IJARIIT INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 4 - V7I4-1208)
Available online at: https://www.ijariit.com

Web application security – Automating the manual exploitation methods and eliminating false positives

Jayshish M. Popat
jayshish00@gmail.com
Atria Institute of Technology, Bengaluru, Karnataka

Mohammad Ashar Nawab
ashar.nawab123@gmail.com
Atria Institute of Technology, Bengaluru, Karnataka

Asuri Ritesh Kumar
asuriritesh1413@gmail.com
Atria Institute of Technology, Bengaluru, Karnataka

Nishant Prakash
nishantprakash1336@gmail.com
Atria Institute of Technology, Bengaluru, Karnataka

Manjula M.
manjula.m@atria.edu
Atria Institute of Technology, Bengaluru, Karnataka

ABSTRACT

Many web application security scanners are prone to false alarms indicating that your website is vulnerable when it isn't. False positives are a major problem in web application security, as they make security testing slower, less accurate and more frustrating. Proof-Based Web Vulnerability Scanning Technology in DevSecOps/SecDevOps environment will let you eliminate security vulnerabilities as early as possible, helping you save a lot of resources. It would automatically exploit identified web security vulnerabilities and also produce a proof of exploits which confirms that the identified vulnerabilities are genuine and not false positives. In this paper we present a technique to find web vulnerabilities using our proposed algorithm and provide extracted sample data as proofs. Our tool identifies vulnerabilities with the same level of certainty as a penetration tester or bounty hunter. This will assist developers and security teams in fixing vulnerabilities in less possible times.

Keywords— *Vulnerabilities, exploits, proofs, false positives, resources.*

1. INTRODUCTION

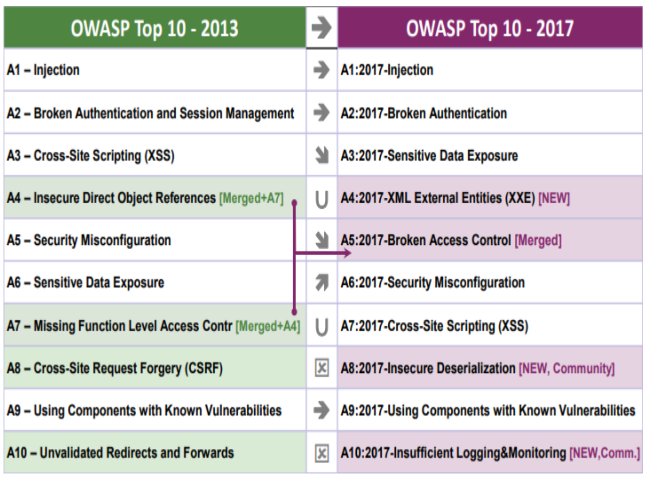
With the increase of the number of websites each and every day, the security aspect of the websites has become a more concerning factor. Since most of the software developers are not aware of various security measures to be introduced into the system as their motive is just to make the software application run in a desired state without taking into consideration the flaws that the programming language might have introduced into the system; to protect the website and end users from the risk of being attacked by any unauthorised access, it becomes significantly more important to devise new strategies and methodologies that will analyse the security breaches to which the website is vulnerable to.

Not only does the web application developed with flaws make the website vulnerable to attacks, but also end users become a key factor by compromising the security aspect. Assessing and eliminating the vulnerabilities requires the knowledge and deep understanding of these vulnerabilities. It becomes necessary enough to know the basic idea that works behind these vulnerabilities such as what makes them to appear in the system.

1.1 Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entity
5. Broken Access Control
6. Security Misconfiguration
7. Cross-site Scripting
8. Insecure Deserialization
9. Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

One can find, obviously, many different ways to compromise and breach applications which go beyond the OWASP Top Ten, however the list is a fairly nice beginning. Security exploits are constantly changing and emerging as time passes, there have been a number of changes towards the listing from OWASP Top Ten 2013 to 2017 as shown in the Figure 1. This paper concentrates on our automated tool based on the OWASP Top Ten2017.

The OWASP top 10 attacks of 2017 are:



OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Figure 1: Owasp top ten 2017

Various vulnerability scanners have been deployed to identify these Top Ten vulnerabilities and automate the task to do the work as fast as possible. Vulnerability Scanner which claims accuracy, low false positives and automation, usage of these vulnerability scanners is not so effective as it generates false positives and security experts have to manually exploit the attack what flaws need to be corrected to make the system free from these vulnerabilities, which can be achieved by the OWASP top 10, which gives a complete guide to how the attackers attack and how to defend it and many more.

which again takes time and resources. However, the use of vulnerability scanners only to detect the vulnerabilities is considerable but true automation is not possible. In this Paper, we have designed an algorithm and developed a way to detect the vulnerability and exploit it using Python modules, which we believe is complete automation. As the output of the scanner will be the final output and no extra time is wasted to exploit it manually. They play a significant role in detection of attacks and speeding the process. The implementation and the workflow is explained in the Methodology section.

2. METHODOLOGY

Proof Based Vulnerability scanning means identifying the vulnerabilities in the system and exploits identified vulnerabilities in a read-only and safe way, in order to confirm identified issues. This Paper discusses algorithms and workflow of our tool that how the website is crawled, tested and exploited. Our Tool is the free and open-source web application vulnerability scanner that automatically exploits vulnerabilities and also presents proof of the vulnerability so that you do not need to waste time manually verifying it. For example, in the case of a detected SQL injection vulnerability, it will show the database name as the proof of the exploit.

2.1 Overview

In our proposed method, a set of libraries and algorithms are used for identification of vulnerability, crawling and confirming with relevant output within the scanning process. These tasks were performed systematically with the help of the proposed framework. Figure 2 shows the flowchart of how the identification and confirmation of vulnerabilities of the target system is performed using automated scripts.

The above steps define the core working of our tool in which the power of true automation is achieved and algorithms are utilized for saving time and resources. Our proposed method helps to secure web applications easily without any fuss, so one can focus on fixing the reported vulnerabilities. If any proof is automatically confirmed, it will show a prefix [‘Vulnerability Confirmed’] in the output, so that one knows what should be fixed immediately as shown in figure 3 and 4.

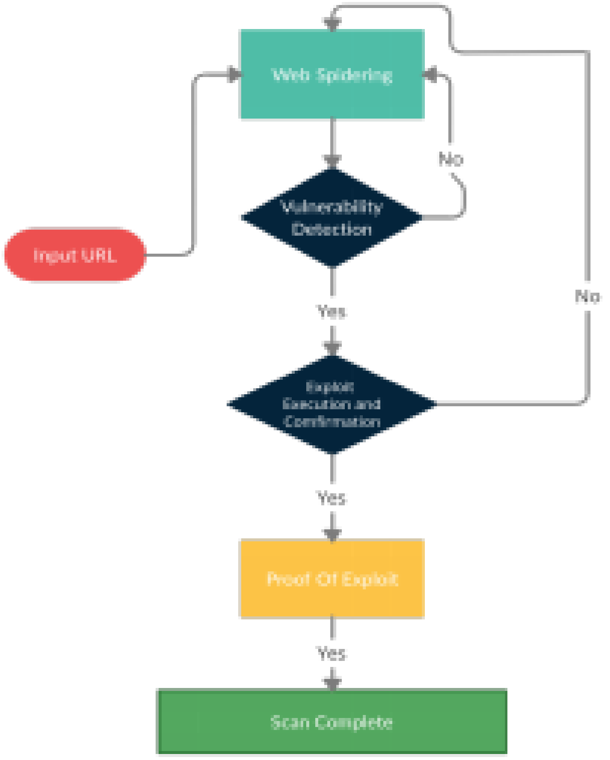
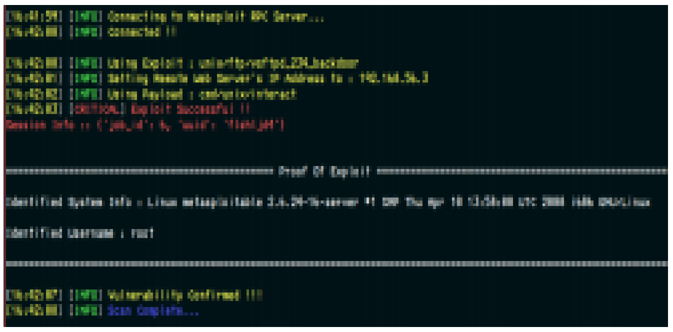


Figure 2 : Tool Flowchart

Our scanning methodology uses Python3 Metasploit automation library (pymetasploit3) to achieve automation and beautifulsoup to scrape information from web pages such as web forms, links as shown in the figure 3.



```
[16:41:59] [INFO] Connecting to Metasploit RPC Server...
[16:42:00] [INFO] Connected !!

[16:42:00] [INFO] Using Exploit : unix/ftp/vsftpd_234_backdoor
[16:42:01] [INFO] Setting Remote Web Server's IP Address to : 192.168.56.3
[16:42:02] [INFO] Using Payload : cmd/unix/interact
[16:42:03] [CRITICAL] Exploit Successful !!
Session Info :: {'job_id': 0, 'uuid': '[illegible]'}


================================ Proof Of Exploit ================================

Identified System Info : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux

Identified Username : root

==================================================================================

[16:42:07] [INFO] Vulnerability Confirmed !!!
[16:42:08] [INFO] Scan Complete...
```

Figure 3 : Automated Command Injection

2.2 Workflow

The proposed framework can easily be divided into 6 steps on the bases of its working and outputs that are used in the next phase. The first step takes a URL as an input from the user. The second step uses the output from the previous step that is URL and starts crawling the website in loop and each website is passed to the next step i.e., third step for vulnerability Detection. The third step uses detection techniques to find the vulnerability, here if the vulnerability is not found then control is given back to the crawler else it moves to fourth step i.e., Exploit execution. The fourth step runs the exploit on the detected URL and if the exploit runs then proof of exploit is generated, that is step five else control is given back to the crawler again to generate a new URL as shown in the Figure 1.These steps will continue until all the URLs in the web application are crawled and tested, Finally the Scan completes in the last step with the output on the screen.

3. MODEL DESIGN

Our Proposed model consists of two Operating Systems connected with each other with a NAT network in Oracle VirtualBox. Attacker machine is Kali Linux and the vulnerable Web application is Mutillidae. Our Proof based Scanner is in Kali Linux to scan the Web application. As our tool automates all the process, users just have to enter the target URL to start scanning. So once all the URLs are exploited and proof is gathered it is displayed on the screen which is nothing but the vulnerability report as shown in the figure 4.

3.1 Web Vulnerability Scanner

The proposed algorithm that is used to make the tool is given below :

```
Algorithm1:
start
-->Connect Metasploit RPC Server
-->Load the exploit
-->Set RHOSTS to victims IP
-->Set suitable payload
-->Execute the payload and get session
-->Grap Proof of exploit from the session such as System Information and Username
End
Algorithm 2:
start
DEFINE FUNCTION run_scanner(self):
    FOR link IN self.target_links:
        SET forms TO self.extract_forms(link)
        FOR form IN forms:
            → Test XSS IN the form
            IF is_vulnerable_to_xss:
            → Discovered XSS IN the following form of the URL
            → Run Exploit IN the target form
            → Test exploit_in_form
            IF is_exploitable:
→ Vulnerability Confirmed !
            ELSE:
                → false positive
        ELSE:
            → XSS not discovered. Form is not vulnerable !!
        Return
End
```

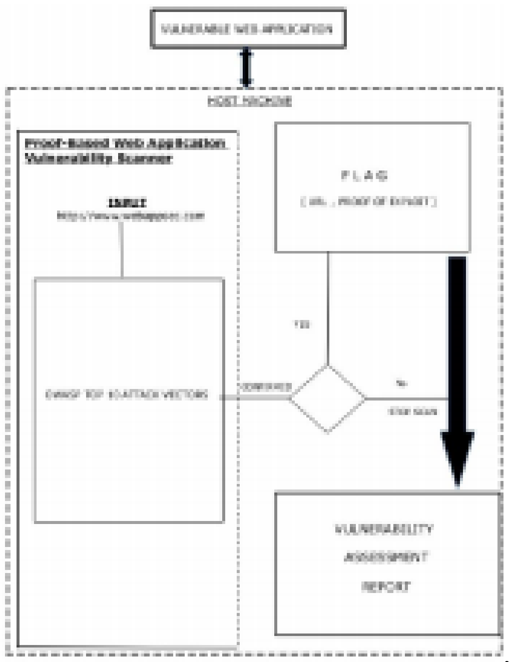
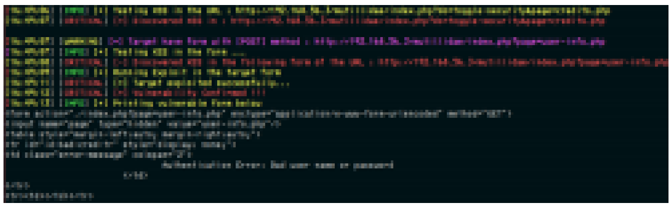

VULNERABLE WEB APPLICATION
HOST MACHINE
Proof-Based Web Application Vulnerability Scanner
INPUT
http://www.webappsec.com
FLAG
(URL , PROOF OF EXPLOIT)
YES
OWASP TOP 10 ATTACK VECTORS
CONFIRMED
NO
STOP SCAN
VULNERABILITY ASSESSMENT REPORT

Figure 4: Proposed Model

4. RESULTS AND DISCUSSION

The Testing was done on a web application (Mutillidae) of a vulnerable machine (Metasploitable2) with host machine as Parrot Security OS. Figure 5 shows the PoC (Proof Of Concept) for our proposed algorithm tested for Cross-Site Scripting attacks. Table 1 depicts the total vulnerabilities detected, number of false-positives, and number of vulnerabilities successfully exploited.



```
[illegible]
                Authentication Error: Bad user name or password
[illegible]
```

Figure 5 : Automating Cross Site Scripting

Table 1. Comparison table for tested Web attacks outputs

Attacks	Vulnerabilities Detected	False Positives	Successfully Exploited
Command Injection	23	10	13
Cross-Site Scripting	17	12	5
SQL Injection	12	5	7
Broken Authentication	2	0	2

5. CONCLUSION

The Objective of this project is to study the foundations of web application security and to learn how to protect these assets by securing the applications to create hack resilient code and provide a Automated Proof based tool to be preventive and rely on a secure platform, to improve security in the organization. This ensures that the information is secured according to the standards we need, to apply these measurements and follow the approaches that best suit our security needs.

Each identified risk in the project is prioritised according to prevalence, detectability, impact and exploitability. Committing to ensure your applications consider each of the top ten risks serves as an ideal starting point for focusing on application security. Our Project includes the theoretical background, definitions and security fundamental information including models, standards and information about security. The Project consists of a testing methodology conducted at the application. Whenever the vulnerability scanner matches and identifies the known vulnerabilities of the web application in its database, it will raise a flag with "CONFIRMED" status along with proof of exploits.

5.1 Vulnerability Assessment Report

This report will contain accurate vulnerabilities along with URLS and proof of exploit to plan and prioritize your resolution efforts for maximum effectiveness level, using penetration testing tools. The analysis explains the most known security threats based on the results obtained for the IT Company. In our Project, Vulnerability is confirmed and proven, the issue is definitely real – as it has already safely exploited it. Bug along with proof and Combined with severity ratings and technical information provided in each vulnerability report. An automated tool can identify vulnerabilities with the same level of certainty as a penetration tester or bounty hunter. It means that the issue is real and you can move to address it without any additional checking by the security team. If you see “Confirmed”, you have a vulnerability that you need to fix.

6. REFERENCES

[1] Doshi, Jignesh & Trivedi, Bhushan. (2017). A Novice Approach for Web Application Security.10.1007/978-981-10-2750-5_1.

[2] Sedek, Khairul Anwar & Norlis, Osman & Osman, Mohd & Jusoff, Kamaruzaman. (2009). Developing a Secure Web Application Using OWASP Guidelines. Computer and Information

[3] 8. Bozic, Josip & Wotawa, Franz. (2020). Planning-based security testing of web applications with attack grammar. Software Quality Journal. 10.1007/s11219-019-09469-y. [4] https://github.com/waylan/beautifulsoup

[5] H.Huang, Z.Zhang, H.Cheng and S.Shieh, "Web Application Security: Threats, Countermeasures, and Pitfalls" in Computer, vol. 50, no. 06, pp. 81-85, 2017. doi: 10.1109/MC.2017.183 url: https://doi.ieeecomputersociety.org/10.1109/MC.2017.183

[6] Anis, Arafa & Zulkernine, Mohammad & Iqbal, Md Shahrear & Liem, Clifford & Chambers, Catherine (2018).Securing Web Applications with Secure Coding Practices and Integrity Verification 618-625.10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.0 0 112.

[7] Perera, Ashan & Kesavan, Krishnadeva & Bannakkotuwa, Sripa & Liyanapathirana, Chethana & Rupasinghe,Prabath. (2016). E-commerce (WEB) Application Security: Defense against Reconnaissance. 732-742.10.1109/CIT.2016.105.

[8] https://github.com/DanMcInerney/pymetasploit3