Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 3, 2026

A SIMULATION-BASED LEARNING PLATFORM FOR TESTING WEB APPLICATION SECURITY

Reshma[1], Mujtaba Hassan[2], Mansoor Qadir[*3], Umm E Rubab[4], Sadeeq Jan[5]

[1,2]*Department of Computer Science, University of Engineering & Technology, Peshawar, 25100, Pakistan*
[*3]*Cecos University of IT and Emerging Sciences, Peshawar, 25100, Pakistan*
[4,5]*National Centre for Cyber Security, University of Engineering & Technology, Peshawar, 25000, Pakistan*

[*3]mansoor.qadir@hotmail.com

DOI: https://doi.org/10.5281/zenodo.19563929

Keywords
Simulation; Web Application; Vulnerabilities; Security; Penetration Testing.

Article History
Received: 15 February 2026
Accepted: 11 March 2026
Published: 25 March 2026

Corresponding Author: *
Mansoor Qadir

Abstract
Web applications have become an integral part of the digital world. However, with the increased use of such applications, the number of vulnerabilities has also increased exponentially in recent times. Attackers exploit these vulnerabilities for malicious purposes. One of the main reasons for such exploitation possibilities is the lack of knowledge and proper training of users and developers about such vulnerabilities. In this paper, we propose a simulation environment for web application vulnerabilities that can be utilized by normal users as well as developers to educate themselves about such vulnerabilities and also test their applications. We develop a customizable framework for the simulation of various stages and levels of vulnerabilities and practical training. Moreover, the security of applications related to the web is improved with such a platform, as it will help users to learn, identify, and assess web application vulnerabilities. Existing learning and simulation environments are very limited and struggle to keep up with the dynamic nature of web applications and emerging attack vectors. Our proposed simulation environment is Intuitive and adaptive, aiming at enhancing web application security testing. The simulation environment is tested by a number of users from various categories to check its effectiveness, accuracy, and user-friendliness compared to the existing tools and environments.

INTRODUCTION

Web application security is a major concern due to the growing reliance of both individuals and organizations on web-based systems for data storage, communication, and commerce. This same dependency on web applications has made them an appealing target for cybercriminals to leverage weaknesses in these systems to gain unauthorized access, obtain sensitive data, or halt services. As per the statistics (Abdulghaffar et, al 2023), 66% of web applications examined in 2022 were found to have vulnerabilities with high risk compared to 50%. Applications 48% of Web applications had medium-risk vulnerabilities according to the report. With these stats in mind, measures need to be effectively implemented to identify and mitigate web application vulnerabilities.

To build the simulation environment, there is an opportunity to improve the quality of web application security with previously unseen attack exploration and entrance paths. The current simulation environments are limited in terms of their coverage, flexibility, and user-friendliness; thus, a well-designed simulator would greatly help identify the vulnerabilities of the web application. It has been challenging to perform security testing

of web-based applications. The main reasons are a rapidly changing environment, a slow learning curve of security testers, and the complexities of these applications (Druin, 2021; Villegas-Ch et al., 2024). Although several learning platforms exist that help in getting yourself equipped with hands-on training for discovering vulnerabilities in web applications, they are limited and often difficult to use. For this purpose, simulation plays a vital role in skill development in safe experimentation and defense mechanisms (Veksler et al., 2018; Kavak et al., 2021). Moreover, an algorithm is introduced (Chai et al., 2018) that is based on the Elementary cellular automata (ECA), Compressive Sensing (CS), and Memristive Chaotic System (MCS). MCS ensures a large key space, makes high alert sensitive, responds to brute force attacks, and overcomes the consumption of data transmission and high resistance towards threats. But this algorithm has some criticalities, like taking a lot of time in decryption for several algorithms.

In this paper, an easy-to-use simulation environment has been proposed for learning top web application vulnerabilities as outlined in OWASP (OWASP Foundation, 2024). It not only supports multiple vulnerabilities but also multiple users and provides real-time feedback and guidance. The graphical user interface is depicted in Figure 1. An adaptive, beginner-friendly training and testing workflow. Validation through structured test cases across five vulnerability classes. A user evaluation highlighting perceived usability and learning value.

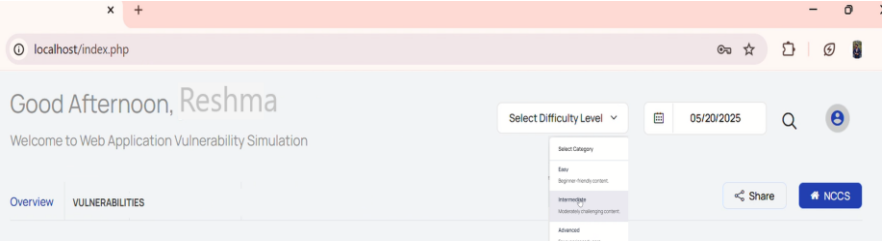

Figure 1: Adaptive dashboard with difficulty selection.

Literature Review

Web applications have become an essential part of daily life, enabling services such as online banking, online shopping, and social networking. However, because of the increasing number of users and transactions, web application security has become an extreme issue. One of the most vital steps in identifying and addressing vulnerabilities that might result in cyberattacks is testing the security of online apps. Improving software security is a priority of the non-profit Organization, OWASP. They offer free and open resources, tools, and standards to help developers, agencies, and users improve the safety of their software products. The OWASP top 10 list of web application vulnerabilities is considered one of its biggest contributions. A group of international security experts updates this list, which lists the most important security threats that online applications are now facing, every 3 to 4 years. Therefore, developers, designers, managers, and corporations need to take preventative or mitigating measures; the list aims to tell them about these vulnerabilities, such as Injection, damaged Authentication and authorization control, pass-site Scripting (XSS), and others, which are inside the list of the OWASP top 10.

Web applications are becoming a major target for cyberattacks because of the rapid growth of online services. Making sure the security of these programs is updated since they maintain sensitive statistics and perform important tasks. Recent research has focused on developing simulation environments that are more adaptable and intuitive. An intuitive simulation environment reduces the complexity related to employing security testing tools by allowing the testers to work without training and extensive technical knowledge. These tools' user interfaces enable a wider range of users, from novice developers to

experienced penetration testers, to access security testing (Veksler et al., 2018; Kavak et al., 2021). In web application security, vulnerable-by-design environments are widely used to teach common weaknesses, particularly those highlighted by OWASP Top 10 (OWASP Foundation, 2024). However, prior research and tool surveys indicate gaps in accessibility, guidance, and alignment with evolving practice (Mayhew et al., 2020).

There exist several learning platforms for web application security, e.g., OWASP WebGoat, Mutillidae II, Juice Shop, Security Shepherd. They are different from architecture point of view as well as their graphical user interface and guidance level for users (Druin, 2021). Numerous cyber security disciplines, including network protection, cyber-physical systems, social engineering, and web application security, have made use of simulation environments. The most effective assets for hands-on activities on vulnerabilities and exploiting attacks are insecure web applications like Webgoat, Juice Shop, BWAPP, DVWA, etc. There's a lack of literature on simulation environments designed completely for web application security testing (Singh et al., 2020, Aljebry et al., 2021; Luan, 2024; Chaudhary et al., 2023; Younas et al., 2024). (Wang et al., 2018) proposed a solution to secure physiological information based on the encryption algorithm in Body Area Network (BANs). However, this entire algorithm of encryption is based and built on the combine chaos to secure psychological information. To generate two sub matrices, Kent mapping approach and logistic chaotic model are used.

Table 1: Comparison between Cryptographic Algorithms of DES, AES, TDES, RC4

Algorithm	Key Size	Block Size	Round	Structure	Flexible
DES	64 bits	64 bits	16	Feistel	No
DH	Variable	-	-	Public Key Algorithm	Yes
E-DES	1024 bits	128 bits	16	Feistel	-
RSA	1024 to 4096	128 bits	1	Public Key Algorithm	No
T-DES	112 or 168	64 bits	48	Feistel	Yes
ECC	More than symmetric and variable	Variable	1	Public Key Algorithm	Yes
EEE	1024 bits	-	-	Public Key Algorithm	Yes
RC4	Variable	40-2048	256	Feistel Stream	Yes
RC2	8,128,64 by	64 bits	16	Feistel	-

(Alloghani et al., 2019) Another study combined an inspection recap of Homomorphic Encryption via various research papers. (Lu et al., 2018) proposed an encryption algorithm that consists of chaotic sequences and Polarization codes that have uniqueness in correlation to each other because the chaotic sequences are in frozen bits of polar codes. (LI et al., 2020) , formulated the algorithm under the mechanism of Two-Dimensional Lorenz and logistics for image encryption.

Security Techniques and Cloud Computing

Various security vulnerabilities are exposed due to cloud computing's decentralized nature (Faiz et al., 2022) explained that homomorphic encryption offers a solution, as shown in Figure 2. In the cloud resource-driven environment, trust mechanisms are deemed crucial. Challenges in cloud security encompass usage risk, semantic gaps, data handling, trust building, and risk of data-driven attacks.

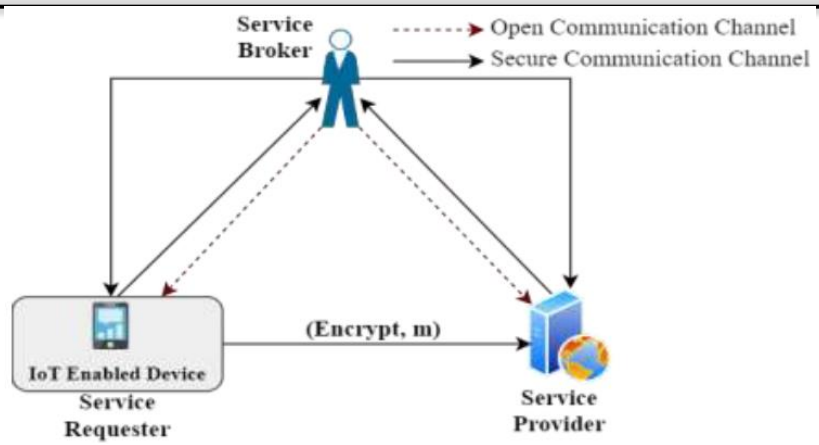

Figure 2: Homographic Encryption Technique for security in Cloud

The productivity of crypt-steganography schemes in Cloud infrastructure. (Rahman et al., 2018) and (Kouchay., 2018) used to exaggerate the infrastructure for the security of Data stored in the cloud by the quantitative methodology.

Steganalysis and Stenography in Cloud Computing

With the rapid growth of cloud-based data exchange, advanced techniques such as steganography and steganalysis have emerged as promising approaches for enhancing data security. In this context, (Prasad et al. 2022) proposed a deep learning-driven framework that integrates neural networks and steganography to strengthen security within cloud environments. Unlike traditional methods such as watermarking and cryptography, steganography enables covert embedding of sensitive information within digital media, particularly images, making detection significantly more challenging. The findings of the study demonstrate that the proposed approach achieves high performance in terms of accurate data embedding, mapping, and recovery. Moreover, the framework shows strong potential for further enhancement by incorporating error-correction mechanisms to improve robustness and reliability. In a related study, (Sharath et al. 2019) explored a secure and coherent platform for multimedia communication and transformation in cloud computing, highlighting the importance of integrating advanced security mechanisms for protecting data in dynamic cloud environments. (Kurt et al., 2018; Asif et al., 2025) explained the Cyber breaches and Cyber – offenses against the Smart Grid. As per intention, with effective responses to the stealthy attacks , the assessment showed better solution. Consequently, (Sloan, T., & Hernandez-Castro, J., 2018) dissented on the security concern and sensitivity effectiveness of the OnePuff Steganography approach for the PDF files. For small data manipulation it might be useful.

Cloud Computing Security at different Layers

Several recent investigations have shown that non-linear behaviors, the very dynamicity of traffic, and the need to manage high-dimensional feature sets are among the features of DoS attack detection in the PaaS environment. The following solution is proposed by Vallabapurapu et al. (2025) to overcome these challenges: RNN and the Oppositional Crow Search Algorithm (OCSA) are used to select the required features to allow for attack detection and classification, as well as filtering of malicious traffic while preserving the required data. In addition, LW-ABE is applied to preserve data security and integrity in cloud environments. The use of AI-based solutions significantly increases the reliability of the system. The evaluation includes the evidence insertion, response time, communication overhead, key generation time, evidence verification time,

overall alteration time rate, and decryption time. To manage cyber risk effectively, the Cyber Framework is followed as guidelines. (Xu, S. et al., 2020) focused on the challenges relevant to sharing of data in environments like cloud fog computing, where data must be kept secret, accessible e to authorized recipients, and identifiable by ratified resources. Cloud fog computing is a paradigm that extends the capabilities of cloud computing by utilizing edge devices (fog nodes) to provide various on-demand data services.

Methodology

The research methodology for this work is depicted in Figure 2. It starts with a literature review where the mapping of the implemented vulnerabilities has been made with the OWASP Top 10 list, as depicted in Table 1. Also, a detailed comparison is made with the existing tools which is depicted in Table 2. The approach is organized into four key phases that ensure both analytical rigor and practical applicability.

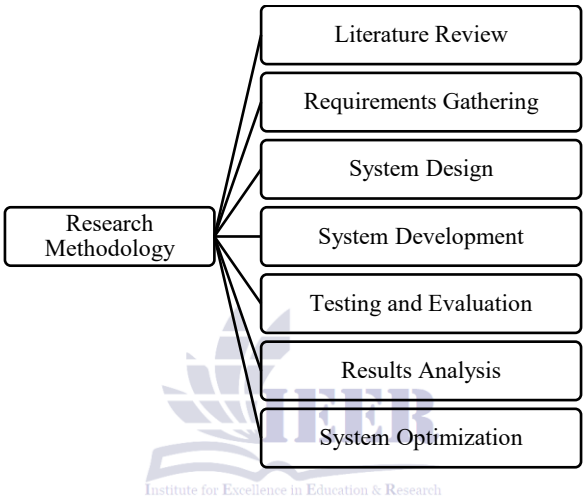

Figure 2: Proposed Research Methodology

Initially, the study focuses on recognizing the current state of web application security testing tools. A comprehensive survey of widely used vulnerable-by-design platforms such as WebGoat, DVWA, and OWASP Juice Shop is conducted to evaluate their capabilities in terms of vulnerability coverage, usability, adaptability, and feedback mechanisms. This analysis establishes a baseline understanding of existing solutions and highlights their role in security training and testing, as also reflected in the comparative analysis presented in the study. To determine the shortcomings and gaps in current simulation environments. The evaluation reveals several limitations, including static scenario design, lack of adaptive learning mechanisms, limited user guidance, and insufficient support for multi-level users. Furthermore, many existing tools fail to reflect evolving threat landscapes and do not provide real-time feedback or structured learning progression. These identified gaps form the foundation for defining the design requirements of the proposed system.

The third phase entails analyzing the most common techniques employed in the attacks as well as the vulnerabilities in modern web applications. The process entails matching the popular vulnerabilities like SQLi, XSS, CSRF, and IDORs with standardized vulnerability assessment frameworks such as OWASP top 10 to confirm that the simulation environment focuses on vulnerabilities that have significant impacts, as well as those used in common attacks against web applications. This step will be very useful in informing us about the development of vulnerability modules and the payloads employed in the system. Finally, the strategies to employ the potential users concerning web application

security issues were determined. With the aid of information from the preceding phases, an adaptive learning framework that allows for varying levels of difficulties (easy, moderate, and difficult), as well as provision of feedback, has been developed. The system offers learning through scenarios, adjustments in levels of difficulty and immediate guidance to mitigation. The effectiveness of these techniques has been confirmed through functional testing and user evaluation.

Table 2: Comparison with Existing Tools (Juice Shop, WebGoat, DVWA)

Feature	Proposed Tool	OWASP Juice Shop	WebGoat	DVWA
Vulnerabilities Covered	SQLi, XSS, CSRF, IDOR,	SQLi, XSS, CSRF, IDOR, XXE	SQLi, XSS, CSRF, XXE	experts only
Supports Multiple User Levels	Yes (Beginner, Intermediate, Advanced)	Uniform difficulty	Static scenarios	Complex for beginners
User- Friendly Interface	Highly intuitive with responsive design	Good interface	Outdated UI	None
Gamification Elements	Progress tracking, scoring, achievements	Challenges and Achievements	Minimal gamification	Technical logs only
Reporting & Analytics	Mitigation Strategies with detailed reports	Basic Feedback	Very basic	Not primality educational
Educational Use	Designed for progressive Learning	Used in academic settings	Suitable but outdated	Not primarily educational
Up-to-Date Content	Regular Planned updates	Actively maintained	Infrequent updates	Updated regularly

Adaptive Secure Key Pool Generation

When the adaptive simulation is taking place, the control parameter is dynamically chosen from a repository of parameters for each vulnerability scenario being simulated. As with choosing a cryptographic key, the selected parameter must meet certain validity criteria to ensure the simulation remains consistent and correct. The selected parameter r, r must be admissible in the operation domain. To support this, a generation mechanism is proposed to construct a shared parameter repository accessible to the simulation engine across different vulnerability modules. This repository is structured as an ordered set of parameter tuples of the form. $[v^{-1}k]$, where $k \in \{1,2,\dots,H\}$, and H denotes the repository size (with $H < G$, where G represents the global parameter space. Initialize simulation environment and load vulnerability modules corresponding to *Vulnerability_Set* where Vulnerability_Set ∈ {SQLi, XSS, CSRF, IDOR}. For algorithm 1, this parameter pool will be used during step 3 (Dynamic scenario creation), where suitable parameter combinations are chosen to create different attack scenarios. This creates variation, repetition, and control within the scenario generation process, all while ensuring structural consistency for each of the vulnerability types.

Algorithm 1 (v_k, v_k^{-1}) generation

1: Set difficulty parameters based on *User_Level*.
2: **procedure** $(v_{k,}\ v_k^{-1}) \in Vulnerability\_Set$: GENERATION ($ML_d$, k, β, H)
3: $s \leftarrow RC4(ML_d, k, \beta, H)$
4: $v_k^{-1} \leftarrow 0$
5: $j \leftarrow 0$
6: **while** $(v_k)^{-1} = 0$ **do**
7: $u(k,j) \leftarrow s_{k+j}$
8: $product \leftarrow 1+\theta \times m' \times u(k,j)$
9: $i \leftarrow 2$
10: **while** $(i \leq N\text{-}1)$ **do**
11: **if** $mod\ (product, i) = 0$ **then**
12: $v_k^{-1} \leftarrow i$ **break**
13: **else**
14: $i=i+1$
15: **endif**
16: **end while**
17: $j = j+1$
18: **end while**
19: $v_k \leftarrow mod\left(\frac{product}{v_k^{-1}}, \theta \times m'\right)$
20: **return** $(v_{k,}\ v_k^{-1})$
21: **end procedure**

In the pseudocode of Algorithm 1, the two end hosts generate the secret key. v_k for each k iteration. ML_d generates a secret sequence s of length β, where β is a secret integer here. From this sequence, a mapping function $u(k,j)$is used to select valid and admissible parameters v_k from a constrained domain.

Architecture of the Simulation Environment

A flexible and scalable architecture has been built on which the adaptive simulation environment for testing the web applications' security is constructed to overcome the key challenges in web application security testing, including evolving threats, usability to non-security experts, and integration with development workflows. It gives a detailed description of the system's architecture, design principles, and essential elements. It explains how the environment's modules cooperate and facilitate achieving the goal of adaptive security testing.

Adaptive to the different user levels, the design and interaction of the frontend is a user interface (UI). Processing user inputs and creating the attack environment is the function of the middleware. The structure of the backend database that hosts the vulnerability data includes attack payloads, application configurations, and testing logs. The attacker attack process is done through which the system can simulate attacks such as SQL Injection, XSS, and Broken Access Control. Part of this is designed to include visual aids such as diagrams and flowcharts to make the architecture a bit clearer to participants. Figure 3 illustrates the main components of the architecture, i.e., Frontend, Middleware, and Backend.

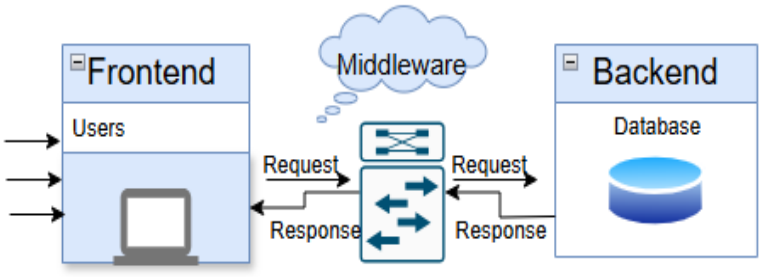

Figure 3: System design & architecture

Statistical Analysis

To carry out this research work, we developed a state-of-the-art tool for simulating the OWASP top 10 vulnerabilities. Our simulation tool has been built based on the architecture described in the previous section. It has several extra features and outperforms all of the existing tools for learning web application vulnerabilities. An evaluation of its performance and security is done using several steps. First, there is a measurement of the execution time of the algorithm and its security features compared to other algorithms, and the results show that it is more efficient without any extra storage cost. Secondly, there is a test of statistical attacks by analyzing the security of the algorithm statistically in the vector sets Ti produced after processing.

The properties of the distribution of the main web weaknesses, namely SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and Insecure Direct Object Reference (IDOR), are studied with a Gaussian modeling framework. In the case of the given distribution, common attacks, for example, SQLi and XSS, tend to cluster around the mean, whereas less frequent attacks, such as CSRF and IDOR, lie at the tails of the distribution. Such a

A biased but normal distribution shows that attack behavior is not uniform, hence supporting weighted simulation methods. Overall, the evaluation framework integrates performance testing, statistical validation, and probabilistic modeling to ensure that the proposed approach achieves efficiency, robustness, and a realistic representation of vulnerability behavior.

Distribution Test

The real-world vulnerability distribution follows a skewed Gaussian curve, while the simulation baseline (uniform) deviates significantly. This mismatch highlights that equal-probability simulations introduce distribution bias, leading to unrealistic training outcomes. Therefore, the proposed system uses adaptive weighting aligned with real-world statistics. To resist statistical attacks, a strong cipher algorithm should not reveal any information about its cipher distribution. Thus, encrypted data distribution should be close to uniform. A Gaussian plain-text distribution with a mean value equal to 128 and standard deviation equal to 16 is taken in Fig. 4a, and the distribution of the obtained set of ciphertexts is illustrated in Fig. 4b

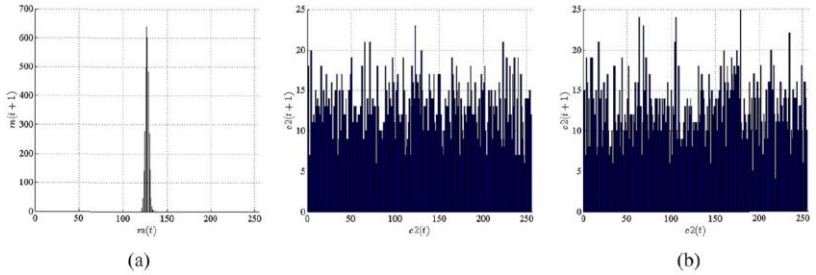

Figure 4: Distribution Analysis (a) Original Message, (b) Enhanced Output (T1)

Recurrent Test

A simulation model for the security assessment framework must display a significant amount of randomness and variance in the creation of attacks. The recurrence test helps determine this by examining the progression of the level of diversity in the generated scenarios and determining correlations within the sequence. Figure 5 shows the recurrence of adjacent vulnerabilities differences for SQLi-XSS, XSS-CSRF, and CSRF-IDOR. The absolute percentage measures show the difference between the vulnerability types that lie next to each other. The biggest difference, which is around 14%, is seen between XSS and CSRF. This shows that there is a huge shift from frequent vulnerabilities to uncommon vulnerabilities.

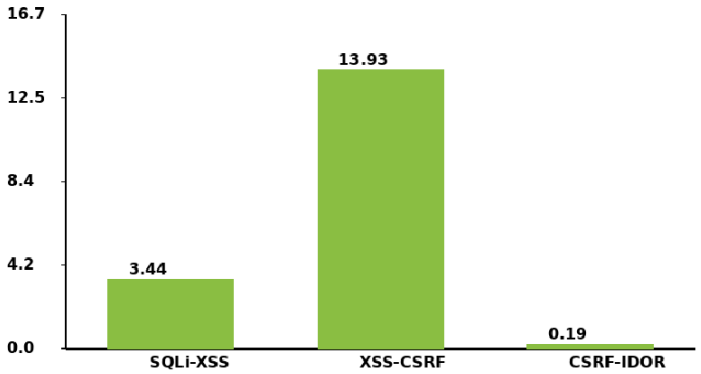

Figure 5: Adjacent vulnerabilities absolute differences at (~14%) between XSS and CSRF

Let $m_i = \{m(i,1), m(i,2), m(i,3), .., m(i,n)\}$ To represent the sequence of scenario parameters, a delayed representation is defined as $m_i(t) = \{m(i,t),\ m(i,2t), .., m(i,xt)\}$ where t should be greater than or equal to 1.

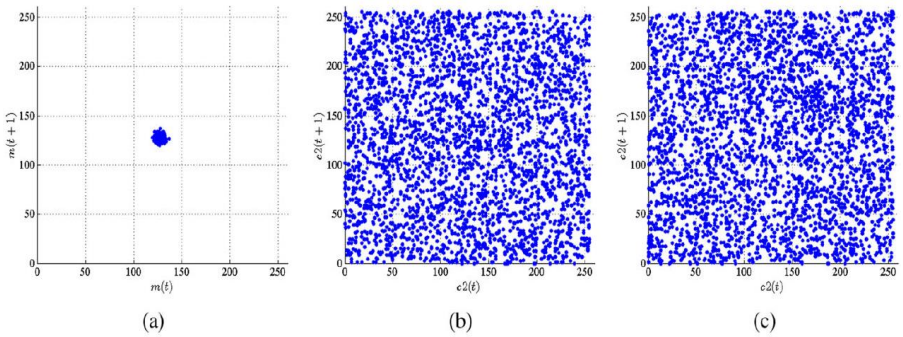

Figure 6: Recurrent test (a) correlation and structure (b) and (c) proposed adaptive scheme correlation for consecutive values

Figure 6 shows the correlation between successive delayed sequences. $m_i(t)$and $m_i(t+1)$ or both baseline (non-adaptive) and proposed adaptive simulation outputs. From Fig. 5(a), it is evident that the base scenarios have strong correlation and structure, implying a lack of variability and predictability in the process of generating scenarios. However, from Figs. 5(b) and 5(c), which represent the proposed adaptive simulation approach, there is a minimal degree of correlation between consecutive values. There is a significant presence of randomness and variability in the created scenarios. This shows that the proposed simulation environment has been able to generate a suitable level of randomness and variability. This makes the testing process more realistic, and the users are better able to learn from this experience.

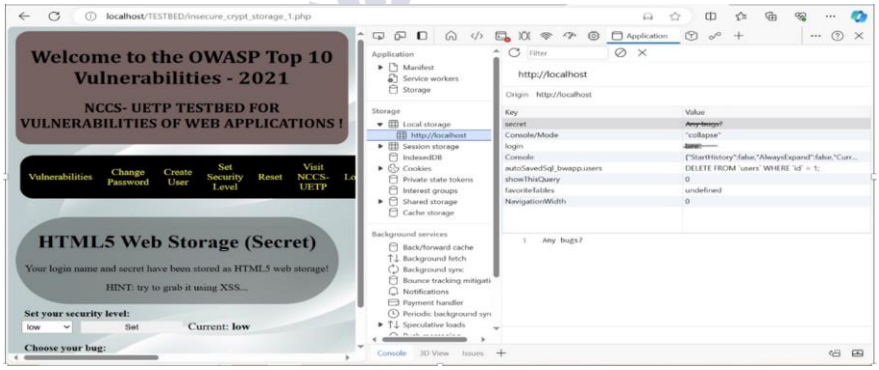

Figure 7. Sensitive data exposure scenario (storage/secret illustration)

Figure 7 shows a storage-related sensitive data exposure scenario. The tool includes all of the Top 10 vulnerabilities of OWASP and offers three levels of difficulty for various categories of users, e.g., Easy, Intermediate, and Advanced. There exist more than 100 practical labs to demonstrate each vulnerability in detail to the user. Figure 4 visualizes user performance perceptions (ease of use, flexibility, and scalability) compared to common tools. Participants reported positive perceptions regarding interactivity and structured learning. In the study, 78% highlighted the value of hands-on exploration, and more than 75% reported improved organization and learning performance after guided use.

Assessment of Performance

The performance of the simulation tool has been analyzed for factors like usability, schooling for user feedback, responsiveness, and balance.

Usability

The system's user interface (UI) was evaluated for accessibility and value. According to user feedback, the device's layout is easy. Novices can start assessing vulnerabilities without knowing it. With the usage of real international situations, customers have been able to effectively attack vulnerabilities, including SQL Injection and XSS. The accuracy of vulnerability exploitation and achievement rates has been used to gauge the simulation's effectiveness, and the findings validated that the machine achieved success in simulating real attack strategies.

Impact on the performance of schooling for user feedback evaluated the tool's capability to enhance online security talents. In keeping with the findings, users' self-assurance to understand and deal with vulnerabilities in actual international applications improved after the usage of the system. Indicating the tool's efficacy as a training aid, more than 75% of users stated they felt much more organized, as shown in Table 3.

Table 3: User Evaluation Summary

Metric (User Evaluation)	Key Finding
Interactive learning experience	78% of participants reported that the tool enabled hands-on exploration of vulnerabilities.
Perceived improvement in organization/learning	More than 75% reported improved organization and learning performance after using guided flow.
Usability (qualitative)	Most participants described the GUI as user-friendly and distraction-free for beginners.
Requested enhancements	Support for advanced vulnerabilities (e.g., RCE/SSRF), richer reporting, and cross-platform optimization.

Performance Benchmarks

The tool's responsiveness and balance in trying out conditions have been taken into consideration while comparing its performance. Even in conditions with heavy demand, the system's response time remains powerful, and it may mirror weaknesses in actual time. This demonstrates the scalability of the simulation environment for further improvements. Table 4 summarizes representative test cases executed in the simulation environment. The observed outcomes demonstrate that the tool reproduces intended vulnerability effects and presents corresponding mitigation guidance.

Table 4: Summary of Representative Test Cases

Test Case	Payload / Action	Expected Outcome	Observed Outcome
TC-1 (SQLi – login bypass)	admin' OR '1'='1	Authentication bypass / unauthorized access	Bypass achieved; improper input handling observed
TC-2 (XSS – reflected)	<script>alert('XSS')</script>	Script executes in browser context	Alert executed; XSS confirmed
TC-3 (Broken Access Control)	Direct access to the/admin endpoint	Unauthorized user denied	Access is possible under weak authorization checks
TC-4 (Sensitive Data Exposure)	Observe data transmitted/stored in plaintext	Sensitive data protected (encrypted/hashed)	Plaintext exposure simulated; mitigation guidance shown

TC-5 (IDOR)	Manipulate object identifier in URL/parameter	Access restricted to the owner	Unauthorized record access simulated; control weakness highlighted

Discussion

The summarized results show that the software successfully replicated several online available web-based application vulnerabilities like Cross-Site Request Forgery (CSRF), SQL Injection, and cross-site scripting (XSS), and others. Maximum participants deemed the tool's UI to be user-friendly, needing little or no previous safety testing understanding. The software's adaptability made it easier to modify the degree of the problem, making it appropriate for each novice and professional college student.

Several tests were performed for the quantitative evaluation, while user surveys and interviews were used for the qualitative assessment. The findings indicated that the simulation device became more demanding than conventional static training materials. The difference shown in Figure 8 between the real-world distribution of vulnerabilities (blue) and the baseline distribution of simulations (orange), which assumes an equal likelihood for each (25%), emphasizes the importance of adaptive simulation. Excessive emphasis on rare attacks may be misleading, while a lack of emphasis on more prevalent attacks like SQLi and XSS does not reflect reality.

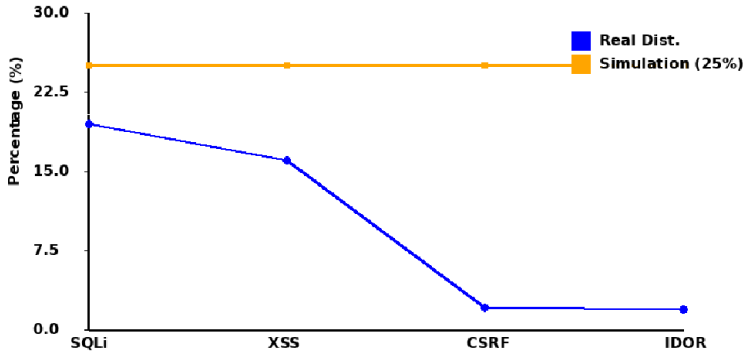

Figure 5: Comparison of the real distribution vs. the proposed simulation distribution.

Conclusion

This paper presented the design and development of an intuitive, adaptive simulation environment that aims to educate users about web application security vulnerabilities. This indigenous system is tailored to cater to novice, intermediate, and advanced users with specialized training modules that can be used by virtually any user who wants to deepen their knowledge of web application security threats through hands-on experience. The overall output of this research consists of a customizable simulation environment for testing web application vulnerabilities that is customizable. The Realistic Attack simulation, User Experience & Learning Ability, and finally promoting security awareness and skills building. In the future, research can be performed on adding more vulnerabilities in the developed framework, optimizing the performance of the framework by including machine learning techniques, designing more user-friendly analytics and reporting modules, and providing automated methods for fixing the discovered vulnerabilities.

REFERENCES

Abdulghaffar, K., Elmrabit, N., & Yousefi, M. (2023). Enhancing web application security through automated penetration testing with multiple vulnerability scanners. *Computers*, *12*(11), 235.

Wang, W., Si, M., Pang, Y., Ran, P., Wang, H., Jiang, X., ... & Jeon, G. (2018). An encryption algorithm based on combined chaos in body area networks. Computers & Electrical Engineering, 65, 282-291

Veksler, V. D., Buchler, N., Hoffman, B. E., Cassenti, D. N., Sample, C., & Sugrim, S. (2018). Simulations in cyber-security: A review of cognitive modeling of network attackers, defenders, and users. *Frontiers in Psychology*, *9*. https://doi.org/10.3389/fpsyg.2018.00691

Chai, X., Zheng, X., Gan, Z., Han, D., & Chen, Y. (2018). An image encryption algorithm based on a chaotic system and compressive sensing. Signal Processing, 148, 124-144.

Vallabapurapu, S., & Vankdothu, R. (2025). PaaS Platform Security Enhancement and DOS Attack Detection In Cloud Computing and Its Prevention. *Journal of Computational Analysis & Applications*, *34*(12).

Alloghani, M., Alani, M. M., Al-Jumeily, D., Baker, T., Mustafina, J., Hussain, A., & Aljaaf, A. J. (2019). A systematic review on the status and progress of homomorphic encryption technologies. Journal of Information Security and Applications, 48, 102362.

Lu, X., Lei, J., Li, W., Lai, K., & Pan, Z. (2018). Physical layer encryption algorithm based on polar codes and chaotic sequences. IEEE Access, 7, 4380-4390.

Li, T., Du, B., & Liang, X. (2020). Image encryption algorithm based on logistic and two-dimensional lorenz. Ieee Access, 8, 13792-13805.

Faiz, M., Fatima, N., Sandhu, R., Kaur, M., & Narayan, V. (2022). Improved Homomorphic Encryption for Security in Cloud using Particle Swarm Optimization. Journal of Pharmaceutical Negative Results, 4761-4771.

Singh, N., Meherhomji, V., & Chandavarkar, B. (2020). Automated versus manual approach of web application penetration testing. In *Proceedings of the International Conference on Computing, Communication and Networking Technologies* (pp. 1–7). IEEE.

Rahman, M. O., Hossen, M. K., Morsad, M. G., & Chandra, A. (2018). An approach for enhancing security of cloud data using cryptography and steganography with e-lsb encoding. IJCSNS, 18(9), 85.

Prasad, P. V. H., & Rao, K. G. (2022). A Security Approach using Steganography, Deep Learning and Conventional Neural Networks in a Cloud Environment.

Sharath, M. N., Rajesh, T. M., & Patil, M. (2019, July). Analysis of secure multimedia communication in cloud computing. In 2019, the 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT) (Vol. 1, pp. 136-144). IEEE.

Kurt, M. N., Yılmaz, Y., & Wang, X. (2018). Real-time detection of hybrid and stealthy cyber-attacks in smart grid. IEEE Transactions on Information Forensics and Security, 14(2), 498-513.

Asif, U., Behnam, R., & Shadmand, M. B. (2025, November). AI-Driven Real-Time Detection and Mitigation of Cyber-Intrusions in Grid-Forming Inverters. In 2025 IEEE PES Conference on Innovative Smart Grid Technologies-Middle East (ISGT Middle East) (pp. 1-6). IEEE.

Aljebry, A., Alqahtani, Y., & Sulaiman, N. (2021). Analyzing security testing tools for web applications. In *Proceedings of the International Conference on Innovative Computing and Communications* (pp. 411–419). Springer.

Luan, T. (2024). A comprehensive review of simulation technology: Development, methods, applications, challenges, and future trends. *International Journal of Emerging Technologies and*

Xu, S., Ning, J., Li, Y., Zhang, Y., Xu, G., Huang, X., & Deng, R. H. (2020). Match in my way: Fine-grained bilateral access control for secure cloud-fog computing. IEEE Transactions on Dependable and Secure Computing, 19(2), 1064-1077.

Chaudhary, P., Sharma, R., Rao, M., & Tiwari, M. (2023). Role of machine learning applications in enhancing cybersecurity effectiveness: An empirical study. *Rivista Italiana di Filosofia Analitica Junior, 14*(1), 405–411.

Mayhew, B., Baars, N., White, J., & Zubčević, R. (2020). *OWASP WebGoat*.

Druin, J. (2021). *OWASP Mutillidae II*.

Villegas-Ch, W., Govea, J., & Ortiz-Garces, I. (2024). Developing a cybersecurity training environment through the integration of OpenAI and AWS. *Applied Sciences, 14*(2), 679.

Younas, F., et al. (2024). An efficient artificial intelligence approach for early detection of cross-site scripting attacks. *Decision Analytics Journal, 11*, 100466.

Positive Technologies. (2022). *Threats and vulnerabilities in web applications 2020–2021*.

OWASP Foundation. (2024). *OWASP top ten*.