udes v ulnerabilities from the OW ASP TO P 10 list, as w ell as m any other vulnerabilities found in real applications.

The dev eloped program m es Acuneti x and XSpi der dem onstrated the best perf orm ance in detecting vulnerabilities, as shown in Figure 7. Despite detecting the same number of vul nerabilities as Acunetix and XSpi der, the dev eloped programm e spent 44% and 20% less time in detection compared with Acunetix and XSpider, respectively, as shown in Figure 8.

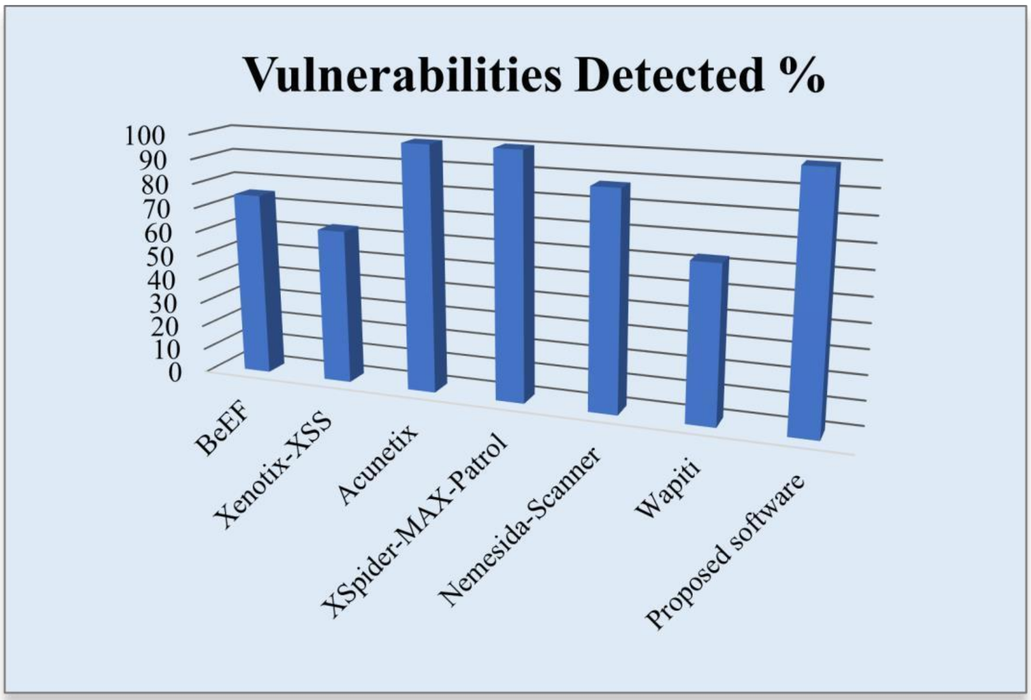


Figure 7. Number of vulnerabilities detected by each software.

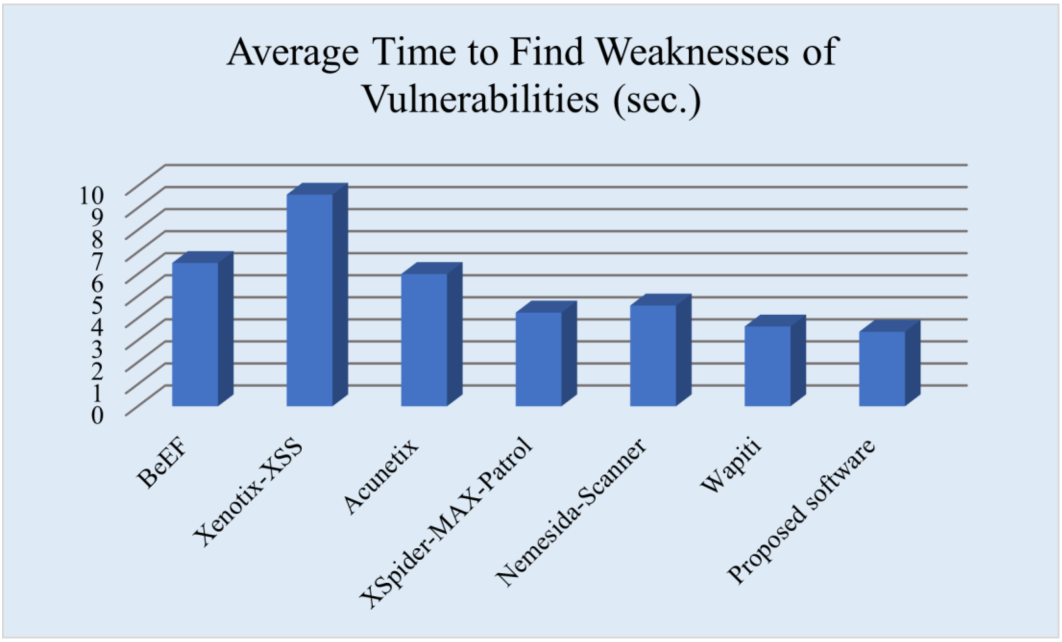


Figure 8. Average time spent by each software in finding vulnerabilities.

The developed programme also spent 25% and 20% less time searching compared with Acunetix and XSpider, respectively, as shown in Figure 9.

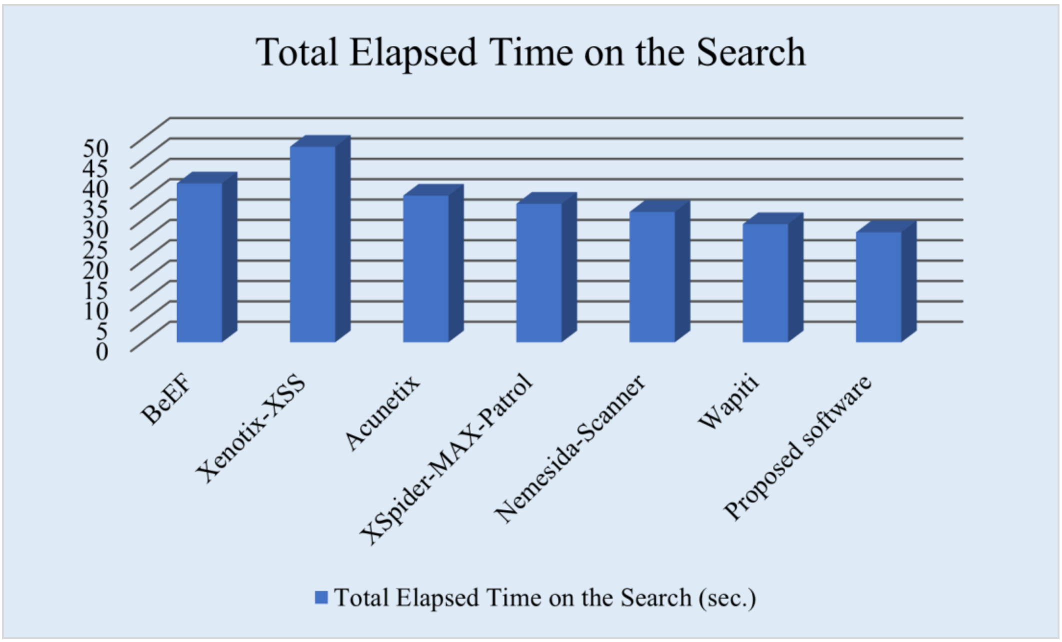


Figure 9. Totaltime spent by each software in the search process.

The above results highlight the following advantages of the developed programme:

- searches for XSS in closed parts of web resources;
- finds more XSS vulnerabilities compared with similar programmes;
- spends the least amount of time in detecting XSS vulnerabilities; and
- proposes recommendations for eliminating the detected vulnerabilities.

Furtherm ore, accuracy, precision and recall were m easured. More specifically, accuracy provides the percentage of the three ty pes of threats that w ere correctly detected. Precision determines the f raction of w eb pages correctly cl assified as XSS ov er all w eb pages cl assified as XSS vulnerabilities. Recall is the fraction of web pages correctly classified as XSS vulnerabilities over all web pages of XSS vulnerabilities. The proposed system has 99.47% accuracy, 100% precision, and 81% recall.

How ev er, this work is limited to only one ty pe of injection attack: XSS. I njection attacks are i ncl uded in the O W ASP T op 10 list, especially SQ Li (SQL i njection) and XSS, w hich are not only very widespread but also very dangerous, especially in older applications.

Future work will focus on others injection attacks and different injection detection tools that can be used to better understand vulnerabilities. Other injection attacks include code injection, and NoSQL.

5. Conclusions

Most applications that look for XSS v ulnerabilities hav e sev eral limitations attributabl e to the nature of building clean applications. Whilst existing XSS vulnerability packages merely scan the public areas of internet resources, vulnerabilities are usually stored in the non- public areas of these resources. The program m e dev eloped in this paper def ends internet functionalities against XSS attacks by detecting XSS vulnerabilities by completely mapping internet applications. This programme utilises the most environment-friendly fixed softw are- based al gorithmic methodology to detect unusual XSS v ulnerabilities. This programme also scans previously approved XSS vulnerabilities to generate a complete map of useful internet resources. Discov eri ng XSS vul nerabilities will increase the eff ective - ness of internet utility protection. The developed programme also simplifies the process of cleaning applications. Even customers unfamiliar with the basics of IS can use this programme thanks to its ability to generate documents and propose actions against the detected XSS vulnerabilities.

References

1. Mohammed, B.A.; Al-Mekhlafi, Z.G. Accuracy of Phishing Websites Detection Algorithms by Using Three Ranking Techniques. *IJCSNS* **2022**, *22*, 272.
2. Al-Mekhlafi, Z.G.; Mohammed, B.A.; Al-Sarem, M.; Saeed, F.; Al-Hadhrami, T.; Alshammari, M.T.; Alreshidi, A.; Alshammari, T.S. Phishing Websites Detection by Using Optimized Stacking Ensemble Model. *Comput. Syst. Sci. Eng.* **2022**, *41*, 109–125. [CrossRef]
3. Kaur, M.; Raj, M.; Lee, H.N. Cross Channel Scripting and Code Injection Attacks on Web and Cloud-Bas ed Applicat ions: A Comprehensiv e Review. *Sensors* **2022**, *22*, 1959.
4. Mohammed, B.A.; Al-Mekhlafi, Z.G. Optimized Stacking Ensemble Model to Detect Phishing Websites. In *International Conferenceon Advances in Cyber Security*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 379–388.
5. Wibowo, R.M.; Sulaksono, A. Web Vulnerability Through Cross Site Scripting (XSS) Detection with OWASP Security Shepherd. *Indones. J. Inf. Syst.* **2021**, *3*, 149–159. [CrossRef]
6. Gupta, B.; Gupta, S.; Gangw ar , S.; Kumar, M.; Meena, P. Cross-site scripting (XSS) abuse and defense: Exploitation on sev eral testing bed environments and its defense. *J. Inf. Priv. Secur.* **2015**, *11*, 118–136. [CrossRef]
7. Kasim, Ö. An ensembl e classification-bas ed approach to detect attack lev el of SQL injections. *J. Inf. Secur. Appl.* **2021**, *59*, 102852. [CrossRef]
8. Sarjitus, O.; El- Yakub, M. N eutr alizing SQL injection attack on w eb applic ation using serv er side code modification. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.* **2019**,*5* .
9. Yu, L.; Chen, L.; Dong, J.; Li, M.; Liu, L.; Zhao, B.; Zhang, C. Detecting malicious w eb r equests using an enhanced textcnn. I n Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applic ations Confer ence (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 768–777.
10. Cris¸an, A.; Florea, G.; Halasz , L.; Lemnaru, C.; Oprisa, C. Detecting malicious URLs based on machine learning algorithms and word embeddings. In Proceedings of the 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 3–5 September 2020; pp. 187–193.
11. Rev enkov, P.V.; Berdy ugin, A.A.; Makeev, P.V. Research on Brute Force and Black Box Attacks on AT Ms. 2021. Availabl e online: http://ceur-ws.or g/Vol-3035/paper17.pdf (accessed on 11 May 2022).
12. Rodríguez, G.E.; Torres, J.G.; Flores, P.; Benavides, D.E. Cross-site scripting (XSS) attacks and mitigation: A survey. *Comput. Netw.* **2020**, *166*, 106960.
13. Al-Mekhlafi, Z.G.; Mohammed, B.A. Using Genetic Algorithms to Optimized Stacking Ensemble Model for Phishing Websites Detection. In *International Conference on Advances in Cyber Security*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 447–456.
14. Barde, S. Blockchain-Based Cyber Security. In *Transforming Cybersecurity Solutions using Blockchain*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 55–69.
15. Da Rocha, H. *Learn Chart. js: Create Interactive Visualizations for the Web with Chart. js 2*; Packt Publishing Ltd.: Birmingham, UK, 2019.
16. Al-Sarem, M.; Saeed, F.; Al-Mekhlafi, Z.G.; Mohammed, B.A.; Al-Hadhrami, T.; Alshammari, M.T.; Alreshidi, A.; Alshammari, T.S.An optimized stacking ensemble model for phishing websites detection. *Electronics* **2021**, *10*, 1285. [CrossRef]
17. Ibarra-Fiallos, S.; Higuera, J.B.; Intriago-Pazmiño, M.; Higuera, J.R.B.; Montalvo, J.A.S.; Cubo, J. Effective filter for common injection attacks in online web applications. *IEEE Access* **2021**, *9*, 10378–10391.
18. Rao, G.R.K.; Satya Prasad, R. A Three-Pronged Approach to Mitigate Web Attacks. In *Advances in Smart System Technologies*; Springer: Berlin/Heidelb er g, Germany, 2021; pp. 71–83.
19. Gan, J.M.; Ling, H.Y.; Leau, Y.B. A Review on Detection of Cross-Site Scripting Attacks (XSS) in Web Security. In *International Conference on Advances in Cyber Security*; Springer: Berlin/Heidel ber g, Germany, 2020; pp. 685–709.