IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.10534

A Study on Strengths and Limitations of Commercial Automated Dynamic Security Testing Tools for Web Applications

Melbin K Mathew

Independent Researcher, Kerala, India

Abstract- With the tremendous growth of the Internet in recent years, web applications are becoming more and more popular among users. At the same time, they suffer from malicious attacks that affect basic attributes Such as confidentiality, integrity, and availability of the system or the data. In response to these malicious threats, web application developers and information security professionals have used web Application vulnerability scanners as a security tool to Periodically audit their web applications to scan for security vulnerabilities. Today, there are a lot of automated web application scan tools available. But we need to find that is how efficient the scanners are in finding security vulnerabilities in web applications during an automated scan. The main focus of this study is to assess the effectiveness of existing commercial Automated Dynamic Application Security Testing (DAST) Tools in Black-Box web application security Testing. Our goal is to find out the strength and limitations of automated web application security testing tools. This study is not meant to compare different commercial scanners in the market. We do not report comparison data regarding the performance of the tools or make any recommendations about the purchase of any scanning tools.

Keywords- Web Application Security, Vulnerability Analysis, Penetration Testing, Security Scanners

INTRODUCTION

In the last few years, the use of the Internet has been increased tremendously.[1] Web applications are being used in almost all industries including healthcare, finance, social media, etc. As web applications have such a large user base, a large amount of data is stored, processed, and transferred using such applications. Attackers are particularly interested in this large amount of data. As a result, they exploit the security vulnerabilities in the applications to steal the user data from the application.[2]

WEB APPLICATION

A web application is a computer program that can be accessed with a web browser through a network such as internet or intranet. A web page contains HTML tags, CSS, and JavaScript code. These components are sent from the server to the user's browser, and it is executed in the user's browser.[3]
The client makes HTTP requests through the Internet to the webserver. HTTP is a protocol that is the foundation of the Internet, and it deals with the algorithms involved in the transfer of data to and from the web browser ad browser. The HTTP request from the client is sent to the server and the server processes the request and sends the response to the client according to the application logic. The web application in the server is written using server-side scripting languages such as PHP, Java, Python, etc. The data storage in web applications are usually done using databases. MySQL, PostgreSQL, and MariaDB are examples of database software.

CLASSIFICATION OF TESTING TYPES

In order to reduce the risk of data theft by attackers, the web application owners should audit the application to find the security vulnerabilities. The testing phase is a particularly important phase in the Software Development Life Cycle

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.10534

(SDLC) to prevent cyberattacks on web applications. There are three different types of tests in web applications namely Black box, Gray Box, and white box testing.[4] [5]

1. **Black Box Testing** - Black box testing is a type of testing in which the input or the requests are given to the application by the tester or testing tool and the output received from the application is checked, the internal working of the application is not considered here. It is a test done without any knowledge of the internal working of the application code.
2. **White Box Testing** - White box testing is a method of testing where the tester knows how the application works internally. This test involves testing an application with full access to source code and other documents.
3. **Grey Box Testing** - Grey box testing is a combination of White Box and Black box testing. This is a method of testing where the tester has partial knowledge about the structure and working mechanism of the application.

WEB APPLICATION VULNERABILITIES

Web Application vulnerabilities are the security issues in the application that may lead to affect the confidentiality, integrity, and availability of the internal data of the application if exploited by an attacker. The Open Web Application Security Project (OWASP) is a non-profit foundation and a community of engineers and security IT professionals. Their goal is to improve the security of Web Applications and other software. OWASP has published a list of critical security vulnerabilities in web applications. This list is called OWASP Top 10.[6] This list is developed based on a consensus among web application security experts from around the world. The issues are ranked based on the frequency of detection of the vulnerabilities, their severity, and their potential impacts if exploited by an attacker.

OWASP TOP 10 VULNERABILITIES

1. **Injection** - A code injection occurs when malicious input is sent by an attacker to the target web application. This input from the attacker is for making the application doing something which it is not intended to do. For example, SQL Injection is an Injection attack that exploits an insecure coding practice of not sanitizing user input before passing to the database, to extract sensitive data from the underlying database of the web application.
2. **Broken Authentication** - In some applications, authentication and session management may not be implemented securely. This allows attackers to compromise user sessions and passwords. This can also lead to leakage of sensitive data and account takeover. For example, if an application doesn't implement rate-limiting in the OTP submission for a password reset, this can be exploited by an attacker to take over a user account by resetting the user's password.
3. **Sensitive Data Exposure** - When a web application or an API transmits data using insecure methods, attackers can exploit these vulnerabilities to gain access to sensitive information including usernames, passwords, social security numbers, etc.
4. **XML External Entities** - When a web application is using a vulnerable library to parse XML files uploaded by the user, attackers can exploit this vulnerability to get access to sensitive data from the server where the web application is running.
5. **Broken Access Control** - Access control is a function that controls a user's access to a resource. If access controls are not implemented properly, attackers can get access to restricted systems and sensitive information.
6. **Security Misconfiguration** - Security Misconfiguration is one of the most common security vulnerabilities found in web applications. These are usually due to the failure to set the required security controls, use of default configurations in the applications, or negligence from the side of developers. For example, if a developer deploys a web application that is in debugging mode into production, it will display excessive verbose errors to the user when an error occurs in the application.
7. **Cross-Site Scripting** - Cross Site scripting or XSS vulnerabilities are the security loopholes in a web application when it serves untrusted scripts from the attacker to the browser. These untrusted scripts can be executed in the browser of the user and can be used to execute arbitrary malicious JavaScript code in the client browser.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.10534

8. **Insecure Deserialization** - Deserialization is the process of conversion of structured data into individual objects that can be fed into web applications for further processing. Insecure deserialization happens when deserializing data from untrusted sources and passing this data into the web application without further analysis. Insecure Deserialization can lead to RCE (Remote Code Execution) and DoS (Denial of Service) attacks.
9. **Using Components with Known Vulnerabilities** - When the third-party components included in the web applications are vulnerable, this affects the whole security posture of the application. These third-party components include APIs, Libraries, frameworks, etc. which helps the developers avoid redundant work and include functionalities into the application.
10. **Insufficient Logging and Monitoring** - Failure in logging and monitoring of a web application can lead to time delays in identifying attacks, and this leaves the web applications insecure. To prevent such issues, a developer should make sure that all sensitive events like login failures, user input validation failures, server error messages are logged so that malicious activity can be detected by examining the logs.

WEB APPLICATION VULNERABILITY SCANNERS

A Web Application Vulnerability Scanner is an automated tool that crawls the specified web application, analyzes the server responses for crafted requests, and finds out security vulnerabilities by matching the server responses with the ones in its database.[7] In this study, we analyze the performance of 3 commercial web application vulnerability scanners with Manual Web Application Penetration Testing to identify the strengths and limitations of Automated Web Vulnerability Scanning.

COMPARISON OF WEB VULNERABILITY SCANNERS WITH MANUAL TESTING

The aim of this evaluation is to find out the effectiveness of Dynamic Application Security Testing Tools with Manual Testing by an experienced Penetration Tester. A Web Application was tested using Three leading Commercial Web Application Security Scanners and the results were compared using a Penetration test report Prepared by a Certified Penetration Testing Professional. The tables below show the detection of security vulnerabilities using commercial security scanners and findings from a manual Penetration Test.

Scanner/Issue	XSS	SQL Injection	CSRF	Broken Authentication	IDOR	Security Misconfiguration
A	9	3	1	0	0	3
B	7	3	1	0	0	3
C	8	2	0	1	0	1
Manual PT	12	3	4	3	4	6

Scanner/Issue	Insecure Components	Privilege Escalation	Clickjacking	Sensitive Data Leakage	Business Logic Issues	Others
A	5	0	1	0	0	3
B	5	0	1	0	0	5
C	3	0	1	0	0	4
Manual PT	6	3	1	4	4	7

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

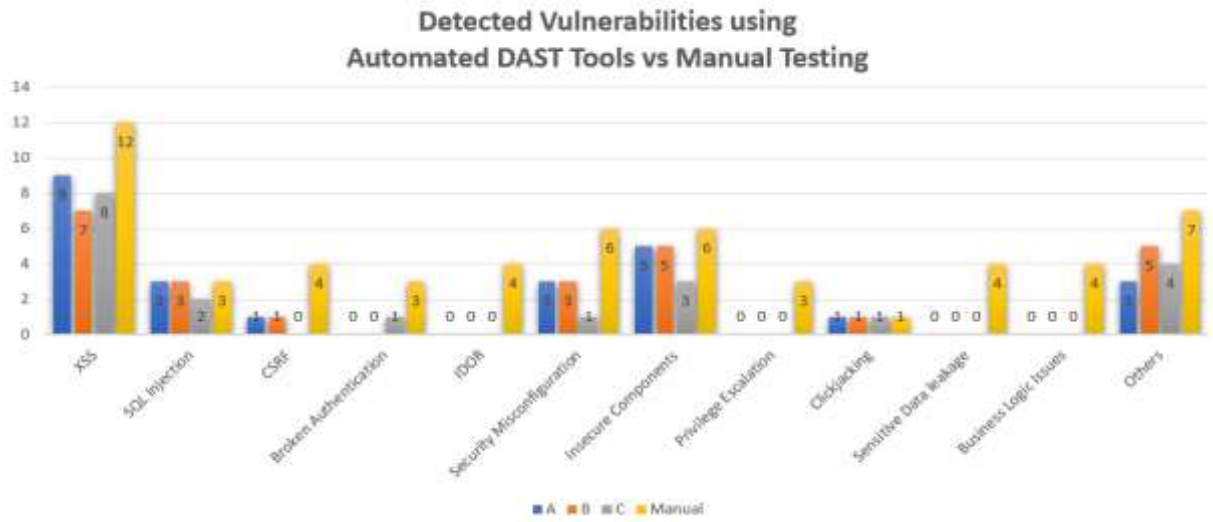
Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.10534

From the above tables, it is evident that manual penetration testing outperforms automated scanning of web applications. Automated scanners are proficient in detecting security vulnerabilities like XSS and SQL injection, but their performance was poor in detecting sensitive data leakage and Business Logic Issues.

Out of the total 12 XSS Vulnerabilities, Scanners A and B were able to detect 9 and 7 issues, respectively. These issues consist of Reflected and Stored XSS. All the three scanners couldn't detect Blind XSS in the test. 2 out of 3 Scanners could detect all the 3 SQL injection Vulnerabilities in the application. In the case of Cross Site Request Forgery (CSRF), The scanners detected them with an accuracy of 50%.

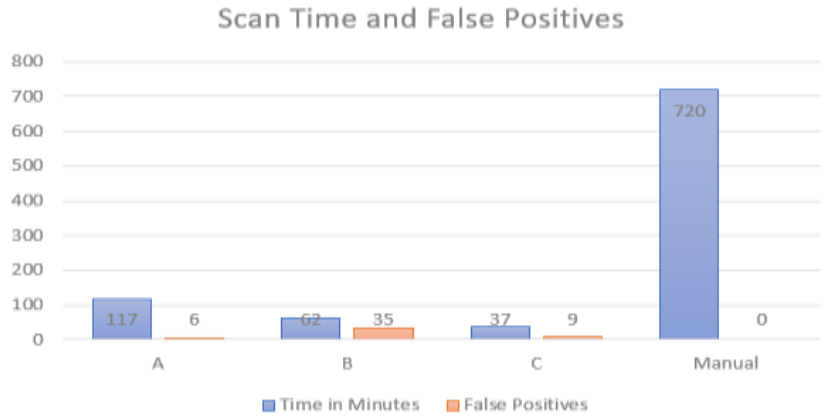
Issues like Insecure Direct Object Reference, Sensitive Data Leakage, and Business Logic Issues were not detected by any of the scanners.



The above chat shows that Manual testing detected more security vulnerabilities than the usage of Automated Security Testing Tools. Especially in the case of Sensitive data leakage, IDOR, and Business Logic Issues. Manual testing by professionals is needed for testing and detecting such issues as business logic and workflow differ from one application to the other. Other issues such as XSS, SQL Injection, Security Misconfigurations, etc. are detected based on preset rules which are matched from server responses for the requests from the scanner.

CONSIDERATIONS

When considering the scan times and the number of false positives, Automated Scanners consume less than 20% of the time than manual testing but the number of false positives in the scan results is higher in the case of automated testing.



IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 5, May 2021

DOI 10.17148/IJARCCE.2021.10534

Testing web applications using automated scanners is a cost-effective way of ensuring web application security, but the security vulnerabilities reported may not be complete and there may be still issues undetected such as sensitive data leakage or business logic vulnerabilities. The number of false-positive findings are also higher when using automated tools. On the other hand, manual penetration testing of web applications is better for ensuring the security of the applications, but that comes with a cost, and the time taken for the tests and the associated costs will be considerably high. Automated testing tools are capable of examining hundreds of server responses per second, whereas this is time consuming in manual testing.

LIMITATIONS AND FUTURE WORK

While our evaluation included three widely used commercial Web Application Security Scanners, our results might not be generalized for all Commercial and Open-source Web Application Security Tools. Our test considered the most common security vulnerabilities only for a single dynamic web application with 300+ Pages. Future work might also consider including other commercial and open-source vulnerability scanners and repeating the test multiple times to find inconsistencies in the scan results.

CONCLUSION

The need for security of web-based applications is higher than ever. This study compared the strengths and limitations of automated DAST tools for scanning web applications. Through this study, it was made clear that automated security testing tools are faster but manual testing is required for finding all vulnerabilities including business logic issues that cannot be detected by the scanners. So, A combination of automated and manual testing is better for making sure that a web application is safe from attackers.

REFERENCES

1. Beyond Moore's law: Internet growth trends. L.G Roberts, IEEE, 2000
2. Web-based Data Leakage Prevention. Sachiko Yoshihama, Takuya Mishina, and Tsutomu Matsumoto, IBM Research, 2010
3. Some Trends in Web Application Development, Mehdi Jazayeri, IEEE, 2002
4. Black Box and White Box Testing Techniques –A Literature review, Srinivas Nidhra, and Jagruthi Dondeti, IJESA, 2012
5. A Comparative Study of White Box, Black Box, and Grey Box Testing Techniques. Mohd. Ehmer Khan, Farmeena Khan, IJACSA, 2012
6. Vulnerabilities Mapping based on OWASP-SANS: A Survey for Static Application Security Testing (SAST), Jinfeng Li, AETiC, 2020
7. Evaluation of web vulnerability scanners, Yuma Makino, Vitaly Klyuev, IEEE, 2015.