Automated Testing: Testing Top 10 OWASP Vulnerabilities of Government Web Applications in Bangladesh

Azaz Ahamed
Computer Science & Engineering
Independent University, Bangladesh
2120637@iub.edu.bd

Nafiz Sadman
Silicon Orchard Ltd.
Bangladesh
nafiz@siliconorchard.com

Touseef Aziz Khan
Computer Science & Engineering
Independent University, Bangladesh
2120638@iub.edu.bd

Mahfuz Ibne Hannan
Computer Science & Engineering
Independent University, Bangladesh
2120635@iub.edu.bd

Farzana Sadia
Dept. of Software Engineering
Daffodil International University, Bangladesh
sadia_swe@diu.edu.bd

Mahady Hasan
Computer Science & Engineering
Independent University, Bangladesh
mahady@iub.edu.bd

***Abstract*—With an increase in the popularity of the Internet, there is also a rise in the number of security threats and vulnerabilities. The Open Web Application Security Project (OWASP) is an online community-driven project that provides a set of 10 most crucial security vulnerabilities to monitor and mitigate to have safer Internet connectivity. Automated software testing provides invaluable insights into the current situation regarding OWASP Top 10 2017 vulnerabilities for Web applications from the five sectors of the Bangladesh Government. In this research, comprehensive testing has been carried out using BurpSuite, ZAP and Netsparker to see recurring vulnerabilities among the sections of Web applications. We draw data-driven comparisons between these tools and evaluate them against Web applications from respective sectors and the results are presented accordingly. We found the Services and the Transportation sectors to be most vulnerable.**

***Index Terms*—Software Testing; Automated Testing; OWASP; Web Vulnerability; Testing Tools**

I. INTRODUCTION

With the advancement and adoption of Web technology, more and more government services are provided by online Web applications these days. Sophisticated online portals are visited by thousands of citizens every day as they become more and more reliant on their convenience. Naturally, Web applications like these store sensitive user information. With such adoption, comes great concern for security to protect the data and privacy of the users of such platforms [1]. Newer and more sophisticated attacks need to be monitored around the clock due to the advancements in Web technology [2].

Current issues remain where Web applications like these are not properly or regularly tested for OWASP (Open Web Application Security Project) vulnerabilities as seen in test results disseminated in later sections of this paper. Our purpose is to understand how the selected government Web applications from different sectors fare against the testing tools.

According to a news article [3], approximately 147–200 Bangladeshi entities, including government agencies, were exploited during April 2021. Before this, according to another article in 2019 [4], 3 private banks were targeted and exploited to steal almost $3 million. In 2016 in a similar type of heist, the hackers stole approximately $81 million from Bangladesh Bank's federal reserve. These are only a handful of the reported attacks.

This paper provides a selection of three automated vulnerability testing tools to find the OWASP Top 10 2017 vulnerabilities [5], [6]. The tools used in this research were: Netsparker, BurpSuite, and ZAP. The main motivations for selecting these three tools were the availability of existing security research [7], OWASP compliant threat detection features, reporting, and documentation of usage. According to past research, they are fast and reliable, especially when tested against known vulnerabilities [8]. The different sectors of government Web applications that were tested using the tools are Services, Telecommunication, Welfare, Health, and Transportation. Each test was run three times on the same Web application using the same tool. This methodology was adapted to mitigate any inconsistencies and establish a measurement of correctness for each testing tool. The results were averaged to come to a consistent comparison between the sectors, and testing tools were carefully compared for consistency. Common vulnerabilities among the Web applications are pointed out and different sectors are compared against one another based on how secure they are.

In this paper, we intend to:

- Explore the three popular OWASP compliant testing tools and their effectiveness in finding and recording OWASP's top 10 vulnerabilities.
- Test live Bangladesh Government Web applications, find their vulnerabilities and draw a comparison between them.

Our target is to find answers to the following questions:

 ISBN: 978-1-61208-997-3

- What is the current status of Bangladesh's sensitive Government Websites in terms of vulnerabilities?
- Which testing tools used in this research can best detect the most vulnerabilities of the Bangladeshi Government Websites?
- What do these vulnerabilities tell us about the Websites?

The organization of the paper is as follows: In Section 2, we present a brief technical background on OWASP and its list of top 10 vulnerabilities. In Section 3, we look into different literature reviews. We present our research methodology in Section 4 and discuss the results in Section 5. Finally, we conclude our research and future scope in Section 6.

II. TECHNICAL BACKGROUND

In this section, we briefly introduce OWASP. It is a non-profit, community-driven project whose primary aim is to study contemporary vulnerabilities in modern Web applications. The foundation present a set of standards for identifying the severity of the vulnerabilities, their possible causes, and their mitigation plans.

Table I summarizes the Top 10 OWASP 2017 Vulnerabilities with their acronyms. Throughout this paper, we will use these acronyms to denote corresponding vulnerabilities.

TABLE I
SUMMARY OF TOP 10 OWASP 2017

Vulnerabilities	Description	Denotations
A1:2017	Injection	A1
A2:2017	Broken Authentication	A2
A3:2017	Sensitive Data Exposure	A3
A4:2017	XML External Entities (XXE)	A4
A5:2017	Broken Access Control	A5
A6:2017	Security Misconfiguration	A6
A7:2017	Cross-Site Scripting (XSS)	A7
A8:2017	Insecure Deserialization	A8
A9:2017	Using Components with Known Vulnerabilities	A9
A10:2017	Insufficient Logging & Monitoring	A10

The OWASP Top 10 is a list of the ten most important and common vulnerabilities that may be found in most Web applications. The popularity of the Top 10 list enabled its adoption as a standard in many vulnerability testing tools. In Table 1, we can see the list of the OWASP Top 10 vulnerabilities and their denotations. All the tools used in this research adhere to the Top 10 classes of OWASP to report found vulnerabilities. BurpSuite, Netsparker, and ZAP are all OWASP compliant. They provide rich reports which accurately identify vulnerabilities according to the OWASP Top 10 classification. After finding the class of a vulnerability, we can refer to the OWASP Website to get a better understanding of its severity and possible mitigation ideas.

III. LITERATURE REVIEW

Deployed Websites often come with several vulnerability issues. These vulnerabilities have been extensively studied and systematically categorized into vulnerability standards. Batch-Nutman [5] studied the most frequent Web application vulnerabilities that can help firms better secure their data from such threats. The research aimed to help users and developers be better equipped to deal with the most common attacks and design strategies to avoid future attacks on their Web apps. The author tested 10 vulnerabilities listed in OWASP (Open Web Application Security Project) Top 10 [9] designed and developed a secure Web application by following the guidelines of the OWASP. The paper focused on the mitigation of Web application vulnerabilities through configuration changes, coding, and patch application. SQL injection, broken authentication, sensitive data exposure, broken access control, and XML external entities are among the OWASP top ten vulnerabilities. The Web application's security has been tested and proven to have a defense mechanism in place for the aforementioned vulnerabilities. There are several Web application vulnerability testing works [10]–[12]. Yulianton et al. [13] proposed a framework to detect Web application vulnerabilities using a combination of Dynamic Taint Analysis, Static Taint Analysis, and Black-box testing. The research showed that the combination of Dynamic and Static Taint Analysis fed to Black-box testing as metadata yielded greater accuracy and fewer false positives of vulnerabilities. The aforementioned research gives us the potential attacks on Websites and potential mitigations. However, the importance of testing during the development period is emphasized by Rangau et al. [14], who explained the emergence of DevSecOps [15] and how it has come to be important due to fast-paced deployment and a lack of proper security documentation. Afterward, they introduced tools like ZAP, JMeter, Selenium, etc., to implement dynamic testing for Web applications in CI/CD pipelines. Interestingly, the authors in [16] initially expressed concern regarding manual testing how many resource it requires, and the complications it introduces. Later the advantages of the Automated Testing tools are explored and the findings indicate to 68-75% increase in efficiency when it comes to time and effort in testing.

In this research, we test the effectiveness of three different testing tools namely Netsparker, BurpSuite, and ZAP on Bangladeshi Government Service Websites based on OWASP Top 10 2017 Web vulnerabilities. Comparative analysis of testing tools [1], [6], [17]–[19] focused on determining the efficiency of load testing and detection of several Web attacks, studying pen testing on several Web applications, and how network information is gathered using various tools about Websites to find the possibility of cyberattacks. Anantharaman et al. [8] discussed OWASP A09:2017 (i.e., Using Components with Known Vulnerabilities) and pointed out several ways software can be component-based vulnerable proof by having updated technological stacks and secure SSL. They have also noted some standard developer and tester practices and tools like BurpSuite that can help prevent vulnerabilities.

Various scanners have also been compared to check which type of scanners can detect the maximum vulnerabilities. [20] presented a systematic comparison between ZAP and Arachni testing tools across four vulnerabilities (SQL injection, XSS, CMDI, and LDAP). The authors used OWASP and WAVSEP as benchmarks. The authors conclude that ZAP outperformed Arachini and recommended that OWASP be used as standard benchmarking to evaluate testing results. [21] presented a

 ISBN: 978-1-61208-997-3

comparative study of 8 Web vulnerability scanners (Acunetix, HP WebInspect, IBM AppScan, OWASP ZAP, SNLS-VK, Arachni, Vega, and Iron WASP) and tested on WebGoat and Damn Vulnerable Web Application (DVWA) which have pre-built vulnerabilities. Precision, recall, Youden index, OWASP Web benchmark evaluation (WBE), and the Web application security scanner evaluation criteria (WASSEC) were used to evaluate the performances of the tools. The authors concluded that all of the testing tools require improvement in terms of code coverage, detection rate, and reducing the number of false positives. Karangle et al. [22] compared modern security scanning tools, such as Uniscan and ZAP tools for testing vulnerabilities in Web applications through experimentation on 20 Websites. The paper goes into detail about penetration testing using these tools and how the Website URL's information is gathered to find the possibility of cyberattacks. Ultimately, they found that the ZAP tool performed faster than Uniscan, but Uniscan performed a deeper vulnerability analysis.

The scope of our research involves comparing the effectiveness of the three tools in terms of capturing the vulnerabilities listed in OWASP Top 10 2017 on the Bangladeshi Government Web services. Setiawan et al. [23] performed vulnerability analysis for government website applications and carried out using the Interactive Application Security Testing (IAST) approach. The study used three tools, namely Jenkins, API ZAP, and SonarQube. Moniruzzaman et al. [24] performed a systemized combination of black box and white box testing to detect vulnerabilities of different Bangladeshi Government and popular Websites using most of the common testing tools. They have found out that about 64% of the selected Websites are at risk of vulnerabilities. However, we also explicitly point out in our study the consistencies and inconsistencies in these tools.

IV. RESEARCH METHODOLOGY

To ensure proper documentation of vulnerabilities reported by each tool for each Website, an individual tool was run three times on each Website to give a proper baseline. The following data was collected from every test run:

- Number of runs (number of test runs on the same Website using the same tool).
- Time taken to complete the test (in minutes).
- Counts of vulnerabilities found for severities: Low, Medium, and High.
- Total number of vulnerabilities.

Multiple tests were run this way to understand the consistency of vulnerability reporting for each tool for a particular Website. Running multiple tests also helped to understand the UI/UX of the individual tools, which essentially gave us a way to judge their usability, accessibility, and applicability for finding OWASP Top 10 vulnerabilities.

Another goal was to determine how the government Web applications compare to each other in terms of vulnerabilities and determine if there is a correlation between the popularity of a Website and the number of vulnerabilities found there. For this, five government Web applications from different sectors were thoroughly tested with the testing tools for OWASP Top 10 vulnerabilities. The sectors are:

- Services
- Transportation
- Welfare
- Healthcare
- Telecommunications

After collecting the data found through testing, all the vulnerabilities were cross-matched with their corresponding OWASP vulnerabilities category and presented in a graphical format. There, we see the severity of the vulnerabilities: vulnerabilities found in sectors by tool and overall vulnerabilities found in different sectors.

Figure 1 represents our methodology workflow.

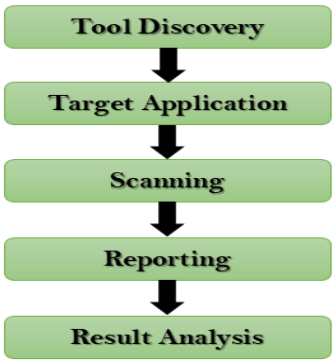


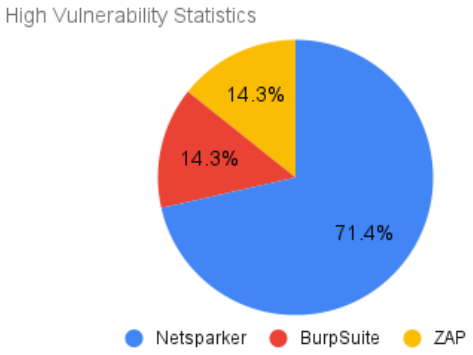
Fig. 1. Workflow of our methodology.

We can break down the process into 5 steps.

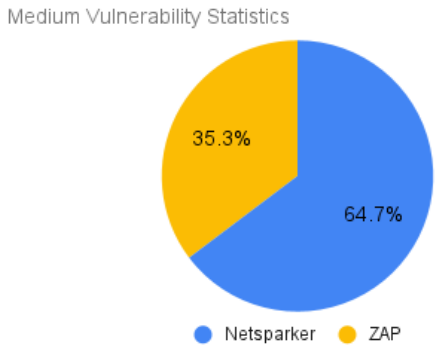
1) **Tool Discovery**: In this initial phase, the tools we selected were the most popular among their class for OWASP, as they had many resources and documentation and also, according to [7].
2) **Target Application**: In the target application phase, we chose some government Web applications that were used by citizens of the country and narrowed it down to the top 5 government Web applications.
3) **Scanning**: Several activities are carried out during this phase to perform vulnerability scanning. The vulnerability scan's goal is to identify a list of vulnerabilities in the test target. This study uses three tools: Netsparker, BurpSuite, and ZAP.
4) **Reporting**: During this phase, the tester/developer will document the possible results generated by the three tools throughout the vulnerability assessment process.
5) **Result Analysis**: In this final phase, the tester/developer analyzes the discovered vulnerability under the OWASP Top Ten 2017.

[24] tested several Bangladeshi Government Websites, but the vulnerability was mapped with Common Vulnerabilities and Exposures (CVE) and the test category was limited to 5 vulnerabilities. In this study, we covered major Bangladeshi Government Websites and presented an overview of vulnerabilities as per OWASP Top 10 2017 in the selected sites.

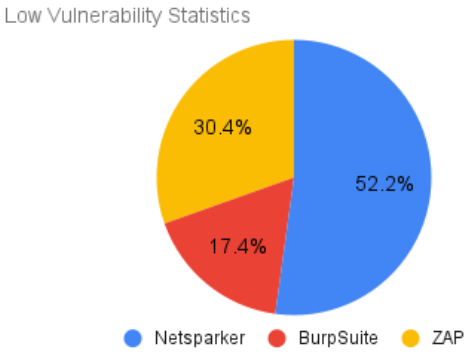
 ISBN: 978-1-61208-997-3



(a) High Vulnerability



(b) Medium Vulnerability



(c) Low Vulnerability

Fig. 2. Vulnerability statistics of each of the testing tools on our targeted Websites.

Our classification of High, Medium and Low vulnerabilities was aided by the OWASP Risk Factor (RF) table documented by [25]. Here, the top 10 vulnerability classes are given a Risk Factor score based on exploitability, security weakness detectability, and their technical impact on business. We have classified severity based on RF scores in the following manner:

- **LOW** *if* $4 \leq RF < 5$
- **MEDIUM** *if* $5 \leq RF < 7$
- **HIGH** *if* $RF \geq 7$

As all the testing tools report the classes of OWASP vulnerabilities, they are directly comparable to the RF scores and their corresponding severity classes.

Using the 2017 OWASP standard may seem like a limitation of this research when OWASP has announced 2021 standards. All the tools used to test vulnerabilities in this research have the option to generate reports for OWASP Top 10 2017. Therefore, we have selected OWASP 2017 as it is compatible with all the tools and the results can be compared consistently. OWASP 2021 classifications are not yet fully integrated with most vulnerability assessment tools on the market.

V. RESULT ANALYSIS

In this section, we dive deep into our findings in this research. We believe that the collected data gives us an accurate picture of the security and testing aspects of Government Web applications that deal with millions of sensitive user data every day.

If we take Figure 2 into account, we can observe that Netsparker can record the highest number of threats compared to BurpSuite and ZAP. According to the findings in Figure 2a, Netsparker records 5 times more high vulnerability threats than BurpSuite and ZAP. From Figure 2b, we can also observe that BurpSuite did not detect any Medium level threats which indicate that BurpSuite might not have enough features to test parameters when compared to Netsparker or ZAP. In Figure 2c, we can conclude that Netsparker had recorded the highest number of low vulnerability threats followed by ZAP then BurpSuite.

In Figure 3, we can see a breakdown of vulnerabilities and their severity for each sector of Web applications. OWASP vulnerabilities found by all the tools are aggregated for each sector of Web applications and classified into severity categories, as discussed earlier, based on RF scores. As we can see from the graph, Transportation and Services have the highest numbers of high-severity vulnerabilities. The high-severity vulnerabilities in Transportation and Service Web applications may result in the following scenarios:

1) *A1 - Injection*: Where unauthorized users can gain access to sensitive data to discover personal National Identity or License Information with malicious intent.

 ISBN: 978-1-61208-997-3

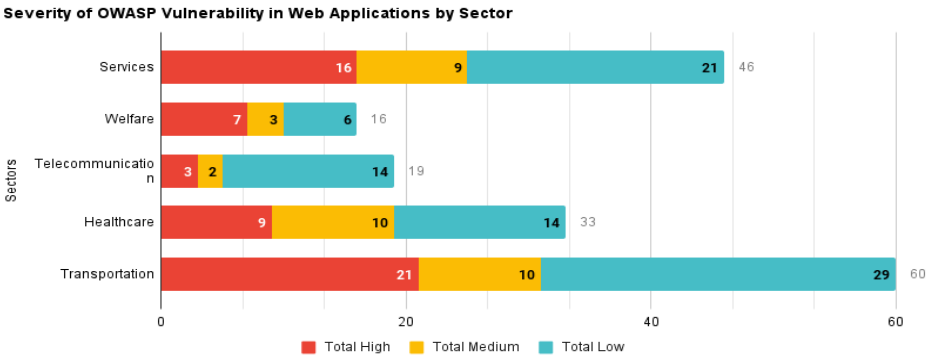


Fig. 3. Severity of OWASP Vulnerability in Web Applications by Sector.

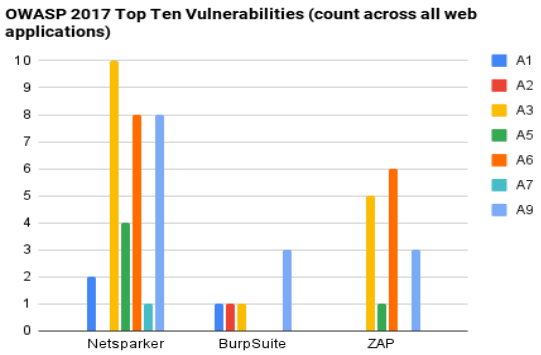


Fig. 4. OWASP 2017 Top 10 Vulnerabilities (Count across all Web applications).

2) *A2 - Broken Authentication*: Where attackers can use brute force to potentially gain access to make unauthorized changes to user identity information or forge official documents.
3) *A3 - Sensitive Data Exposure* Attackers exploit weak public/private key generation to gain access to data in transit using man-in-the-middle attacks to steal user details for illegal promotional material.
4) *A4 - XML External Entities* Older XML may enable uploading of hostile XML code through Uniform Resource Identifier (URI) network requests. This may allow remote code execution and possible denial-of-service attacks. This may result in system service downtime, causing very difficult circumstances.

Consequently, if we take a look at the medium/high-severity vulnerabilities in Healthcare, the following attack scenarios might also be true:

1) *A5 - Broken Access Control*: Where unauthorized users can make changes without any required permission. Here, the medical information of a user can be altered to make fake prescriptions to purchase unauthorized drugs.
2) *A7 - Cross Site Scripting*: Here, attackers can directly manipulate a user's browser to alter information using their credentials with remote code execution. This might result in false reports of ailments and also theft of user credentials.

These scenarios hold for the other sectors of Web applications. Low-severity vulnerabilities are not actively threatening, but may be exploitable in niche cases. Web applications from the Telecommunication sector had the lowest number of high/medium-severity vulnerabilities. We have an assumption that it may be more secure as the sector relies almost entirely on current technology and practices. Surprisingly, Welfare reported the lowest number of vulnerabilities in general, though it had a higher number of high-severity vulnerabilities than the Telecommunication sector. We think this may be due to smaller and relatively newer Web applications built with current tools and practices, whereas the applications in the Telecommunication sector were relatively mature.

According to Figure 4, we can observe that Netsparker detected the most vulnerabilities among the three tools, while ZAP was second and BurpSuite detected the least amount. It must also be noted that even though Netsparker and ZAP have similar results, Netsparker found more vulnerabilities such as A1 and A7, which ZAP did not detect. ZAP also failed to scan Welfare and Telecommunication Web applications, whereas Netsparker managed to scan all the Web applications. BurpSuite failed to scan the Transportation Web application after running for a long time. This indicates that ZAP and BurpSuite only executed surface-level scanning, while Netsparker did much deeper vulnerability analysis while scanning the Web applications. However, BurpSuite detected A2, which neither Netsparker nor ZAP were able to detect. If we observe the time taken by the testing tools to scan each Web application, ZAP is the fastest among all the tools, while BurpSuite comes in second and Netsparker takes the longest overall.

Across three runs for each Website with each testing tool, the test run with the highest number of classified vulnerabilities is kept. In this Figure 4, we can see the total number of OWASP Top 10 vulnerability classes found with each testing tool across all the tested applications.

Figure 5 illustrates the vulnerabilities discovered in various sectors. According to the chart, A6 (Security Misconfiguration) appears to be the most common vulnerability, closely

 ISBN: 978-1-61208-997-3

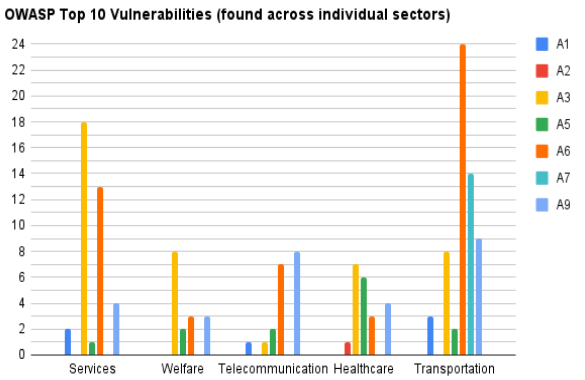


Fig. 5. OWASP 2017 Top 10 Vulnerabilities (Found across individual sectors).

followed by A3 (Sensitive Data Exposure) and A9 (Using Components With Known Vulnerabilities). According to the remediation provided in [5], we can resolve these issues for all of the vulnerabilities. Based on our targeted applications, we can see that Transportation is the most vulnerable, followed by Services.

Transportation plays a major role in the daily lives of citizens. According to Bangladesh Transport Data [26], there are approximately 4.5 million registered vehicles in Bangladesh as of June 2020. Based on this relevant data, we can conclude that there are approximately 4.5 million users of these applications whose information is at high risk. The second most dangerous application we discovered was Services, which includes business information, citizen information, banking services, etc. Bangladesh's current population is 164.7 million as of 2020, with some of them directly or indirectly using services whose data are at risk. Vulnerabilities in healthcare are much lower than what we found in Transportation and Service sectors. Telecommunications and Welfare have the lowest number of vulnerabilities. According to our findings, Severity can manifest itself in a variety of ways, such as A3, SSL (Secure Sockets Layer) certificates about to expire, session cookies not marked as secured, and weak ciphers enabled, and so on. For A6, it could be an insecure framework or a DB (database) user with administrative privileges, among other things. In A9, it typically deals with Apache, Tomcat, Bootstrap, JQuery, OpenSSL, PHP, Nginx, and other versions. These are some of the most common threats discovered in our research that developers may have overlooked. This has a domino effect, putting sensitive information at risk, such as registered holders' data, etc.

From the vulnerability results we discovered in this research, we can see that government Web applications in Bangladesh suffer from important security oversights. Most of the vulnerabilities arise from common software development pitfalls such as:

- Not writing maintainable code
- Not writing reusable code
- Not writing unit or integration tests
- Not maintaining up-to-date documentation of the project
- Not updating software packages used in software development

These are some of the steps that help catch 99% of the security issues or bugs in Software Development.

It is positively alarming to discover a high number of high-severity vulnerabilities in 3 out of 5 selected sectors of government Web applications. We hypothesize that this is due to the current state of maturity in technology in Bangladesh. More in-depth analysis and studies are required to validate this hypothesis, and we believe this to be an interesting arena for future research.

VI. CONCLUSIONS

In this research, we have compiled, compared, and contrasted the data collected to see how effectively BurpSuite, Netsparker and Zap record OWASP Top 10 vulnerabilities and report them. We have also seen OWASP vulnerabilities across all tested applications and common vulnerabilities and referred to their remediation possibilities.

The driving purpose of this study was to understand OWASP vulnerabilities and their impact on the sectors of Government Web applications in Bangladesh. Our target was to run as many tests using as many tools as possible to find consistent results of vulnerabilities. Additionally, during this research, we understood and learned how automated testing tools work. Though only three testing tools are used, in the future we intend to expand the list and draw a comparison between many other popular tools. In future research, we would like to find if specific tools are better for a specific type of Website e.g., Lighthouse [27] for Progressive Web Applications).

All the tests done in this paper are exclusively black box testing with a specified scan policy. This is due to not having direct access to source code to perform any static analysis. We believe that a combination of both black box and static analysis white box testing will provide better and deeper insight. Yulianton et al. [13] For now, black-box testing only provides us with information about what and how many vulnerabilities are there, but the crux of the issue might arise from the more apt question, "Why are the vulnerabilities there in the first place?"

Our next task is to broaden the scope of Websites used for testing vulnerabilities and use the latest OWASP Top 10 2021 guidelines. Along with black-box testing, we also look forward to adding white-box and grey-box testing. Having more insights can provide solutions toward better software development methodologies with strict adherence to testing guidelines.

REFERENCES

[1] L. F. de Lima, M. C. Horstmann, D. N. Neto, A. R. Grégio, F. Silva, and L. M. Peres, "On the challenges of automated testing of web vulnerabilities," in *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 2020, pp. 203–206.

[2] A. Sołtysik-Piorunkiewicz and M. Krysiak, "The cyber threats analysis for web applications security in industry 4.0," in *Towards Industry 4.0—Current Challenges in Information Systems*. Springer, 2020, pp. 127–141.

[3] S. Rahman, "Latest cyber attack hit at least 147 bangladeshi entities," 2021. [Online]. Available: shorturl.at/qrz36
[4] S. Rehman, "Three banks hit by cyberattacks," 2016. [Online]. Available: https://www.thedailystar.net/frontpage/news/three-banks-hit-cyberattacks-1760629
[5] M. Bach-Nutman, "Understanding the top 10 owasp vulnerabilities," *arXiv preprint arXiv:2012.09960*, 2020.
[6] R. S. Devi and M. M. Kumar, "Testing for security weakness of web applications using ethical hacking," in *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*. IEEE, 2020, pp. 354–361.
[7] F. Ö. Sönmez and B. G. Kiliç, "Holistic web application security visualization for multi-project and multi-phase dynamic application security test results," *IEEE Access*, vol. 9, pp. 25 858–25 884, 2021.
[8] N. Anantharaman and B. Wukkadada, "Identifying the usage of known vulnerabilities components based on owasp a9," in *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)*. IEEE, 2020, pp. 88–91.
[9] S. K. Lala, A. Kumar, and T. Subbulakshmi, "Secure web development using owasp guidelines," in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2021, pp. 323–332.
[10] D. Omeiza and J. Owusu-Tweneboah, "Web security investigation through penetration tests: A case study of an educational institution portal," *arXiv preprint arXiv:1811.01388*, 2018.
[11] N. Abdinurova, M. Galiyev, and A. Aitkulov, "Owasp vulnerabilities scanning of a private university websites," *Suleyman Demirel University Bulletin: Natural and Technical Sciences*, 2021.
[12] F. Holík and S. Neradova, "Vulnerabilities of modern web applications," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2017, pp. 1256–1261.
[13] H. Yulianton, A. Trisetyarso, W. Suparta, B. S. Abbas, and C. H. Kang, "Web application vulnerability detection using taint analysis and black-box testing," in *IOP Conference Series: Materials Science and Engineering*, vol. 879, no. 1. IOP Publishing, 2020, p. 012031.
[14] T. Rangnau, R. v. Buijtenen, F. Fransen, and F. Turkmen, "Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines," in *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, 2020, pp. 145–154.
[15] "What is devsecops?" Apr 2018. [Online]. Available: https://www.redhat.com/en/topics/devops/what-is-devsecops
[16] M. Hanna, A. E. Aboutabl, and M.-S. M. Mostafa, "Automated software testing framework for web applications," *International Journal of Applied Engineering Research*, vol. 13, no. 11, pp. 9758–9767, 2018.
[17] R. Abbas, Z. Sultan, and S. N. Bhatti, "Comparative analysis of automated load testing tools: Apache jmeter, microsoft visual studio (tfs), loadrunner, siege," in *2017 International Conference on Communication Technologies (ComTech)*. IEEE, 2017, pp. 39–44.
[18] S. Tyagi and K. Kumar, "Evaluation of static web vulnerability analysis tools," in *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*. IEEE, 2018, pp. 1–6.
[19] D. Dagar and A. Gupta, "A comparison of vulnerability assessment tools owasp 2.7. 0 & pentest on demo web application," *CPJ Global Review A National Journal of Chanderprabhu Jain College of Higher Studies*, pp. 46–50.
[20] B. Mburano and W. Si, "Evaluation of web vulnerability scanners based on owasp benchmark," in *2018 26th International Conference on Systems Engineering (ICSEng)*. IEEE, 2018, pp. 1–6.
[21] R. Amankwah, J. Chen, P. K. Kudjo, and D. Towey, "An empirical comparison of commercial and open-source web vulnerability scanners," *Software: Practice and Experience*, vol. 50, no. 9, pp. 1842–1857, 2020.
[22] N. Karangle, A. K. Mishra, and D. A. Khan, "Comparison of nikto and uniscan for measuring url vulnerability," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2019, pp. 1–6.
[23] H. Setiawan, L. E. Erlangga, and I. Baskoro, "Vulnerability analysis using the interactive application security testing (iast) approach for government x website applications," in *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*. IEEE, 2020, pp. 471–475.
[24] M. Moniruzzaman, F. Chowdhury, and M. S. Ferdous, "Measuring vulnerabilities of bangladeshi websites," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. IEEE, 2019, pp. 1–7.
[25] A. van der Stock, B. Glas, N. Smithline, and T. Gigler, "Owasp top 10 2017: The ten most critical web application security risks," *OWASP Foundation*, p. 23, 2017.
[26] A. M. A. Obaida, "Number of motor vehicles," Aug 2022. [Online]. Available: http://dsce.edu.bd/db/Number_of_Motor_Vehicles
[27] "Overview - lighthouse," May 2022. [Online]. Available: https://developer.chrome.com/docs/lighthouse/overview/

 ISBN: 978-1-61208-997-3