Exploiting Vulnerabilities in Weak CAPTCHA Mechanisms within DVWA

Mohammad Shinaz Bhanu[1], Durgam Varshini[2], Poosala Srikanth[3], Payyavula Lokesh[4]

[1]Assistant Professor, [2-4]Student, Department of CSE (Cyber Security), CMR College of Engineering & Technology, Hyderabad, India.

E-mail: [1]md.shinazbhanu@cmrcet.ac.in, [2]varshinidurgam222@gmail.com, [3]poosalasrikanth123@gmail.com, [4]plokeshlucky35@gmail.com

Abstract

This research focuses on identifying vulnerabilities in the CAPTCHA implementation of the Damn Vulnerable Web Application (DVWA). We utilize Optical Character Recognition (OCR) with Tesseract, capture internet traffic using OWASP ZAP, and develop Python-based automated scripts to bypass substandard CAPTCHA implementations. Throughout the study, we uncover critical vulnerabilities, including the lack of CAPTCHA verification for sensitive actions such as password changes. We provide a detailed step-by-step analysis of how attackers can exploit these vulnerabilities. We conclude by comparing these weak CAPTCHA methods with more robust alternatives, such as Google reCAPTCHA, and recommend best practices, including server-side validation, CAPTCHA obfuscation, and the implementation of multi-layered security systems. The research employs software tools including Tesseract OCR v5.3, OWASP ZAP 2.12.0, Python 3.10, and DVWA 1.10 on XAMPP.

Keywords: DVWA, CAPTCHA, Vulnerabilities, Exploitation, OCR, CAPTCHA Bypass, XAMPP, OWAS ZAP.

1. Introduction

CAPTCHA, or "Completely Automated Public Turing test to tell Computers and Humans Apart," has long served as a crucial defense mechanism against bots and automated scripts that

DOI: https://doi.org/10.36548/jitdw.2025.2.003
Received: 31.03.2025. received in revised form: 02.05.2025, accepted: 19.05.2025, published: 30.05.2025

attempt to exploit online services. Traditionally employed in login forms, comment sections, and registration forms, CAPTCHA distinguishes genuine human users from spam software. However, with the rapid advancement of Artificial Intelligence (AI), Optical Character Recognition (OCR), and automation tools, conventional CAPTCHA systems are increasingly vulnerable. Techniques once considered secure, such as image-based or distorted text CAPTCHAs, can now be circumvented by modern machine learning algorithms with remarkable accuracy. This decline in effectiveness poses a significant challenge for web security, particularly for applications relying on outdated CAPTCHA systems.

To explore these vulnerabilities in a controlled environment, this study utilizes the Damn Vulnerable Web Application (DVWA), an intentionally insecure PHP/MySQL web application designed for testing and educational purposes in web security. DVWA features a simplified CAPTCHA module that is deliberately vulnerable, creating an advantageous setting for demonstrating exploitation techniques. The primary issue examined is DVWA's enforcement of CAPTCHA solely during the login process, leaving critical post-authentication actions, such as password updates, unprotected. This selective enforcement creates opportunities for exploitation.

The methodology employs widely available and popular tools: Tesseract OCR (for reading CAPTCHA images), OWASP ZAP (for intercepting and manipulating HTTP traffic), and Python scripts (for automating bypass attempts). The attacker process mimics real-world exploitation: logging into DVWA, accessing the vulnerable CAPTCHA module, intercepting the password change request with OWASP ZAP, modifying CAPTCHA validation parameters, and forwarding the altered request to the server. Since DVWA lacks server-side CAPTCHA enforcement for this critical operation, the forged request is accepted without challenge, thereby revealing the security vulnerability.

This research aims not only to demonstrate the feasibility of such an attack but also to highlight broader implications: CAPTCHA schemes must be intelligently designed, uniformly applied, and validated against contemporary threat models. In addition to detailing the attack flow, this study compares weak CAPTCHA configurations, such as those in DVWA, with more secure alternatives, like Google's reCAPTCHA. Recommendations include adopting multi-stage verification, concealing CAPTCHA logic, and incorporating behavioral analysis mechanisms. The long-term objective is to promote robust and adaptive CAPTCHA systems capable of withstanding current and future evasion techniques.

2. Related Work

The contemporary web is confronted with an ongoing and changing threat environment, particularly concerning web application vulnerabilities and the related defense measures. Reference [1] gives a detailed overview of CAPTCHA measures, emphasizingon their deteriorating effectiveness as machine learning-powered automated solvers make conventional CAPTCHA measures progressively out of date. This brittleness in CAPTCHA defenses creates an imperative necessity for more robust forms of authentication. Reference [2] explores the advent of AI-facilitated cyberattacks, illustrating how attackers utilize AI not only to avoid detection but also to perform advanced phishing, evasion, and vulnerability exploitation methods, further straining traditional security defenses.

The intentional development and exploration of intentionally vulnerable web applications, e.g., DVWA (Damn Vulnerable Web Application), has become the norm in security training and research. Reference [3] examines this methodology, showing how DVWA can be used as a good platform for learning injection vulnerabilities, authentication flaws, and session management bugs. Similarly, Reference [4] addresses the OWASP API security threats, highlighting the inherent difficulty of protecting contemporary web APIs due to their sophisticated and often unexamined data exposure habits. Penetration testing continues to be at the heart of web application security testing. Reference [5] sets out typical procedures and tools employed in ethical hacking to mimic attack vectors seen in the real world, and Reference [6] contrasts commercial and open-source vulnerability scanning tools. It determines that open-source products such as OWASP ZAP provide significant utility, albeit perhaps not the same breadth or sophistication as commercial products. Reference [7] builds on this real-world application by introducing PentestHUB, a learning center intended to assist students in acquiring hands-on ethical hacking skills within an organized setting.

Cross-site request forgery (CSRF) remains one of the most significant security issues, as referenced in Reference [8], which analyzes the effectiveness of existing mitigations such as synchronizer tokens and same-site cookie attributes. Reference [9] further outlines a logic programming-based security testing framework with a knowledge-driven approach to the identification of complex and composite attacks frequently overlooked by signature-based tools. Lastly, the consequences of such vulnerabilities reach beyond prevention to detection and responsibility. Reference [10] considers digital forensics within the realm of network security, highlighting the role that thorough tracking methods and evidence-gathering protocols play in

aiding post-attack investigations and legal proceedings against cybercriminals. Together, this work details both the vulnerability of existing web defenses and the multidimensional efforts of education, tooling, and forensic examination necessary to strengthen contemporary web infrastructures.

3. Proposed Work

The suggested methodology identifies a step-by-step exploitation of a vulnerable CAPTCHA implementation within the Damn Vulnerable Web Application (DVWA) by leveraging a mix of manual interactions and automated tools. It starts with a user logging into the DVWA application, which displays a simple CAPTCHA puzzle on the login page. After solving the CAPTCHA and authenticating successfully, the user is logged into their account. Interestingly, DVWA applies CAPTCHA checks only on login and fails to perform similar testing on other sensitive actions, like password reset.

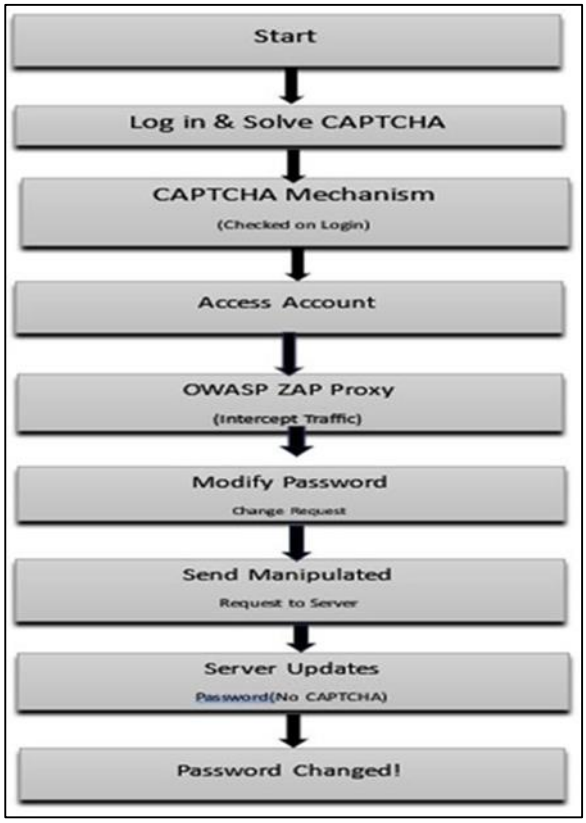

Figure 1. Flow Chart of the Proposed Work

This partial use of CAPTCHA is the foundation of the vulnerability. For the exploitation of this flaw, the attacker activates the OWASP ZAP proxy tool to listen and intercept the

communication between the web server and the client (browser). Upon initiating the password change operation, OWASP ZAP intercepts the associated HTTP request. This intercepted request is examined and manually altered to mimic a valid password update and avoid client-side validation. As no CAPTCHA validation is implemented at this point, the manipulated request passes w. server without challenge. Thus, the password is updated without further verification, demonstrating an essential security loophole. This technique illustrates the significance of applying CAPTCHA, or similar security tests, at various points of interaction, particularly in post-login operations involving account adjustments. The entire chain, from the original login through the successful tampering and completion of the password update, illustrates how attackers can exploit weak server-side controls in a systematic fashion to evade security measures. This entire process, from login to a successful CAPTCHA bypass during a password update request, is depicted in the methodology diagram, Figure 1. It draws attention to DVWA's security flaw, which is that CAPTCHA checks are only available on the login page.

4. Design Workflow

The process flow for exploiting vulnerable CAPTCHA implementations in the DVWA testing setup is shown in the diagram above and involves a number of ordered steps with the help of open-source tools. The process starts with booting up the XAMPP control panel to start the Apache server and the MySQL database, which are necessary for hosting and executing DVWA. Simultaneously, the OWASP ZAP tool is launched, and proxy settings are enabled in order to capture HTTP traffic, which is normally set to 127.0.0.1:8080. With the backend in place, the DVWA application is accessed through a web browser, and the user logging in with default credentials (admin/password). After login, the user goes to the "Insecure CAPTCHA" module available in DVWA for simulating weak security deployments. Here, a CAPTCHA-related action is carried out, e.g., altering the account password. During this request, OWASP ZAP is actively observing and intercepting the outgoing CAPTCHA request. The intercepted request is then manipulated, usually by changing or deleting CAPTCHA tokens or client-side validations that prove how weak or non-existent server-side CAPTCHA validation can be evaded. The manipulated request is sent to the server within OWASP ZAP. If the server processes this forged request successfully without returning an error, the attack is a success. Finally, the procedure ends by ensuring that the login or password change process was actually performed, which indicates the vulnerability.

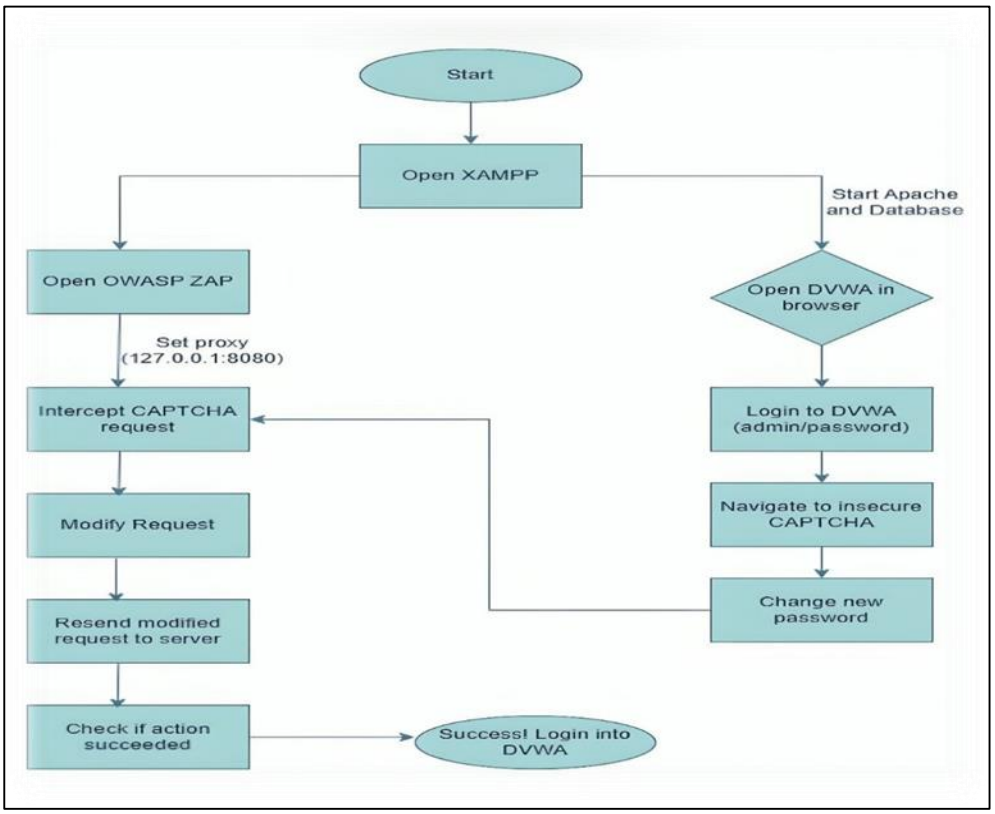

Figure 2. Workflow of the Proposed Work

This step-by-step process explains how attackers can easily exploit design flaws in CAPTCHA logic with basic but effective interception tools.

5. Implementation

The deployment of this research illustrates the exploitation of vulnerable CAPTCHA mechanisms in the Damn Vulnerable Web Application (DVWA) using a mix of manual interception and automation tools. The test environment is created using XAMPP to execute DVWA locally, and OWASP ZAP is used as a proxy to intercept HTTP requests between the client and server. The attacker authenticates with default credentials, evading the basic CAPTCHA using Tesseract OCR. The glaring weakness is the inability of DVWA to check for CAPTCHA on secure post-login operations like password updates. In an attempt to change a password, the HTTP request is intercepted through OWASP ZAP and tampered with—namely by deleting or modifying parameters associated with CAPTCHA. Since server-side CAPTCHA verification is not performed at this endpoint, the spoofed request is honored, and the password change goes by without objection. The exploit is then confirmed through logging in with the new credentials. The

process demonstrates how partial CAPTCHA enforcement, especially if not accompanied by server-side verification, makes applications susceptible to simple automated attacks.

Table 1. Detailed CAPTCHA Mechanism Comparison Table

Feature	**Traditional CAPTCHA**	**Google reCAPTCHA v3**	**Proposed DVWA CAPTCHA**	**Attack Feasibility**	**Defence Recommendation**
OCR Resistance	Low	High	Very Low	High	Use dynamic font & background distortions
Server-side Validation	Partial	Strict	Weak	Easy to bypass	Implement server-side CAPTCHA verification
Time Limiting	Rare	Enforced	None	Allows brute force	Apply rate-limiting on requests
Human Detection	Simple logic	Behavior analysis	None	Bypass via automation	Add behavioural anomaly detection
CAPTCHA update frequency	Static	Dynamic/ session based	Static	Vulnerable	Rotate CAPTCHA frequency
Replay Resistance	Low	High	Very Low	Replay successful	Add nonces or CSRF tokens
Bot Mitigation	Low	High	Low	Bots succeed	Adopt multi-factor validation

6. Results and Discussion

The project effectively showcased how weak CAPTCHA mechanisms could be compromised in DVWA, demonstrating substantial vulnerabilities. Because image distortions in CAPTCHA are small, Tesseract-OCR and some preprocessing could also decode the image accurately. Brute forcing was feasible due to predictable CAPTCHA logic. There were inconsistent server-side validations; it sometimes depended on rejected client-side validations, which could be bypassed by modifying the JavaScript. Just like in the CAPTCHA bypassed, the

CAPTCHABYPASS text was copied into readable form. Taken together, these findings suggest that gaps in CAPTCHA security must be closed if they are to withstand automated attacks.

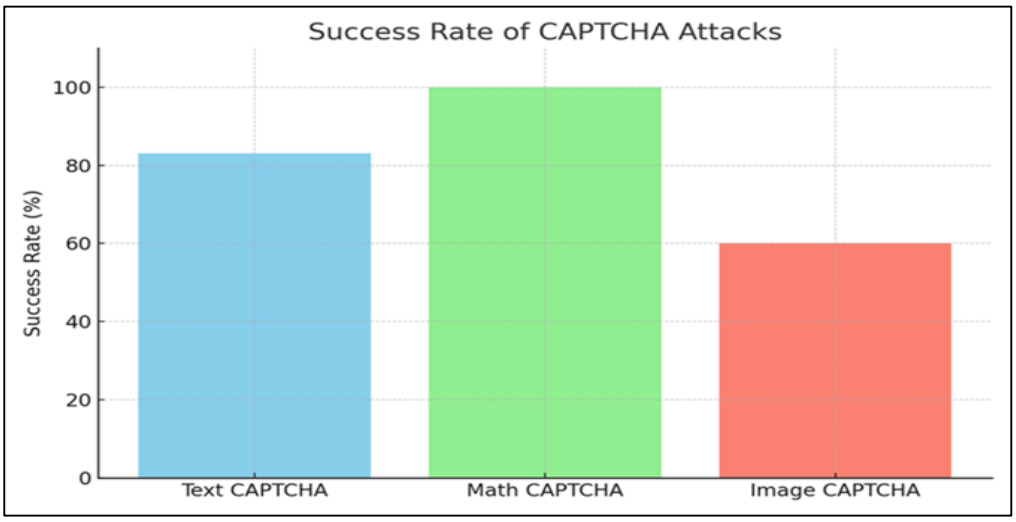

Figure 3. Success Rate of CAPTCHA Attacks

Figure 3 illustrates the differences in CAPTCHA success rates across various attempts, highlighting the variations in security levels. Additionally, it is recommended to implement advanced CAPTCHA solutions, such as Google reCAPTCHA, increase CAPTCHA complexity through dynamic distortion obfuscation, utilize server-side validation, and apply rate-limiting to mitigate brute force attacks.

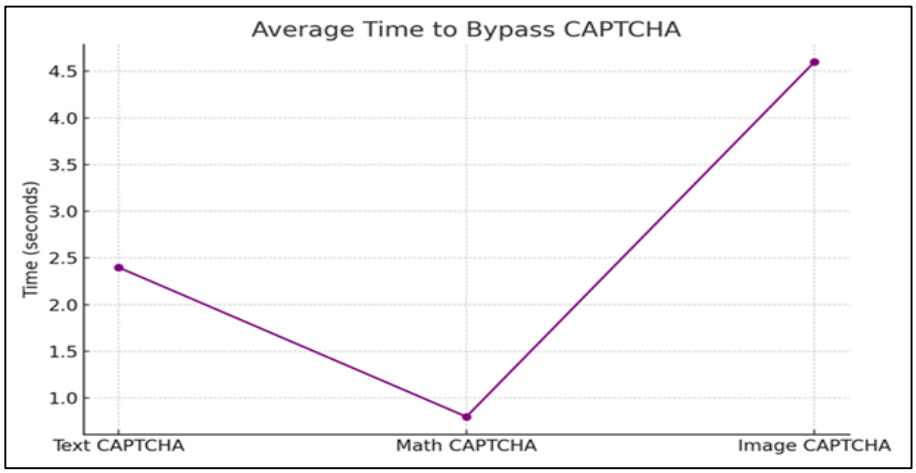

Figure 4. Average Time to Bypass CAPTCHA

Figure 4 illustrates the average time required for successful CAPTCHA bypasses, highlighting its implications for system resilience and user experience.

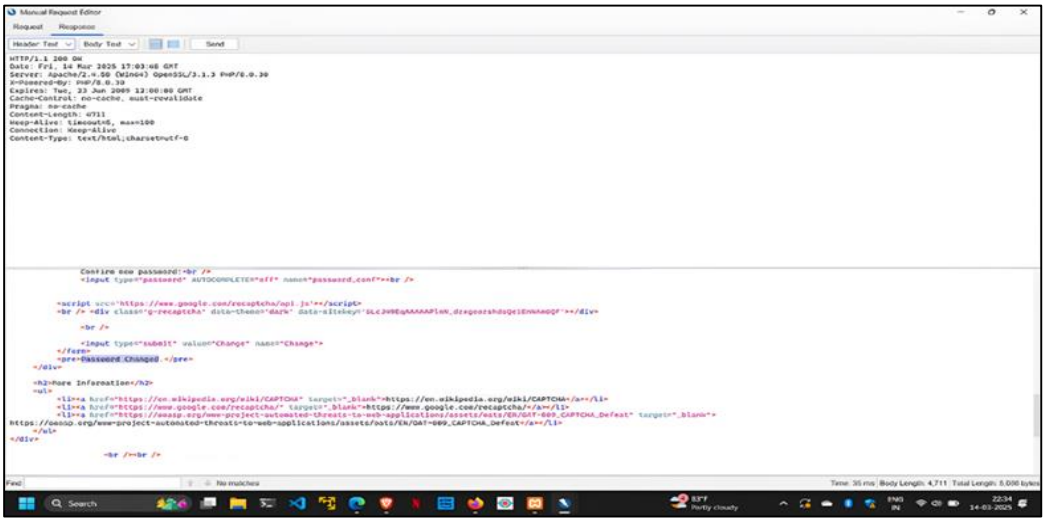

Figure 5: Successfully Modify Password

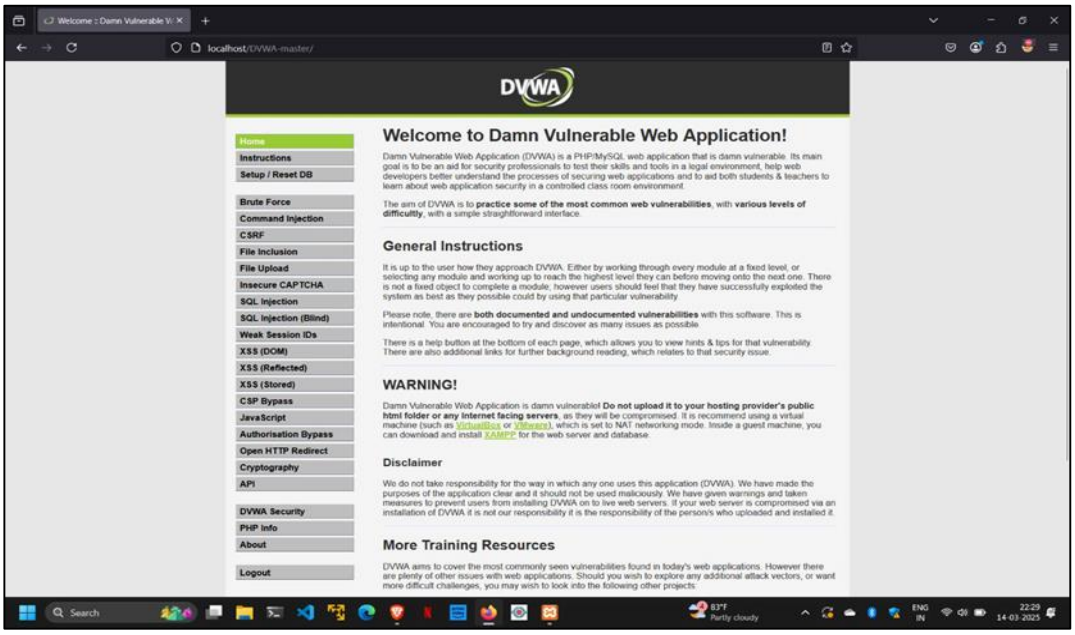

Figure 6: Successfully Login in to DVWA

A significant vulnerability was highlighted by the test's successful modification and circumvention of stored password fields, as seen in Figure 5. Unauthorized login access is made possible by the post-bypass success of CAPTCHA evasion, as shown in Figure 6.

7. Future Scope

The future scope of the project is developing new advanced generation methods of CAPTCHA that will implement machine-learning solutions for superior security. It may involve

creating CAPTCHAs that dynamically change in response to new threats and attempts to bypass them, providing better security against more sophisticated automated systems. An AI-based system that could adapt to, and counteract attempts to break CAPTCHA, would provide another layer of defense. Additionally, it will include CAPTCHAs in real-world applications across web portals, mobile applications, IoT devices, and cloud-based systems in order to evaluate their strengths and weaknesses based on performance, accessibility, and effectiveness in different environments.

8. Conclusion

In conclusion, we explored the weak CAPTCHA implementations in the Damn Vulnerable Web Application (DVWA). By demonstrating an attack as presented within this document and considering the extensive internet interactions of all users on DVWA, we increased the amount of time they were at risk of being detected due to the auto-adaptable roles that were exposed as part of the area of interest. The fact that there was insufficient corroboration stemmed from not combining variables in their designs. For example, they could have interjected randomization in both the location of characters and the texture of the background distortions. These stringed characters are general mechanisms that can be applied to defend against auto-attacks. Finally, we provided conclusions with a number of suggestions for future adapted capture from deformation findings, including suggestions for safe adapted mitigation, enhancing randomness to counteract CAPTCHA identities, and implementing dynamic capture challenges that adapt to user behavior. Greater intricacy in their distortion techniques, such as non-character overlays, should be tailored to work well across platforms and devices while still ensuring no loss of security background against competition to hide OCR-based attacks. Integrating human-driven behavior can help differentiate a female from a bot, making it more secure. The novelty of our work lies in demonstrating an end-to-end real-world CAPTCHA bypass scenario in DVWA using freely available tools like Tesseract OCR and OWASP ZAP. Unlike many theoretical papers, our work provides a reproducible and practical implementation of CAPTCHA exploitation.

References

[1] Ousat, Behzad, Esteban Schafir, Duc C. Hoang, Mohammad Ali Tofighi, Cuong V. Nguyen, Sajjad Arshad, Selcuk Uluagac, and Amin Kharraz. "The Matter of Captchas: An

Analysis of a Brittle Security Feature on the Modern Web." In Proceedings of the ACM Web Conference 2024: 1835-1846.

[2] Sampaio, Lauren Silva Rolan. "An overview of AI-enabled attacks: concepts, state-of-the-art, and evaluation of prototypes." (2021).

[3] Wang, Junmei, and Xinning Liu. "Research on Software Security Based on DVWA." In 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI), IEEE, (2023): 38-42.

[4] Idris, Muhammad, Iwan Syarif, and Idris Winarno. "Development of vulnerable web application based on OWASP API security risks." In 2021 International Electronics Symposium (IES), IEEE, (2021): 190-194.

[5] Gaur, Bonika. "Penetration Testing for Web-Applications." PhD diss., Institute of Technology, 2015.

[6] Amankwah, Richard, Jinfu Chen, Patrick Kwaku Kudjo, and Dave Towey. "An empirical comparison of commercial and open-source web vulnerability scanners." Software: Practice and Experience 50, no. 9 (2020): 1842-1857.

[7] Pinchuk, Alla D., Roman S. Odarchenko, and Oleh O. Polihenko. "Ethical hacking skills development through the PentestHUB platform." (2025).

[8] Buvana, M. "Mitigating Cross-Site Request Forgery Vulnerabilities: Evaluating Current Strategies and Proposing Defense Mechanisms."

[9] Zech, Philipp, Michael Felderer, and Ruth Breu. "Knowledge-based security testing of web applications by logic programming." International Journal on Software Tools for Technology Transfer 21, no. 2 (2019): 221-246.

[10] Oyelakin, Oyetunji, Abel Ofori-Yeboah, Aishat Ganiyu, and Oluwole Oguntoyinbo. "Digital forensics investigations and network security issues in tracking the trails of cybercriminals." In 2024 International Conference on Electrical and Computer Engineering Researches (ICECER), IEEE, (2024): 1-8.