Automated Detection of Cross-Site Scripting in Websites

Ahmed E, Mohamed K, Hamdy N, Badreldin E, Osama R
Military Technical College, Cairo, Egypt,
ahmedehab25299@gmail.com, mshaqwieer@yahoo.com, hamdunasser22@gmail.com,
badr7189@gmail.com, osamaradwanborhan@gmail.com
Supervisor: Dr. Khaled Metwally
Military Technical College, Cairo, Egypt, k.metwally@mtc.edu.eg

***Abstract*—Recently, the fast emerging of 5G networks, the Internet of things (IOT), and cloud computing enhanced the user expectation towards the internet and fastened the digital transformation of most enterprises and leading entities. These emerging technologies led to the wide spreading of web-oriented and web-based services that are accessible over the internet from any place in the world. However, web applications are not always secure, and contain varies vulnerabilities. Many vulnerabilities exist in current web developing, one of most the famous vulnerability is the Cross-Site Scripting (XSS) which is a critical vulnerability that can lead to identity theft and/or data violation. Many research effort has been carried out in this field addressing the detection accuracy. This study aims to automate XSS vulnerability detection in web applications with an improved detection accuracy over that of traditional human penetration testing operations. This research main objective is to ensure and maintain the developed web application's security against unwanted attacks.**

***Index Terms*—xss, vulnerability, automation**

I. Introduction

Nowadays, the fast growing in web applications usage across the world, makes most of our valuable and vital data stored in databases and available online. This gives a security awareness to web users in order to secure their data from possible theft due to existing vulnerabilities in these developed web applications. Moreover, web developers with the help of the recently developed security tools and frameworks enhanced their web security knowledge in developing secure web applications that harden the scanning process on the hackers. This pushes hackers to search and invest more in developing difficult penetration methods that increase the vulnerabilities numbers and criticality. As a result, user's data is exposed to more vulnerabilities and makes cybersecurity engineers' jobs much harder to find ways to block these new developed methods. Cross-site scripting (XSS) is one of these vulnerabilities that this project is focused on it. XSS is one the most famous and critical vulnerabilities, it also counted as one of the top ten list of the highest rated vulnerabilities published by the Open Web Application Security Project (OWASP).

II. Goals

This project aimed to develop an automated tool that automatically detects and exploits reflected XSS vulnerabilities [1] to help the security officers and penetration testers in two aspects; in defense and offense. In the first one during the defense, the tool helps the user to check his website for XSS vulnerability before deployment, while during the offence process, the tool allows the user to test and check any website for the XSS vulnerability.

III. Methodology

The idea behind the reflected XSS vulnerability is explained in Figure (1). [4] The malicious code is prepared that enable the attacker to view users' session data is embedded into a hyper link that's sent to any web site visitors. Once the link sent to the victim via email and the victim clicked on it, the script is executed by the web application and reflected back to the victim's browser. The browser then sends the session cookies to the attacker, enabling access to victim's private data. In this case the website allows reflected input. Reflected XSS vulnerability may exists in the input fields of website forms, however, these input fields can have many layers of validation and filtration. So, in the existing available detection tools, not anything entered by the user reaches the server or the database.

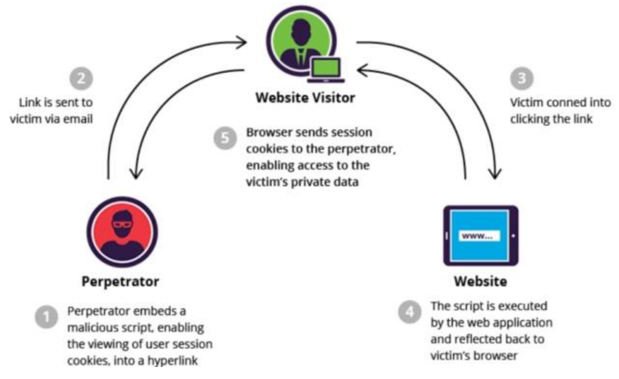

2 Link is sent to victim via email
Website Visitor
3 Victim conned into clicking the link
5 Browser sends session cookies to the perpetrator, enabling access to the victim's private data
www...
Perpetrator
Website
1 Perpetrator embeds a malicious script, enabling the viewing of user session cookies, into a hyperlink
4 The script is executed by the web application and reflected back to victim's browser

Fig. 1. Reflected XSS Attack

In most cases this filtration is done in both client-side and server-side to prevent any unauthorized code or script to reach the webserver and to be executed [3].

In the automatic detection of the reflected XSS, existing tools have to find every single input field in every directory in the entire webserver, and of course as in any automated program it takes a very long time to complete a full scan. The proposed methodology in this project alleviate all of these problems starting with bypassing of input validation, [6] this was achieved by using various encoding methods on the payload as shown in Table (1).

TABLE I
ENCODING TYPES AND METHODOLOGY

Encoding Type	Encoding Mechanism
HTML Entity Encoding	map symbol to another symbol , example: Convert ” to ’quote’
HTML Attribute Encoding	Except for alphanumeric characters, encode all characters with the HTML Entity format, including spaces.
URL Encoding	URL encoding should only be used to encode parameter values, not the entire URL or path fragments of a URL.

Furthermore, automatic detection has been achieved through the use of a web crawling mechanism to find all directories and then a page scanning mechanism to find all the input fields. In order to reduce the processing time in the project, we opted to rely on multithreading techniques in both crawling and scanning as well as during the vulnerability detection.

IV. PROPOSED SYSTEM

The proposed system has been built up of a number of modules that achieved the aforementioned goal. Figure (2) presented the flowchart of the first component in the system, i.e., the crawler [5] to help search through the entire webserver and find all associated web pages with the input URL.

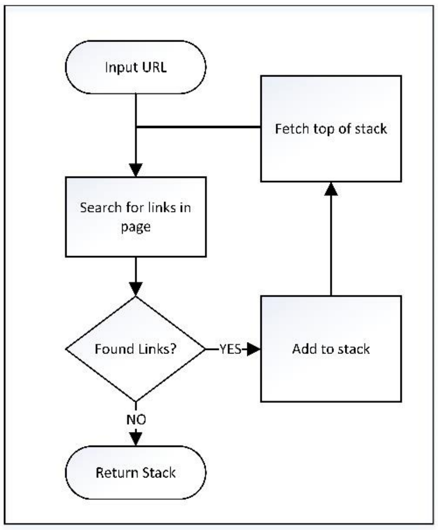


Fig. 2. Crawler Flowchart

The second component is the scanner that mainly relies on the output of the first component. As shown in figure (3). The scanner goes through all possible locations where the payloads can be inserted and tests them against vulnerability allowance, then the payload generator uses the encoding module to test whether an input requires encoding or not and apply the required encoding accordingly. Finally, the results of the scanner component are displayed to the user based on his preferences.

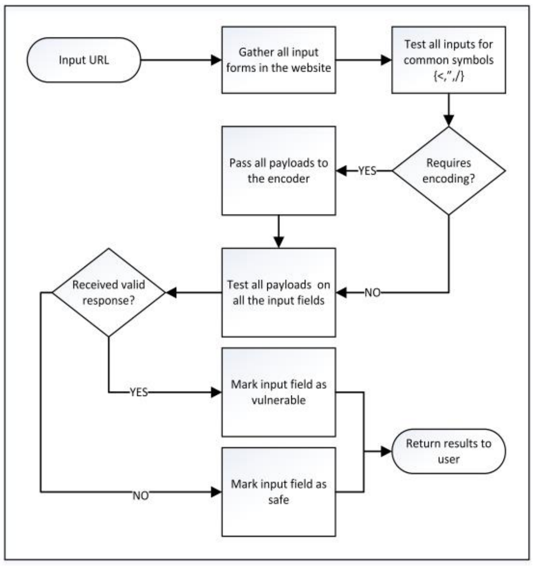
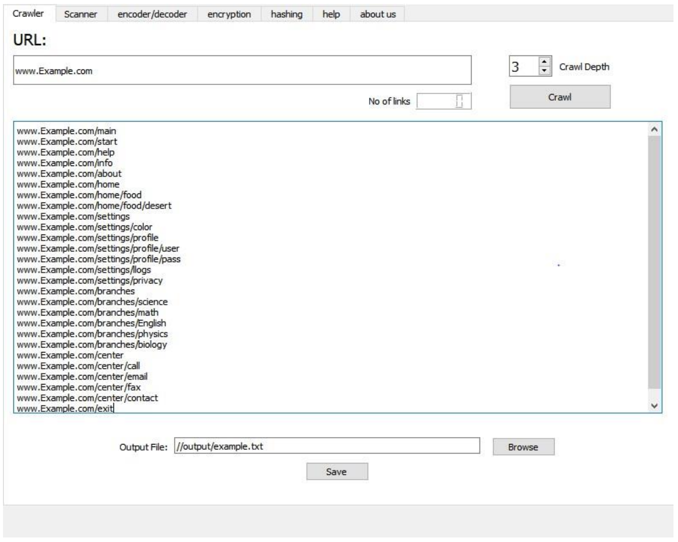

Input URL
Gather all input forms in the website
Test all inputs for common symbols {<,",/}
Requires encoding?
YES
Pass all payloads to the encoder
NO
Test all payloads on all the input fields
Received valid response?
YES
Mark input field as vulnerable
NO
Mark input field as safe
Return results to user

Fig. 3. XSS Scanner Flowchart

V. IMPLEMENTATION

The proposed system was implemented using python programming language [2], it can be used on any operating system. Figure (4) shows the main graphical user interface for the program where the user can start using the tool by entering the intended website URL that need to be in the crawler. The crawler interface gives options to the user like choosing the crawling depth.



URL: www.Example.com

Crawl Depth: 3

No of links:

```
www.Example.com/main
www.Example.com/start
www.Example.com/help
www.Example.com/info
www.Example.com/about
www.Example.com/home
www.Example.com/home/food
www.Example.com/home/food/desert
www.Example.com/settings
www.Example.com/settings/color
www.Example.com/settings/profile
www.Example.com/settings/profile/user
www.Example.com/settings/profile/pass
www.Example.com/settings/llogs
www.Example.com/settings/privacy
www.Example.com/branches
www.Example.com/branches/science
www.Example.com/branches/math
www.Example.com/branches/English
www.Example.com/branches/physics
www.Example.com/branches/biology
www.Example.com/center
www.Example.com/center/call
www.Example.com/center/email
www.Example.com/center/fax
www.Example.com/center/contact
www.Example.com/exit
```

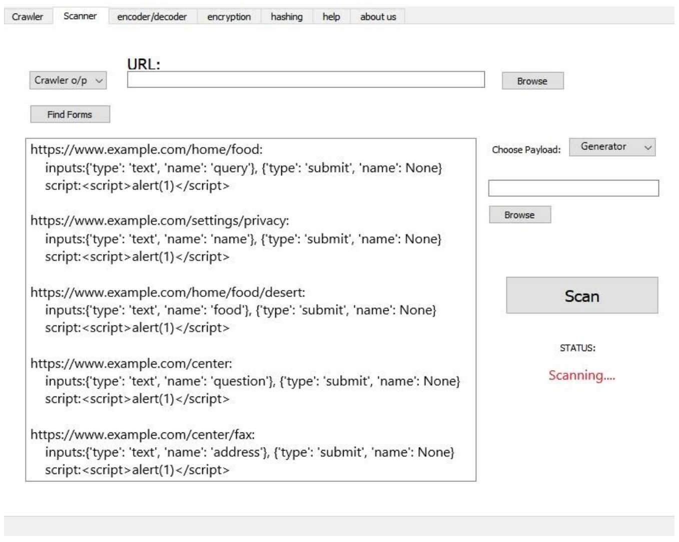
Output File: //output/example.txt

Fig. 4. Crawler User Interface

When the crawler finished its operation, it then passes the crawling result list to the scanner interface where the user can use many given options, e.g., the ability to use the included payload generator or input his own payloads, the ability to use encoding or not.

As shown in figure (5), the user can even choose which website URL to test and which one to not test. Once the user press on Scan button, the tool starts it operation and tests every input field in every URL using all the payloads and encoding for reflected XSS, once the scanning is completed the tool displayed the results a list of vulnerable input fields in the website each accompanied by the accepted payload and type of encoding used.

This results alerts the user for any security backdoors in the website even from the defense or offense point of view.



```
https://www.example.com/home/food:
   inputs:{'type': 'text', 'name': 'query'}, {'type': 'submit', 'name': None}
   script:<script>alert(1)</script>

https://www.example.com/settings/privacy:
   inputs:{'type': 'text', 'name': 'name'}, {'type': 'submit', 'name': None}
   script:<script>alert(1)</script>

https://www.example.com/home/food/desert:
   inputs:{'type': 'text', 'name': 'food'}, {'type': 'submit', 'name': None}
   script:<script>alert(1)</script>

https://www.example.com/center:
   inputs:{'type': 'text', 'name': 'question'}, {'type': 'submit', 'name': None}
   script:<script>alert(1)</script>

https://www.example.com/center/fax:
   inputs:{'type': 'text', 'name': 'address'}, {'type': 'submit', 'name': None}
   script:<script>alert(1)</script>
```

Fig. 5. XSS Scanner

VI. CONCLUSION

The project aimed to achieve main objective in developing an automatic tool that detect reflected XSS vulnerability, one of OWASP top-ten listed vulnerabilities. The scanning tool was developed using python. In traditional tools this process is performed manually, but now it has been Automated starting from crawling websites with any depth, scanning specific URL, and finally testing the endpoint input fields vulnerability using generated payload and encoding system. This tool allowed the scanning and detection of reflected XSS vulnerability. the entire project is developed using Python programming language and can be used on any system with multithreaded capabilities for maximum performance.

REFERENCES

[1] Azshwanth, D., and G. Sujatha. ”A novel automated method to detect XSS vulnerability in webpages.” 2022 International Conference on Computer Communication and Informatics (ICCCI). IEEE, 2022.

[2] Abirami, R., et al. ”Detecting Security Vulnerabilities in Website using Python.” 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC). IEEE, 2020.

[3] Singh, Mehul, Prabhishek Singh, and Pramod Kumar. ”An Analytical Study on Cross-Site Scripting.” 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA). IEEE, 2020.

[4] Pandya, Deven, and N. J. Patel. ”OWASP top 10 vulnerability analyses in government websites.” International Journal of Enterprise Computing and Business Systems 6.1 (2016).

[5] Udapure, Trupti V., Ravindra D. Kale, and Rajesh C. Dharmik. ”Study of web crawler and its different types.” IOSR Journal of Computer Engineering 16.1 (2014): 01-05.

[6] Wassermann, Gary, and Zhendong Su. ”Static detection of cross-site scripting vulnerabilities.” 2008 ACM/IEEE 30th International Conference on Software Engineering. IEEE, 2008.