International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 10, ISSN No. 2455-2143, Pages 55-58
Published Online February 2020 in IJEAST (http://www.ijeast.com)

DETECTION OF WEB APPLICATION VULNERABILITIES USING VATSCAN SCANNER

S. Jayamoorthy, AP
Department of CSE
Manakula Vinayagar
Institute of Technology,
Puducherry, India

C. Thirumalaivasan
Department of CSE
Manakula Vinayagar
Institute of Technology,
Puducherry, India

P. Yogeshwar,
Department of CSE
Manakula Vinayagar
Institute of Technology,
Puducherry, India

S. Sainath
Department of CSE
Manakula Vinayagar
Institute of Technology,
Puducherry, India

Abstract — **Information security is the heart of the computer's current renaissance our project deals with detecting Web Application attacks under the standards of OWASP (Open Web Application Security Project) which is an NGO (Non-Governmental Organization) for universal Web Application Security. The scanner focuses on security threats like XSS (Cross Site Scripting) and (LFI) Local File Inclusion Vulnerability which occurs in Interactive Web Applications. The scanner uses crawling Operation to detect the web application end points, vulnerable input parameters and evaluating whether the input parameter is Vulnerable to exploit or not by executing harmless malicious scripts. If the parameter is vulnerable to Exploit then the parameter is considered as a threat vector and the vulnerability name is displayed. The vulnerabilities are classified according to its severity as low, medium, high by OWASP standards. The scanner is built using Python to collaborate with the web modules and documentation is published for future references. By using this VATSCAN testing strategy the vulnerability scanner detects the vulnerabilities and categorizes them based on its severity, and the documentation is used to prevent from future Threat vectors and Attack vectors.**

Keywords—**XSS, Scripting, SQL, Vulnerabilities, Threat, LFI, OWASP, XXE**

I. INTRODUCTION

[1]Web technology has been increased exponentially in daily volume and interactions involving web-based services, such as self-driving finance in web banking, Chatbot's /AI based assistants and web based recommendation engines in e-commerce, social networking sites and much more. Web Applications has become an integral part of our daily lives, and at the same pace web applications became the primary targets of Attackers. [2]Attackers can exploit the poor code experiences of web developers, vulnerabilities within the code, improper user input sanitization and the non-compliance with the proper security standards by the software package developers. One of the common high risk cyber-attack to web application vulnerabilities is Cross-Site Scripting (XSS), which has placed web applications, users, and even the industrial field at high risk. [3]According to the National Vulnerabilities Database (NVD) database, the number of reported issues are increasing exponentially, especially in 2019. XSS-based vulnerabilities are considered to be the second most common vulnerability occurring and still it is one of the top 10 attacks of OWASP in 2017.

II. PROPOSED WORK

A. Existing Scanner–

[4]Nowadays some of the automated scanners may cause damage to the web application during the testing process and also damages the master source code of the web-application. As we all know that the every web based vulnerability scanners will have a parameter to get the target as an input which itself a vulnerable to the scripting attacks like XSS (Stored and persistence). While performing the automated security scanning, the scanner generates huge number of request to the targeted web application which breaks the connection bond between the web application and the server. Some sensitive parameters must undergo manual testing instead of automated testing because any damage caused to the sensitive part is not recoverable.

B. Proposed Scanner–

The scanner focuses on preventing web application attacks based on the *Keary Eoin*. et.al. (2013) *OWASP testing guide* [5] OWASP (Open web application security Project) standards An NGO for web application security standards. The scanner uses technology detecting mechanism like Wapplyzer (Web application protocol analyzer), an open-source program to detect the technology like HTML, JS (Java Script), ORACLE DATABASE and carry out the testing process based on the technologies that are being used in the web application or website unlike other vulnerability scanners which carry out all the testing processes blindly whereas VATSCAN works on the principle of "Identifying the technology first before testing" this saves the lot of execution time, for example the SQL scanning module is skipped for the web application which uses Mongo DB as a Database and eliminating dead code lines also improves greater value to the function of the program, after the technologies are being identified, the testing process is framed.

Since the URL(Uniform Resource Locator) is the input for the tester, the URL is loaded into the VATSCAN to carry out testing as an initial step, the input parameters are analyzed in the web application and ordered in the format to test the payloads in one by one, the payloads are non-malicious codes collected from open source web application security community such as OWASP community etc., these payloads are organized according to its performance from basic to complex and formatted one by one in a payload list such that the scanner takes the payloads from the lists and perform testing based on the [6] "Test one after another" methodology the harmless payload's are passed into the parameter to find out whether the parameter is vulnerable or not, if the payload is executed successfully causing the predetermined way, which if the harmless payload execution at a parameter is successful then it is considered as the vulnerability based on the characteristic of the payload execution. If a non-malicious harmless code is successfully executed at the parameter then there is all the possibilities to execute a malicious code which allows an attacker to cause potential harm to the entire web application.

[7] The testing mechanism is carried out using python (i.e.) a Marshall Joseph. et.al. (2018) *Hands-On Bug Hunting for Penetration Testers - A Practical Guide to Help Ethical Hackers Discover Web Application Security Flaws* [8] XSS (Cross site Scripting attack) vulnerability can lead to cookie stealing and malicious URL redirection based on the attacker commands and SQLi (SQL Injection) allows to dump the entire database and also read sensitive information in the database without the knowledge of the database administrator.

SSRF (Server Side Requests Forgery) can send illegal request between servers without the knowledge of the Server administrator and CSRF (Client Side Request Forgery) can send Forged requests between one client and another client without the Knowledge of the concern user.

FILE INCLUSION Vulnerability allows an attacker to include a file is classified into two categories.

1) *LFI(Local File Inclusion)*
2) *RFI(Remote File Inclusion)*

(1) LFI (Local File Inclusion) allows an attacker to include a file by exploiting a "dynamic file inclusion" which is carried out in the Web Application.

(2) RFI (Remote File Inclusion) involves in including a remote file via exploiting the vulnerable inclusion procedures implemented in the web application.

(XXE) XML external entity injection is a web application security vulnerability that allows an intruder to interfere with a application processing of XML data and it also permits the intruder to view the files on the web application server file system under certain scenarios, an intruder can escalate XXE to compromise the underlying server or the backend infrastructure, an XXE attack can lead to SSRF attack. SPF (Sender Policy Record) missing Vulnerability allow anyone to send mail from the trusted organization email address, which leads to identity theft. *Mitropoulos Dimitris, Louridas Panos, Polychronakis Michalis, and Keromytis D Angelos*. et.al. *Defending Against Web Application Attacks - Approaches, Challenges and Implications* [9] RCE (Remote Code Execution) which allows an attacker to execute a OS command in the host operating system of the web application (i.e.) "ls" command list all the files in the current directory and "pwd" command print the current working directory of the file hosted. HTML injection allows an attacker to insert arbitrary HTML codes into the web application which may lead to XSS by making a chained attack therefore one attack can lead to another attack and so on. Improper client side validation is the major cause for all the above mentioned attacks.

K.Pranathi, S.Kranthi, Dr.A.Srisaila, P.Madhavilatha et.al. (2018*). Attacks on Web Application Caused by Cross Site Scripting* [10] to avoid this kind of scenarios in the organizations should implement proper (WAF) Web Application Firewall with security filters and carryout routine security checkup by investing more money in *Vest Joe and Tubberville James. et.al. (2019) Red Team Development And Operation* [11] red team (External security team performs security testing) and blue team activities (internal security team performs security testing), after the vulnerable parameter is found based on the payload then the vulnerability is named based on the payload type and the vulnerability is classified based on its severity, the severity is classified based on the OWASP standard, in the category of severe, critical, low after the vulnerabilities are identified the documentation/result is generated through the results the tester can get a clear understanding about the target which he/she encountered. This vulnerability scanner is useful for those who work in web application security where [12] automation plays a vital role in reducing the time which leads to focus on other grey areas of security since security is not a onetime process, "The pretty good security" involves periodic checkup on the target and perform the activities which really matters and perform risk assessment with the tools to ensure the security is up to the level and prevent from future data breaches and make a scan result as documentation for future references and therefore the tester can avoid the data leaks before it occurs. *Stamp Mark. et.al.* (2011) *Information Security Principles and Practice.* [13] As we know security breach is not a reversible process once a data breach is happened, the leaked data roams over the internet forever therefore investment in the employee awareness is the most necessary thing every organization must follow in the cyber security than investing in assessment activities which leads to greater improvement of the organization.

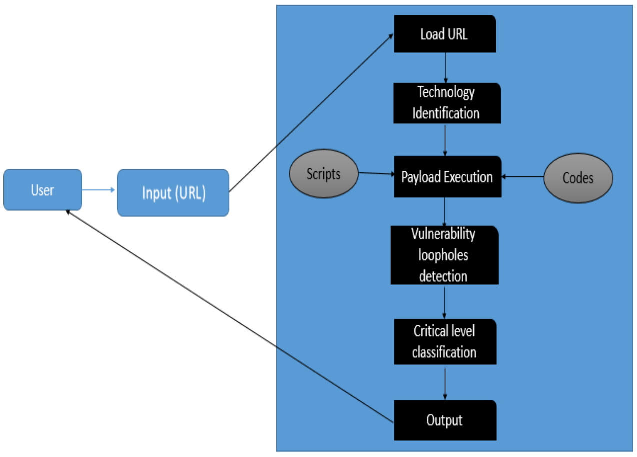
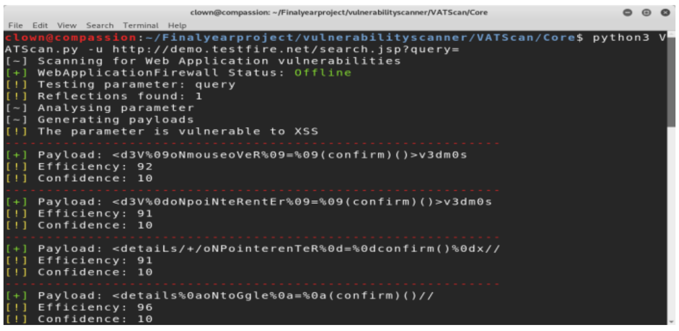

User
Input (URL)
Load URL
Technology Identification
Scripts
Payload Execution
Codes
Vulnerability loopholes detection
Critical level classification
Output

Fig. 1. VATSCAN Scanner Block Diagram

III. EXPERIMENT AND RESULT

According to the OWASP standards we developed a python script to check vulnerabilities in the websites by passing the URL of the website as an argument and execute the python script and finally we have made GUI (Graphical User Interface) which was developed using one of the Framework called python "Tkinters" and now the user can just enter the URL to be tested and click on the button which executes the python scripts which provides the results. Our experimental output can provide a status of web-application vulnerability such as type of vulnerability detected and number of vulnerability detected. It can also analyze the parameters, generates the payloads to be executed and it provides the status of efficiency and confidence of the payload.



```
clown@compassion: ~/Finalyearproject/vulnerabilityscanner/VATScan/Core
File  Edit  View  Search  Terminal  Help
clown@compassion:~/Finalyearproject/vulnerabilityscanner/VATScan/Core$ python3 V
ATScan.py -u http://demo.testfire.net/search.jsp?query=
[~] Scanning for Web Application vulnerabilities
[+] WebApplicationFirewall Status: Offline
[!] Testing parameter: query
[!] Reflections found: 1
[~] Analysing parameter
[~] Generating payloads
[!] The parameter is vulnerable to XSS
------------------------------------------------------------
[+] Payload: <d3V%09oNmouseoVeR%09=%09(confirm)()>v3dm0s
[!] Efficiency: 92
[!] Confidence: 10
------------------------------------------------------------
[+] Payload: <d3V%0doNpoiNteRentEr%09=%09(confirm)()>v3dm0s
[!] Efficiency: 91
[!] Confidence: 10
------------------------------------------------------------
[+] Payload: <detaiLs/+/oNPointerenTeR%0d=%0dconfirm()%0dx//
[!] Efficiency: 91
[!] Confidence: 10
------------------------------------------------------------
[+] Payload: <details%0aoNtoGgle%0a=%0a(confirm)()//
[!] Efficiency: 96
[!] Confidence: 10
```

Fig.2. Reflected XSS Found with Payloads, Efficiency and Confidence.

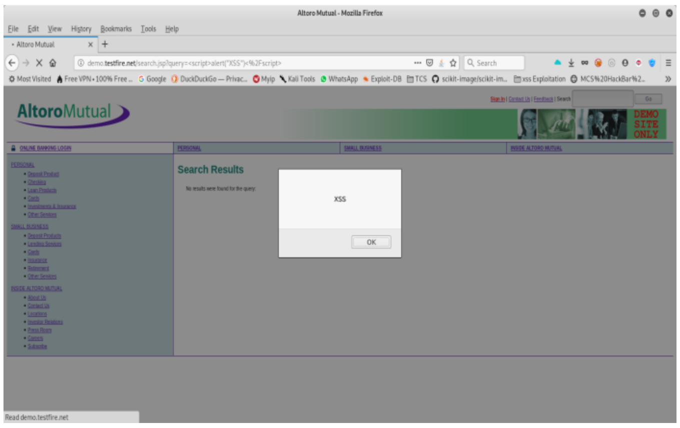

Altoro Mutual - Mozilla Firefox
File Edit View History Bookmarks Tools Help
AltoroMutual
Search Results
XSS
OK

Fig. 3. Reflective XSS on AltoroMutal Lab

IV. CONCLUSION

VATSCAN can be used to detect critical vulnerabilities such as LFI (Local File Inclusion) which leads to sensitive information leak and also security scanning is not a onetime process it is a routine process carried out at interval of time any new feature added to the application the entire web application need to be tested to ensure the security of the entire web application. The Scanner is developed in Python which make it as platform independent and compatible with all the operating systems, a report is finally generated in the desirable format for future references and it is easy for the developers and penetration testers to perform security testing on the targeted web application.

V. FUTURE WORKS

False positive reports are eliminated to fine tune the efficiency of the Vulnerability Scanner. Since the scanner is developed under sustainable development methodology so any future changes can be deployed easily. New vulnerabilities are going to bloom in the future it is necessary to update the scanner engine to test the up to date vulnerabilities in the efficient way.

VI. ACKNOWLEDGEMENT

We would like to thank Cyber Security Researchers, Professors all over the world. This work is supported by Manakula Vinayagar Institute of Technology, Puducherry.

VII. REFERENCE

[1] Zalewski Michał. (2012). The Tangled Web - A Guide to Securing Modern Web Applications, (pp. 1-17).
[2] Erickson Jon. (2008). Hacking the Art of Exploitation, (pp. 118-142).
[3] Husak Martin, Komarkova Jana, Bou-Harb Elias, and eleda C Pavel. (2018). Survey of Attack Projection, Prediction, and Forecasting in Cyber Security, COMST.2018.2871866, IEEE. (pp. 13-14).

[4] https://www.acunetix.com/blog/articles/negative-impacts-automated-vulnerability-scanners-prevent/.

[5] Keary Eoin. (2013). OWASP testing guide, (pp. 5-14).

[6] Watson Colin and Zaw Tin. (2018). OWASP Automated Threat Handbook Web Applications. (pp. 54-57).

[7] Christian Martorella, Buchanan Cameron, lp Terry, Mabbitt Andrew, May Benjamin, Mound Dave. (2015). Python Web Penetration Testing Cookbook. (pp. 40-170).

[8] Marshall Joseph. (2018). Hands-On Bug Hunting for Penetration Testers - A Practical Guide to Help Ethical Hackers Discover Web Application Security Flaws. (pp. 55-72).

[9] Mitropoulos Dimitris, Louridas Panos, Polychronakis Michalis, and Keromytis D Angelos. (2017). Defending Against Web Application Attacks - Approaches, Challenges and Implications, TDSC.2017.2665620, IEEE. (pp. 2-3).

[10] K.Pranathi, S.Kranthi, Dr.A.Srisaila, P.Madhavilatha. (2018). Attacks on Web Application Caused by Cross Site Scripting, ICECA, IEEE Conference Record # 42487; IEEE Xplore ISBN:978-1-5386-0965-1. (pp. 1754-1755).

[11] Vest Joe and Tubberville James. (2019). Red Team Development And Operation. (pp. 77-101).

[12] Rahul Maini, Rahul Pandey. Kumar Rajeev, Gupta Rajat. (2019). Automated Web Vulnerability Scanner. International Journal of Engineering Applied Sciences and Technology, 2019 Vol. 4, Issue 1, ISSN No. 2455-2143, (pp. 132-136).

[13] Stamp Mark. (2011). Information Security Principles and Practice. (pp. 13-14).