Web Application Penetration Testing with Artificial Intelligence: A Systematic Review

Gustavo Sánchez
Karlsruhe Institute of Technology
Eggenstein-Leopoldshafen, Germany
sanchez@kit.edu

Olakunle Olayinka
The University of Sheffield
Sheffield, UK
o.olayinka@sheffield.ac.uk

Aryan Pasikhani
The University of Sheffield
Sheffield, UK
aryan.pasikhani@sheffield.ac.uk

***Abstract*—Penetration testing is an intricate activity, yet vital for the security of web applications and the protection of user data. Due to its time-consuming nature, recent developments have emphasized the use of artificial intelligence to enhance efficiency, shorten testing times, and substantially improve penetration testing results. By combining artificial intelligence with conventional penetration testing techniques, researchers aim to improve the processes, providing organizations with the means to create stronger web applications. This paper presents a thorough review of research conducted between 2013 and 2024 on the application of artificial intelligence in web application penetration testing. We highlight advancements and challenges in employing learning-based methods to enhance penetration testing, providing a comprehensive overview of the current state and future directions in the field. Our results show that leveraging artificial intelligence has proven to be more efficient than traditional approaches, but they still face significant challenges.**

***Index Terms*—Machine learning, security, web applications.**

I. Introduction

Cybercriminals are increasingly targeting web applications to steal data and this activity is motivated by varying goals, including social, political, or financial gains [1]. The early 2020s, specifically through the 2020 Verizon Data Breach Investigations Report [71], revealed that 43% of data breaches were associated with web application vulnerabilities, a figure that has seen a significant year-over-year increase. The onset of the COVID-19 pandemic further accelerated the shift towards remote work, expanding the use of online communication and subsequently the need for enhanced security across an increased number of web endpoints and applications. In the third quarter of 2023, Cisco Talos Intelligence Group [14] reported observing a significant rise in web application threats, which constituted 30% of their total engagements. Nevertheless, the topic of web application security has been a focal point of research in industry and academia for a considerable length of time. Despite widespread adoption of penetration testing (also known as *pentesting*) by companies prior to application deployment, the depth and thoroughness of these tests often remain in question. Integrating Artificial Intelligence (AI) with traditional pentesting methods proposes an advancement in this area, offering intelligent tests that can refine and concentrate efforts more effectively.

This paper provides a comprehensive review of over a decade's worth of scientific research on AI-based pentesting for web applications. In summary, our contributions include identifying and examining the state of the art in this area, discussing prevailing trends and challenges, and suggesting directions for future research. A major contribution of our work is the exploration of Artificial Intelligence (AI) methods tailored to specific stages of penetration testing, which, to the best of our knowledge, has not been previously undertaken.

The rest of the paper is structured as follows: Section 2 presents the necessary background information and explains how our survey compares to similar work. Section 3 describes our review methodology and initial findings. Section 4 further provides and classifies insights extracted from the reviewed papers. Section 5 answers the research questions and states future research directions. We end with a conclusion in Section 6.

II. Background and Related Work

In this section we provide a description of the concepts that will be discussed throughout the review, and compare our work with similar surveys.

A. Penetration Testing.

Penetration Testing is a critical process for identifying and documenting unseen vulnerabilities in a system. This process is typically conducted by a cybersecurity expert, often referred to as a penetration tester or *pentester* for short, aiming to enhance the system's security. For companies aiming for robust security, it's essential to conduct these tests and address any vulnerabilities early in the Software Development Life Cycle. Penetration testing can be categorized into three types based on the level of information the tester has about the system before starting the exercise [58]: *Black-box* testing simulates an external cyberattack without prior system knowledge, focusing on real-world scenarios without source code access. *White-box* testing offers an in-depth review with complete system details, enabling a comprehensive security examination from an insider's perspective. *Grey-box* testing provides a middle ground, with partial system knowledge, reflecting situations where an attacker has limited system information. These methods allow cybersecurity teams to customize their defense strategies effectively.

TABLE I
General comparison of literature reviews on penetration testing with AI.

Ref	Year	Focus	Papers	Period
[9]	2023	Software Vulnerability Prediction	77	2007-2022
[56]	2023	General Pentesting	undefined	undefined
[25]	2023	Software Vulnerability Detection	67	2011-2022
[40]	2019	General Pentesting	31	2002-2017
Our survey	2024	Pentesting Web Applications	49	2013-2024

B. Types of Test

Software pentesting focuses on evaluating standalone software applications, such as desktop or mobile apps, by analyzing their code, functionality, and security mechanisms to identify vulnerabilities like buffer overflows and injection flaws.

Web app pentesting (the main topic of this paper) targets web applications accessible via browsers, examining front-end, back-end, and APIs for common web vulnerabilities like SQL injection and cross-site scripting (XSS).

In contrast, general pentesting encompasses a broader scope [56], assessing an organization's entire security posture by testing network infrastructure, operating systems, servers, and more, to identify and exploit potential weaknesses across various systems and platforms.

Vulnerability prediction [9] involves forecasting potential security weaknesses in software or systems before they are discovered or exploited by malicious actors. This proactive approach uses techniques like machine learning, historical data analysis, and pattern recognition to identify areas that are likely to develop vulnerabilities in the future, allowing organizations to prioritize resources and implement preventive measures.

On the other hand, vulnerability detection [25] is the process of identifying existing security flaws in systems or software through techniques such as scanning, penetration testing, and code analysis. This reactive approach focuses on uncovering and addressing vulnerabilities that are already present, ensuring that they are mitigated before they can be exploited by attackers.

Static code analysis for vulnerability detection, including techniques like taint analysis, has been extensively researched for decades [42]. Static analysis involves examining the source code of an application without executing it, aiming to identify potential vulnerabilities, coding errors, and security issues early in the development cycle. This approach can detect issues such as buffer overflows, injection flaws, and insecure coding practices by analyzing code structure, data flow, and control flow. In contrast, dynamic analysis is essential in current pentesting, especially when testers lack access to source code due to various constraints like intellectual property. Dynamic analysis simulates real-user interaction by testing applications in operation, thereby identifying runtime vulnerabilities that may not be evident through static analysis alone. Hybrid analysis combines both static and dynamic methods to enhance vulnerability detection, offering a comprehensive approach to security assessments.

Note: The general idea is that, the combination of traditional web app pentesting techniques with AI components allows for the optimisation of the processes, eventually improving web application security. In this paper, the focal point is the use of AI for offensive security purposes.

C. Related Work

Penetration testing using AI is an interesting and evolving research area with researchers exploring various aspects of the research domain [31] [49] [43]. Although other studies have presented literature reviews on AI and penetration testing, there has been no study that has comprehensively explored web application penetration testing using AI. Other recent literature reviews highlight the existence of work in the topic of automating penetration learning with learning-based components. We show an overview in Table I for high-level comparison.

In [9], authors put the focus on software vulnerability prediction, and provide guidelines for researchers to increase the productivity of their models. Authors in [56] briefly cover the usage of a decision tree algorithm for enhancing the Metasploit framework [69] and RL to discover exploits in target machines [20]. That review also mentions publications focusing on learning-based malware detection in the Android operating system [24], [4], topic that does not qualify as pentesting (rather a defensive, reactive approach to security analysis). In contrast to our work, the authors do not cover pentesting of web applications comprehensively, and they analyze theses (i.e., [20], [13]) which are not peer reviewed by the community. Authors in [25] aim to identify trends, characteristics of datasets, learning-based models, vulnerabilities covered and futures challenges of automated offensive security with a focus in general software. Out of the 77 papers reviewed, according to the titles, only 11 papers explicitly mention web applications or directly related topics like SQL injection and Cross-Site Scripting (XSS), which are specific to web security. They provide software vendors and other stakeholders with useful insights provided to inform their decisions about software development, procurement, and risk management. An slightly older review and meta-analysis in the topic of AI in pentesting is [40], where the main topic of the reviewed work is network security.

Regarding papers reviewed, in related work, even though they include a number of relevant papers, our review is the most comprehensive in terms of total number of papers about pentesting web applications with AI components. As a distinct contribution, our review is the first to categorize the use of AI methods according to the different stages of penetration testing. This is important in order to identify specific research gaps, motivating further research in the areas less investigated (i.e., AI applied to information gathering and post-exploitation). Furthermore, we address the availability of open source code for the first time in this domain.

As a significant contribution, our review addresses the scarcity of recent literature analyses. This is particularly important because our observations indicate that 2022 and 2023 have been the most productive years to date in the field under study. Additionally, we have for the first time considered papers from incipient research directions, such as the use of Large Language Models (LLMs) for pentesting web applications or adversarial attacks against learning-based defenses.

III. Review Methodology and Results

Our objective is to present insights in a way that allows for reaching conclusions, having a clearer picture of the topic at hand.

A. *Penetration Testing Stages in Web Applications.*

We use the taxonomy provided in [11] to assign a class label to each published paper, according to the contribution towards a specific pentesting stage. *Information Gathering.* In this phase the attacker finds information that can be used in subsequent phases, e.g., domain names; *Scanning and Enumeration.* In this phase the attacker detects services exposed by the target web application; The attacker also enumerates running services (e.g., directories [6]). The goal here is to detect versions of running services and look for potential vulnerabilities; *Exploit.* When the attacker has detected vulnerabilities in the system, she tries to exploit them and get inside the target; *Post-Exploitation.* The attacker tries to obtain higher privileges and persistence inside hacked systems, and performs lateral movement activities to gain access to other internal systems. These stages are mapped to papers in Table II.

B. *Strategy*

The process starts by formulating the review questions using the PICO approach [46]: **RQ1:** What AI methodologies are predominantly used in web applications penetration testing, and for what specific purposes?; **RQ2:** How do AI-driven web application pentesting tools compare in effectiveness and efficiency to traditional methods?; **RQ3:** What are the recognized limitations and challenges for AI-driven web applications pentesting tools as identified in the literature?.

Our study focuses on cybersecurity research with an **offensive approach**, emphasizing exploiting vulnerabilities over defense. The search terms combined to query literature databases are: *Machine Learning*, *Artificial Intelligence*, *Vulnerability Discovery, Pentesting* and *Web Applications*.
We initially retrieved 1203 papers from various literature resources (Google Scholar, IEEE Xplore, ACM Digital Library, Science Direct, and Scopus), which included duplicates. After deduplication, we reviewed the abstracts and applied inclusion and exclusion criteria:

- We include papers that utilize learning-based methods to analyze web application vulnerabilities, requiring empirical data analysis and peer-reviewed validation.
- Exclusions are made for papers not centered on web applications, unpublished works, non-English publications, and non-empirical studies like reviews or opinions, to concentrate on rigorous, empirical research within web application security.

Following deduplication and initial screening, 136 papers remained. We then conducted a full-text review of these papers, applying our criteria comprehensively. Ultimately, 49 papers qualified for inclusion in our review. The selection methodology is depicted in Appendix A.

C. *Review Results*

Now, we break down the results into insights.

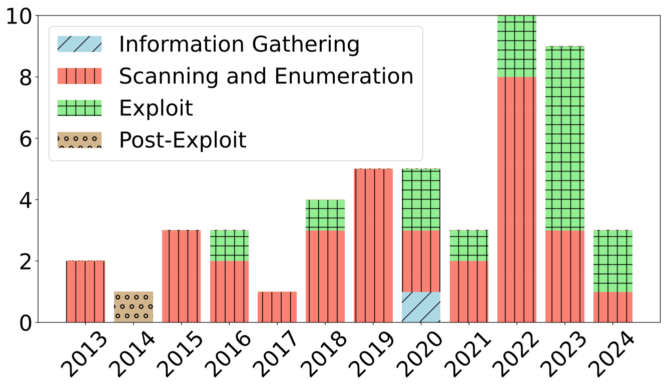


Fig. 1. Papers and pentesting stages investigated over the years.

Research Work Over the Years. Upon applying the exclusion and inclusion criteria, a total of 49 papers were retained for further analysis. Over the ten year period, there has been a steady interest in the topic as can be seen in Fig. 1. Between 2018 and 2021, there was a consistent interest in the topic with at least 4 papers published per year. In 2022, there were 10 papers published on web application pentesting while in 2023, there were 9 papers published. By august 2024, 3 relevant publications were published.
Publication Venues. From the papers reviewed, most of the papers were published in conferences. 29 papers were published in conferences with IEEE and ACM conferences being part of the most popular destinations for publishing. A total of 20 papers were published in journals.
Stages of Penetration Testing. As depicted in Fig. 1, most of the papers focused on scanning and enumeration. Papers in this area focused on vulnerability detection and prediction using a number of AI approaches. 14 papers focused on exploit. The areas of post-exploit and information gathering were not as researched as they only had one paper each. In Fig. 2, we have depicted what AI algorithms are more frequent in each stage, which will be further described in Section 4.

Year and Pestenting Stage. In Fig. 1 we can see that many of the publications in 2023 focused on the exploit phase in pentesting. Compared to previous years this shows a possible increasing interest in researching the application of AI approaches to actually exploit systems. In 2022, we see that there was a reasonable increase in the number of publications focused on scanning and enumeration compared to previous years (although in 2019 we had 5 papers). This shows a renewed interest in the area, as AI approaches become more mature and newer algorithms and models are developed.

Frequency and Type of Test. From our review of the articles, we had 17 papers that focused on static approaches, 25 that focused on dynamic approaches and 2 papers that focused on hybrid approaches. Many of the papers in 2022 focused on injection vulnerabilities detection using either static or dynamic approaches, with SQL injection and XSS being quite popular. It is no surprise that in our review we had more papers that focused on dynamic analysis, as this approach to analysis is more closely related to what actual penetration testers would encounter. It is worth noting that, the majority of papers that focused on the exploitation stage of pentesting formed part of the papers that made use of the dynamic analysis approach. Also, in our review, the 2 papers that focused on hybrid, used dynamic approaches to complement the static approaches, and were both focused on vulnerability prediction.

Occurrences of AI Subgroups. Now we discuss the frequency of different AI subgroups mentioned in the context of pentesting in web applications as found in our review. The AI subgroups listed are Reinforcement Learning (RL), Machine Learning (ML), Natural Language Processing (NLP), and Neural Networks (NNs). RL appears to be used the least, followed by NLP, with ML and NNs being discussed more frequently. The review reveals a strong preference for ML algorithms. There's a notable trend towards employing advanced NLP techniques, such as word embeddings and transformers, indicating interest towards understanding and exploiting textual data within web applications. RL is also gaining traction, suggesting a move towards more adaptive and dynamic testing approaches.

IV. From Review Results to Taxonomies

In this section, we extract insights from the reviewed papers with the objective of understanding the state of the art.

Algorithms. Delving deeper, we have explored the number of times each algorithm was used across the papers reviewed. ML techniques remain heavily favored, with Support Vector Machines (SVM) leading with 16 occurrences, followed closely by Random Forest (RF) at 15, and Logistic Regression (LR) with 12 appearances. Other frequently used supervised learning algorithms include Naive Bayes (NB) and Decision Trees (DT), appearing 11 and 7 times respectively, illustrating the diversity in model selection. K-Nearest Neighbor (KNN) also feature prominently, showing up 5 times. Deep learning techniques such as Long Short-Term Memory (LSTM) networks are noted 6 times, demonstrating their relevance in sequence analysis and natural language processing. While still less common compared to traditional ML methods, RL approaches like Deep Q-Learning (DQN) and Generative Adversarial Networks (GAN), each appearing twice, indicate an emerging interest in more complex model strategies.

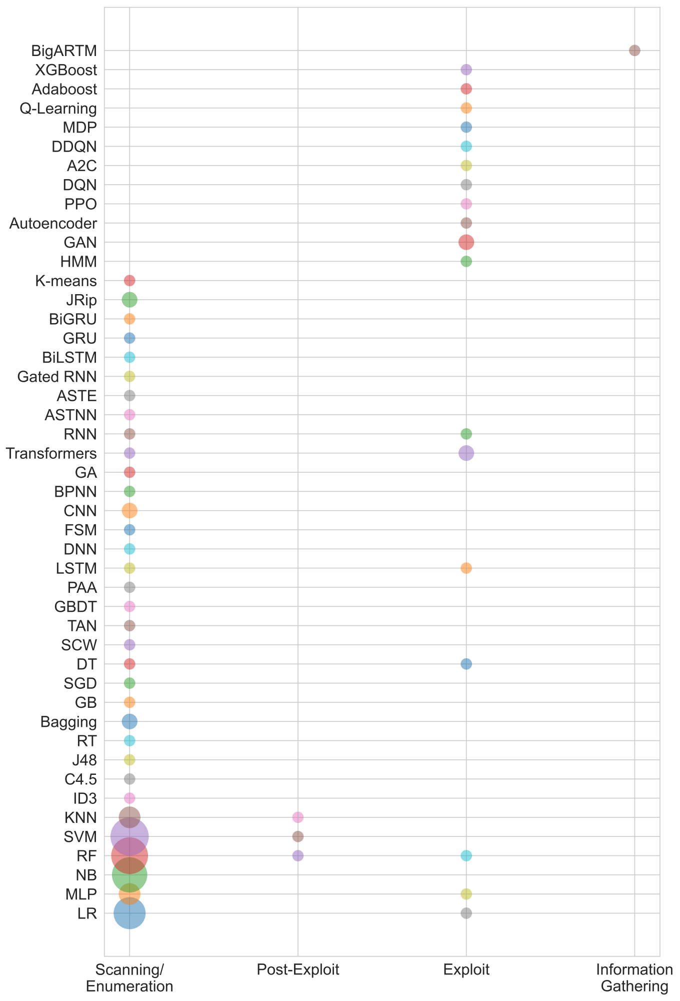


Fig. 2. Frequency of AI algorithms according to each pentesting stage. Larger circles indicate higher frequency.

Furthermore, in order to understand what problem they were trying to solve, in Fig 2 we map the frequency of AI algorithms to each pentesting stage. An acronym list with full algorithm names can be found in the Appendix B.

Shallow algorithms are applied across a wide range of tasks but are particularly prominent in the scanning and enumeration stage. They are effective for tasks that require quick and robust decisions based on clear feature distinctions. These methods are often used due to their simplicity, interpretability, and effectiveness in scenarios with less complex data structures.

Deep learning models are increasingly used in later stages of penetration testing such as exploit detection. These models excel in identifying complex patterns and sequences in data, which are crucial for detecting sophisticated multi-stage attacks.

Datasets. Supervised ML methods rely on annotated corpora. The quality of this data directly impacts the ability of the tool. Static analysis tools leverage source code gathered from online repositories, and manual labeling is usually required. Authors in [35] manually create input-output test case pairs to build their training dataset based on publicly available SQLi test cases, i.e., they mine repositories of fuzz testing or brute force tools from various projects[1]. Another variant is to use a synthetic test case generator. Authors in [22] used a publicly available repository[2] containing synthetic test cases written in PHP for both training and evaluation of XSS vulnerability prediction. The SARD dataset [62] (also known as SAMATE Reference Dataset) provides researchers, developers, and general end users with a set of artifacts with known security errors and fixes for them. The artifacts include designs, source code, binaries, etc., that is, elements from all phases of the software life-cycle. The samples include *synthetic* (i.e., created for testing), collected from production environments, and academic research. This dataset contains real, production software applications with known bugs and vulnerabilities, what allows developers to test their methods and end users to evaluate a tool in a realistic manner. The dataset intends to bring together a wide variety of vulnerabilities, languages, platforms, and compilers. The size of the applications ranges from small to large. The source codes are written in C, C++, Java, PHP and C#, containing over 150 different classes of weaknesses. We observe that PHP samples from SARD are used for both training ML models (e.g., with XSS vulnerable snippets [30], SQLi test cases [30], [18], etc.) and evaluating tools [44], [30], [41] in the context of web application vulnerabilities. Authors in [74] use *SARD-testsuite-103* to test their PHP code-auditing NN approach as it considers the four elements (data source, filtering function, dangerous function and environment) for vulnerability formation during the generation process; this dataset largely simulates the vulnerability situations in real projects. Another type of repository is Xssed[3], which claims to be the largest online archive of websites vulnerable to XSS, even though it seems to have ceased activity. Authors in [21] perform their experiments on 50 websites chosen from Xssed in chronological order. Authors in [73] use Xssed in combination with their own dataset to compare their automatic XSS attack vector generator based on RL against other solutions from literature. From our analysis, we observe that there is a scarcity of datasets available to researchers. Authors in [10] contribute to the research community by making their custom dataset available online[4]. Their dataset consists of 5828 HTTP requests from 60 popular websites of the Alexa ranking, including 939 sensitive requests. Another example is the released BCCC-SFU Adversarial SQL Injection Attack Dataset as found in [26], and a synthetic attack dataset in [12]. In [34], authors make attack grammars for three typical attacks; their trained models and datasets are also publicly available[5]. Authors in [12] trained a generative model based on attack labels and attack features; they reference a repository[6] of payloads to exploit XSS.

[1]https://tinyurl.com/wh94b8t

[2]https://github.com/stivalet/PHP-Vulnerability-test-suite

[3]http://www.xssed.com/

[4]https://github.com/alviser/mitch

Target Web Applications. New approaches are evaluated using common environments. DVWA[7] is an environment specially developed for testing web app security, as it includes a wide range of vulnerabilities on purpose. Authors in [41] run the Web Application Protection tool that implements their detection approach, being able to correct 6 files of DVWA version 1.0.7. Furthermore, authors in [73] attempt XSS vulnerability detection on DVWA. As the most adopted Content Management System (CMS) worldwide, WordPress has a wide variety of plugins available. These plugins constitute interesting targets for studying vulnerabilities. Authors in [43] select 15 plugins according to their development team (5 developed by companies and 10 by individual developers) and the number of downloads (5 with less than 100000 downloads and the rest with more). Their goal was to assess a set of functions to sanitize and validate data types. SourceForge[8] is a software platform for open source and business software. Authors in [59] obtain datasets for evaluation from publicly known vulnerable versions of programs hosted in SourceForge, such as *CuteSITE 1.2.3* (a content management system), *Faqforge 1.3.2* (a document creation and management system) and *Schoolmate 1.5.4* (a school administration system). Authors in [34] perform testing against two famous open-source WAFs: ModSecurity[9] and Naxsi[10].

Additional Tools. We have identified several tools that are commonly used in the papers reviewed. WEKA [55] is an open source, platform-independent and freely available tool that support researchers with the implementation of ML algorithms for data mining and ML experimentation [72]. Authors in [22] and [61] use the tool to extract feature from data sets and build models. Authors in [41] use WEKA to evaluate their dataset of false positive vulnerability instances using individual models and their combination via meta-models. Authors investigating PHP vulnerabilities in source code [41] compare their ML-based solutions against existing tools such as *Pixy* [27]; these tools are commonly preferred due to alternative options mostly being simple prototypes. Web scanners are often used as a baseline to compare new methods. ZAP [52] is the most popular web app scanner worldwide. It is free and open source. Wapiti[11] is a web application auditor that performs back-box scans. Firstly, Wapiti starts by crawling the target site, and identifying scrips that allow data injection. To verify if a script is vulnerable,

[5]https://github.com/hongliangliang/gptfuzzer

[6]https://github.com/payloadbox/xss-payload-list

[7]https://github.com/digininja/DVWA

[8]https://sourceforge.net

[9]https://github.com/SpiderLabs/ModSecurity

[10]https://github.com/nbs-system/naxsi

[11]https://wapiti-scanner.github.io/

Wapiti injects payloads (i.e., fuzzing). Burp Scanner [54] is an automated Dynamic Application Security Testing web vulnerability scanner. It is a commercial tool and therefore not open source. Burp Scanner is designed to replicate actions and methodologies of a skilled manual tester. Black Widow[12] is a Python-based web application scanner that is able to gather Open Source Intelligence Gathering (OSINT) and fuzz for OWASP vulnerabilities on a target website. In the context of discovering real-world vulnerabilities, authors in [32] report detection results and number of attempted requests of each one of the mentioned tools. They do this to produce a comparison against their proposed RL-based solution *Link* for XSS vulnerability detection in black-box settings. In their experiments, Burp found more bugs than *Link* because of a more advance crawling strategy that includes name guessing and extrapolation from naming conventions. When focusing the comparison on only the input parameters and URLs that *Link*'s crawler found, the RL-based solution was superior; it was able to find every vulnerability that Burp found, but with far fewer requests, showcasing promising pentesting capabilities. *Link* was superior than the ZAP, Wapiti and Black Widow in both finding vulnerabilities and number of attack request required. Authors in [73] include ZAP, Burp Suite and Wapiti between others.

As a takeaway, we draw the conclusion that researchers in this domain are inclined to use open-source tools for their development and experiments, limiting the use of proprietary tools to results comparison.

V. Discussion

This section delves into answering research questions, providing insights into the methodologies, effectiveness, and future directions of AI in web application penetration testing.

A. Answering Research Questions

Based on our review results, we can derive the following insights to answer the research questions.

RQ1: What AI methodologies are predominantly used in web applications penetration testing, and for what specific purposes? In the realm of AI-driven pentesting, ML stands as the primary area of focus, complemented by investigations into NNs and NLP, with RL receiving the least attention. Supervised learning, particularly, is extensively explored across various phases of pentesting, including scanning, enumeration, and exploitation. Research that explores the combination of supervised and unsupervised learning, indicates a growing interest in leveraging both approaches to develop more robust pentesting methods. Despite being less commonly investigated, RL is primarily applied in the exploit stage, suggesting its potential in pentesting. The combination of supervised and unsupervised learning, as well as RL, suggests a trend toward developing adaptive and autonomous systems capable of effectively navigating the complexities of web application security. The reviewed body of research predominantly focuses on enhancing the automation, efficiency, and precision of identifying and exploiting vulnerabilities, highlighting an integrated approach to enhancing the scanning, enumeration, and exploitation through AI methodologies. Readers can refer to 2 for a visual representation of the distribution of models for each of the pentesting stages. The scanning and enumeration stages heavily dominates in terms of the variety and frequency of algorithms used, indicating a significant focus on this initial phase of penetration testing within the surveyed literature. SVM and RF, which are both highly versatile and robust, are popular choices for classification tasks in security contexts. NB and LR are also extensively used, likely due to their simplicity and effectiveness in binary or multi-class classification tasks. In the exploit stage, where the goal is to achieve unauthorized actions on the system, there is a smaller variety of algorithms used, with GAN and HMM being notable for their specialized applications. The use of advanced and complex models like Transformers and Deep Reinforcement Learning algorithms (e.g., DQN, DDQN, PPO) is observed in the exploit stages, suggesting a trend towards employing sophisticated AI techniques to automate and optimize attack strategies. In the cases of Post-Exploit and Information Gathering, the lack of focus in these areas stop us from identifying meaningful trends.

RQ2: How do AI-driven web application pentesting tools compare in effectiveness and efficiency to traditional methods? AI-driven web application pentesting tools show promise in effectiveness and efficiency when compared to traditional manual and automated methods. For example, AI-based tools can perform certain tasks with fewer requests than traditional tools, as seen with the RL-based solution Link [32], which outperformed others like ZAP, Wapiti, and Black Widow in detecting vulnerabilities while requiring fewer attack requests. However, traditional tools like Burp Scanner, with advanced crawling strategies[13], can identify more bugs than some AI-based tools. From our review, various studies evaluated the effectiveness of their AI-based pentesting systems against known manual and state-of-the-art tools, showing promising results, e.g., in [7], the number of vulnerabilities discovered was significantly higher with AI; in studies such as [30] and [42], previously undiscovered vulnerabilities were identified using learning models. While most studies did not explicitly report time metrics, the few that did indicated clearly that the AI-based pentesting tools took less time than traditional methods. In [43], authors compare the learning-based vulnerability recognition tool DEKANT against Navex [3] (a non-AI exploit generation framework for web applications). Their findings confirm the effective detection capabilities of DEKANT, highlighting it as the tool with the highest accuracy and precision, along

[12]https://github.com/1N3/BlackWidow

[13]Burp identified more XSS vulnerabilities than Link due to its advanced page crawling features, such as the ability to guess names and extrapolate based on naming conventions.

TABLE II

REVIEWED PAPERS (49) ORGANIZED BY PUBLICATION YEAR AND ORDERED BY DESCENDING NUMBER OF CITATIONS AT THE TIME OF THE SURVEY (AUGUST 2024). LEGEND OF AI PARADIGMS: SUPERVISED (S), UNSUPERVISED (U), REINFORCEMENT LEARNING (RL), OPEN-SOURCE CODE (*). THE REST OF ALGORITHM ACRONYMS AND THEIR FULL NAMES CAN BE FOUND IN THE APPENDIX B.

Ref	Cited	Year	Pentesting Stage	AI	Algorithms
[61]	144	2013	Scanning/Enumeration	S+U	LR, MLP
[60]	111	2013	Scanning/Enumeration	S	NB, MLP
[59]	134	2014	Scanning/Enumeration	S+U	LR, RF
[44]	115	2014	Post-Exploit	S	SVM, LR, KNN, NB
[41]	161	2015	Scanning/Enumeration	S	ID3, C4.5, J48, RT, RF, NB, KNN, LR
[22]	67	2015	Scanning/Enumeration	S	SVM, NB, Bagging, RF, J48, JRip
[21]	45	2015	Scanning/Enumeration	S	KNN, GB, SGD, DT, SVM, NB, RF
[42]	60	2016	Exploit	S	HMM
[64]	2	2016	Scanning/Enumeration	S	SCW, SVM
[65]	31	2017	Scanning/Enumeration	S	LR, SVM
[7]	65	2018	Exploit	S	RF, RT
[30]*	48	2018	Scanning/Enumeration	S	DT, LR, NB, RF, TAN
[66]	8	2018	Scanning/Enumeration	S	NB, LR, SVM
[23]	5	2018	Scanning/Enumeration	S	NB, RT, RF, JRip, J48, SVM, Bagging
[10]*	46	2019	Scanning/Enumeration	S	SVM, DT, RF, GBDT, LR
[28]	10	2019	Scanning/Enumeration	S	RF, NB, J48
[33]	8	2019	Scanning/Enumeration	S+RL	SVM, MLP, DQN, LSTM
[17]	5	2019	Scanning/Enumeration	S	SGBT
[53]	2	2019	Scanning/Enumeration	S+U	SVM, PAA
[35]*	36	2020	Exploit	S+U	Transformers
[18]	20	2020	Scanning/Enumeration	S+U	LSTM
[70]*	5	2020	Exploit	S	GAN
[8]	4	2020	Information Gathering	S	BigARTM
[19]	2	2020	Scanning/Enumeration	S	DT, RF, MLP, NB, KNN, LR, SVM
[45]*	27	2021	Scanning/Enumeration	S	DNN
[5]*	6	2021	Exploit	U	Autoencoder
[37]	3	2021	Scanning/Enumeration	S	BPNN, GA
[48]	2	2021	Exploit	S+U	SVM, PAA, DAA
[32]*	13	2022	Exploit	RL	PPO, DQN, A2C
[31]	12	2022	Scanning/Enumeration	S	CNN
[49]	11	2022	Scanning/Enumeration	S	FSM
[43]	9	2022	Scanning/Enumeration	S	HMM
[39]	4	2022	Scanning/Enumeration	S	DT, KNN, RF, LR, SVM, LSTM, BiLSTM, GRU, BiGRU
[67]	2	2022	Scanning/Enumeration	S	LSTM, RF, GB, LR
[75]	1	2022	Scanning/Enumeration	S	Gated RNN
[47]	0	2022	Scanning/Enumeration	S	CNN, RNN, LSTM, BiLSTM
[38]	0	2022	Scanning/Enumeration	S	DT, SVM, NB, RT, RF, JRip
[36]	7	2023	Scanning/Enumeration	S	Graph CNN, RNN
[2]	2	2023	Scanning/Enumeration	S	Transformers
[68]	0	2023	Exploit	S	NB, LR, DT, RF, XGBoost
[74]	0	2023	Scanning/Enumeration	S	ASTNN, LSTM, SVM, ASTE
[73]	0	2023	Exploit	RL	DDQN
[34]*	0	2023	Exploit	U+RL	Transformers, MDP
[63]*	0	2023	Exploit	RL	Q-Learning
[26]	0	2023	Exploit	U	RF, Adaboost, SVM, RNN
[12]	0	2023	Exploit	S+RL	GAN
[29]	0	2024	Exploit	S	Transformers
[6]	0	2024	Scanning/Enumeration	U	K-means
[51]	0	2024	Exploit	S	LSTM

with the lowest false negative (FN) rate. DEKANT also exhibited the highest false positive (FP) rate, primarily due to the 58 FPs detected in a particularly vulnerable application. In contrast, NAVEX-f showed the poorest performance metrics, partly because it failed to include this application. Nevertheless, even when this application is excluded from the analysis, DEKANT continues to outperform NAVEX-f, and its FP rate shows a substantial decrease. Generally, the absence of standard baselines for evaluation and the diversity of approaches complicates making equitable comparisons between different AI-based methods, as well as understanding their performance relative to traditional methods.

RQ3: What are the recognized limitations and challenges for AI-driven web applications pentesting tools as identified in the literature? AI methods, especially supervised ML, heavily rely on high-quality annotated data. The effectiveness of these tools is directly influenced by the quality of the datasets they are trained on. Creating training datasets often requires significant manual effort, such as manually labeling source code from repositories or generating synthetic test cases. There is a noted scarcity of publicly available, well-annotated datasets for training and testing AI-driven pentesting tools, which limits research and development. While there are datasets like SARD that provide a range of artifacts with known vulnerabilities, ensuring that these datasets reflect real-world complexities and latest vulnerability trends is a continuous challenge. There is a need for common environments like DVWA to evaluate new AI-based approaches, but these may not always reflect the latest or most advanced vulnerabilities found in production environments. From our review, many papers focused on either static or dynamic approaches to vulnerability discovery, therefore, one area for improvement is to further study the use of hybrid-based approaches. Furthermore, only a handful of papers made use of RL. An increased adoption of RL-based approaches would allow to address the problem of dataset scarcity, as RL does not rely on annotated data as heavily as supervised methods. Another area of improvement is supporting open science and reproducible research, since only 9 out out of 44 papers made their artifacts available online (i.e., [10], [5], [34], [30], [63], [35], [70], [45], [32]). This ratio is in line with the statistics presented in other recent surveys [50], [57], therefore, this a generalized issue in the cybersecurity domain. The preferred way to share their projects is via Git repositories. Authors in [63] state that the data and code is available upon request, while authors in [49] mention that because of the size and propriety of the dataset used, it could not be shared. Authors in [60] mention the release of artifacts in the personal website of an author; however, we attempted to follow the provided link and the project was not currently accessible. Additionally, as can be seen in Table II, there is a correlation between releasing open source code and having a higher number of citations in the long term.

B. Future Research Directions

Research Gaps. SQL injection and XSS have consistently appeared in the OWASP top 10 for several years. Given the extensive research already conducted on these types of attacks, we anticipate that future studies will focus on other prominent OWASP vulnerabilities, such as cryptographic failures and Server-Side Request Forgery (SSRF). From our analysis, we can see that the post-exploit and information gathering aspects are minimal, exhibiting research gaps. Other gaps include AI applied to perform a chained attack or attack sequence (i.e., multi-stage attacks). From our side, apart from addressing these research gaps, future work will involve evaluating the effectiveness of specific algorithms across pentesting stages.

Additionally, we anticipate the emergence of several relevant directions in the near future:

Large Language Models (LLMs). We came across very recent pieces of work that leverage LLMs for penetration testing [2], [29]. It is worth noting that, there was a previous paper that still did not go through peer review, and therefore were excluded from the review: Authors in [16] used LLMs for pentesting for the first time, creating a new research direction.

Adversarial Attacks. Attacks against AI-based pentesting tools in web applications represent a sophisticated level of threat. These attacks are designed to exploit the weaknesses in the AI models themselves, rather than the web applications they are meant to protect. Authors in [26] explore this topic in the context of SQL injection detection. More recently, authors in [51] present a framework that utilizes multi-modal artifacts collected from triggered browser events to successfully distinguish human visitors from bots. They explore two adversarial attacks aiming to make the model classify modified bot interactions as human traces. In the reviewed paper [10], authors explicitly mention that their classifier is not designed to be resilient to adversarial attacks; they justify this by highlighting that their tool is not intended for attack detection, but for offensive purposes, that is, Cross-Site Request Forgery (CSRF) vulnerability detection. However, we consider that adversarial manipulations could be used to mislead the pentesting tool into missing CSRF attack vectors.

Additionally, adversarial attacks may transfer across models [15]. If a pentesting tool shares commonalities with other AI models (which is often the case), these transferable attacks could be effective for all of them.

Explainability. Explainable AI can play a significant role in AI-based pentesting for web applications by making the decision-making processes of AI systems transparent and understandable to humans. This is important in cybersecurity, where understanding the rationale behind a detection or a decision can significantly impact the response strategy. Moreover, explainability methods have the potential to support testers in finding workarounds to bypass AI defensive methods [57].

Enhancing Data Privacy in AI-driven Penetration Testing. Future research in AI-driven pentesting should prioritise the privacy of client data, developing methods that safeguard sensitive information during and after security assessments. This includes creating AI models that can effectively identify and protect personal and proprietary data from exposure or misuse, ensuring compliance with global data protection regulations.

VI. Conclusion

Our literature review has highlighted key trends and challenges in employing AI for penetration testing within web applications. Notably, machine learning, especially supervised learning, emerges at the forefront of research efforts, aiming to streamline vulnerability identification and exploitation processes. However, the efficacy of these AI methodologies hinges on access to well-annotated datasets, which are in short supply. While AI-based tools have proven to be more efficient than traditional approaches, they still face significant challenges, such as the need for enriched data and more realistic testing environments. Future research in this field will focus on the security and interpretability of AI-enabled tools, as well as leveraging the potential of generative AI.

Acknowledgment

This work was supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF), KASTEL Security Research Labs (structure 46.23.02) and University of Sheffield.

References

[1] Akamai: The rise of application and api attacks (Accessed: 2024)
[2] Akuthota, V., Kasula, R., Sumona, S.T., Mohiuddin, M., Reza, M.T., Rahman, M.M.: Vulnerability detection and monitoring using llm. In: WIECON-ECE'23. IEEE (2023)

[3] Alhuzali, A., Gjomemo, R., Eshete, B., Venkatakrishnan, V.: {NAVEX}: Precise and scalable exploit generation for dynamic web applications. In: Usenix Sec'18) (2018)

[4] Allix, K., Bissyandé, T.F., Klein, J., Le Traon, Y.: Androzoo: Collecting millions of android apps for the research community. In: MSR (2016)

[5] Amouei, M., Rezvani, M., Fateh, M.: Rat: Reinforcement-learning-driven and adaptive testing for vulnerability discovery in web application firewalls. IEEE Transactions on Dependable and Secure Computing **19** (2021)

[6] Antonelli, D., Cascella, R., Schiano, A., Perrone, G., Romano, S.P.: "dirclustering": a semantic clustering approach to optimize website structure discovery during penetration testing. Journal of Computer Virology and Hacking Techniques pp. 1–13 (2024)

[7] Appelt, D., Nguyen, C.D., Panichella, A., Briand, L.C.: A machine-learning-driven evolutionary approach for testing web application firewalls. IEEE Transactions on Reliability (2018)

[8] Astakhova, L., Medvedev, I.: Scanning the resilience of an organization employees to social engineering attacks using machine learning technologies. In: USBEREIT. IEEE (2020)

[9] Bassi, D., Singh, H.: A systematic literature review on software vulnerability prediction models. IEEE Access (2023)

[10] Calzavara, S., Conti, M., Focardi, R., Rabitti, A., Tolomei, G.: Mitch: A machine learning approach to the black-box detection of csrf vulnerabilities. In: EuroS&P. IEEE (2019)

[11] Caturano, F., Perrone, G., Romano, S.P.: Hacking goals: A goal-centric attack classification framework. In: ICTSS. Springer (2020)

[12] Chowdhary, A., Jha, K., Zhao, M.: Generative adversarial network (gan)-based autonomous penetration testing for web applications. Sensors **23** (2023)

[13] Chu, G.: Automation of Penetration Testing. The University of Liverpool (2021)

[14] Cisco Talos Intelligence Group: Quarterly threat report: Telecommunications and web application threats in q3 2023 (2023), accessed: 2024

[15] Demontis, A., Melis, M., Pintor, M., Jagielski, M., Biggio, B., Oprea, A., Nita-Rotaru, C., Roli, F.: Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In: USENIX Sec'19 (2019)

[16] Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., Rass, S.: Pentestgpt: An llm-empowered automatic penetration testing tool. arXiv (2023)

[17] Elish, M.: Enhanced prediction of vulnerable web components using stochastic gradient boosting trees. International Journal of Web Information Systems **15** (2019)

[18] Fidalgo, A., Medeiros, I., Antunes, P., Neves, N.: Towards a deep learning model for vulnerability detection on web application variants. In: ICSTW. IEEE (2020)

[19] Figueiredo, A., Lide, T., Matos, D., Correia, M.: Merlin: multi-language web vulnerability detection. In: NCA. IEEE (2020)

[20] Goh, K.C.: Toward Automated Penetration Testing Intelligently with Reinforcement Learning. Ph.D. thesis, National College of Ireland (2021)

[21] Guo, X., Jin, S., Zhang, Y.: Xss vulnerability detection using optimized attack vector repertory. In: CyberC. IEEE (2015)

[22] Gupta, M.K., Govil, M.C., Singh, G.: Predicting cross-site scripting (xss) security vulnerabilities in web applications. In: JCSSE. IEEE (2015)

[23] Gupta, M.K., Govil, M.C., Singh, G.: Text-mining and pattern-matching based prediction models for detecting vulnerable files in web applications. Journal of Web Engineering (2018)

[24] Haq, I.U., Khan, T.A., Akhunzada, A.: A dynamic robust dl-based model for android malware detection. IEEE Access **9** (2021)

[25] Harzevili, N.S., Belle, A.B., Wang, J., Wang, S., Ming, Z., Nagappan, N., et al.: A survey on automated software vulnerability detection using machine learning and deep learning. arXiv:2306.11673 (2023)

[26] Issakhani, M., Huang, M., Tayebi, M.A., Lashkari, A.H.: An evolutionary algorithm for adversarial sql injection attack generation. In: ISI. IEEE (2023)

[27] Jovanovic, N.: Pixy: A static analysis tool for detecting web application vulnerabilities. PenTestLab Blog (2012)

[28] Khalid, M.N., Farooq, H., Iqbal, M., Alam, M.T., Rasheed, K.: Predicting web vulnerabilities in web applications based on machine learning. In: INTAP (2019)

[29] Khan, S.: Ll-xss: End-to-end generative model-based xss payload creation. In: L&T'24. IEEE (2024)

[30] Kronjee, J., Hommersom, A., Vranken, H.: Discovering software vulnerabilities using data-flow analysis and machine learning. In: ARES (2018)

[31] Kumar, J., Santhanavijayan, A., Rajendran, B.: Cross site scripting attacks classification using convolutional neural network. In: ICCCI. IEEE (2022)

[32] Lee, S., Wi, S., Son, S.: Link: Black-box detection of cross-site scripting vulnerabilities using reinforcement learning. In: WWW (2022)

[33] Li, L., Wei, L.: Automatic xss detection and automatic anti-anti-virus payload generation. In: CyberC. IEEE (2019)

[34] Liang, H., Li, X., Xiao, D., Liu, J., Zhou, Y., Wang, A., Li, J.: Generative pre-trained transformer-based reinforcement learning for testing web application firewalls. IEEE Transactions on Dependable and Secure Computing (2023)

[35] Liu, M., Li, K., Chen, T.: Deepsqli: Deep semantic learning for testing sql injection. In: ISSTA (2020)

[36] Liu, Z., Fang, Y., Huang, C., Xu, Y.: Mfxss: An effective xss vulnerability detection method in javascript based on multi-feature model. Computers & Security **124** (2023)

[37] Luo, Y.: Sqli-fuzzer: A sql injection vulnerability discovery framework based on machine learning. In: ICCT. IEEE (2021)

[38] Manjunatha, K., Kempanna, M.: Count vectorizer model based web application vulnerability detection using artificial intelligence approach. Journal of Discrete Mathematical Sciences and Cryptography **25** (2022)

[39] Marashdih, A.W., Zaaba, Z.F., Suwais, K.: Predicting input validation vulnerabilities based on minimal ssa features and machine learning. Journal of King Saud University-Computer and Information Sciences **34** (2022)

[40] McKinnel, D.R., Dargahi, T., Dehghantanha, A., Choo, K.K.R.: A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. Computers & Electrical Engineering (2019)

[41] Medeiros, I., Neves, N., Correia, M.: Detecting and removing web application vulnerabilities with static analysis and data mining. IEEE Transactions on Reliability **65** (2015)

[42] Medeiros, I., Neves, N., Correia, M.: Dekant: a static analysis tool that learns to detect web application vulnerabilities. In: ISSTA (2016)

[43] Medeiros, I., Neves, N., Correia, M.: Statically detecting vulnerabilities by processing programming languages as natural languages. IEEE Transactions on Reliability **71** (2022)

[44] Medeiros, I., Neves, N.F., Correia, M.: Automatic detection and correction of web application vulnerabilities using data mining to predict false positives. In: WWW (2014)

[45] Melicher, W., Fung, C., Bauer, L., Jia, L.: Towards a lightweight, hybrid approach for detecting dom xss vulnerabilities with machine learning. In: WWW (2021)

[46] Methley, A.M., Campbell, S., Chew-Graham, C., McNally, R., Cheraghi-Sohi, S.: Pico, picos and spider: a comparison study of specificity and sensitivity in three search tools for qualitative systematic reviews. BMC health services research (2014)

[47] Millar, S., Podgurskii, D., Kuykendall, D., Martínez del Rincón, J., Miller, P.: Optimising vulnerability triage in dast with deep learning. In: AISec (2022)

[48] Moshika, A., Thirumaran, M., Natarajan, B., Andal, K., Sambasivam, G., Manoharan, R.: Vulnerability assessment in heterogeneous web environment using probabilistic arithmetic automata. IEEE Access **9** (2021)

[49] Munonye, K., Péter, M.: Machine learning approach to vulnerability detection in oauth 2.0 authentication and authorization flow. International Journal of Information Security **21** (2022)

[50] Olszewski, D., Lu, A., Stillman, C., Warren, K., Kitroser, C., Pascual, A., Ukirde, D., Butler, K., Traynor, P.: " get in researchers; we're measuring reproducibility": A reproducibility study of machine learning papers in tier 1 security conferences. In: CCS'23 (2023)

[51] Ousat, B., Luo, D., Kharraz, A.: Breaking the bot barrier: Evaluating adversarial ai techniques against multi-modal defense models. In: Companion Proceedings of the ACM on Web Conference (2024)

[52] OWASP ZAP Project: The owasp zed attack proxy project (2023)

[53] Padmanaban, R., Thirumaran, M., Sanjana, V., Moshika, A.: Security analytics for heterogeneous web. In: ICSCAN. IEEE (2019)

[54] PortSwigger Web Security: Burp suite - application security testing software (2023)

[55] Reutemann, P., Frank, E., Hall, M., Trigg, L.: Weka: Data mining tool (2013), accessed: 27/10/2023

[56] Saber, V., ElSayad, D., Bahaa-Eldin, A.M., Fayed, Z.: Automated penetration testing, a systematic review. In: MIUCC. IEEE (2023)

[57] Sanchez, G., Elbez, G., Hagenmeyer, V.: Attacking learning-based models in smart grids: Current challenges and new frontiers. In: eEnergy'24 (2024)

[58] Shah, S., Mehtre, B.M.: An overview of vulnerability assessment and penetration testing techniques. Journal of Computer Virology and Hacking Techniques **11**, 27–49 (2015)

[59] Shar, L.K., Briand, L.C., Tan, H.B.K.: Web application vulnerability prediction using hybrid program analysis and machine learning. IEEE Transactions on dependable and secure computing **12** (2014)

[60] Shar, L.K., Tan, H.B.K.: Predicting sql injection and cross site scripting vulnerabilities through mining input sanitization patterns. Information and Software Technology **55** (2013)

[61] Shar, L.K., Tan, H.B.K., Briand, L.C.: Mining sql injection and cross site scripting vulnerabilities using hybrid program analysis. In: ICSE. IEEE (2013)

[62] Software Assurance Automation Research Center (SAMATE): Software assurance reference dataset (sard) (2023)

[63] Sommervoll, Å.Å., Erdődi, L., Zennaro, F.M.: Simulating all archetypes of sql injection vulnerability exploitation using reinforcement learning agents. International Journal of Information Security (2023)

[64] Sonoda, M., Matsuda, T., Koizumi, D.: On the approximate maximum likelihood estimation in stochastic model of sql injection attacks. In: SMC. IEEE (2016)

[65] Sultana, K.Z.: Towards a software vulnerability prediction model using traceable code patterns and software metrics. In: ASE. IEEE (2017)

[66] Sultana, K.Z., Williams, B.J., Bosu, A.: A comparison of nano-patterns vs. software metrics in vulnerability prediction. In: APSEC. IEEE (2018)

[67] Thaqi, R., Vishi, K., Rexha, B.: Enhancing burp suite with machine learning extension for vulnerability assessment of web applications. Journal of Applied Security Research (2022)

[68] Ussatova, O., Karyukin, V., Zhumabekova, A., Begimbayeva, Y., Ussatov, N.: Designing a vulnerability threat detection scanner with the use of machine learning models. In: IAIT'23 (2023)

[69] Valea, O., Oprişa, C.: Towards pentesting automation using the metasploit framework. In: ICCP. IEEE (2020)

[70] Valenza, A., Demetrio, L., Costa, G., Lagorio, G.: Waf-a-mole: An adversarial tool for assessing ml-based wafs. SoftwareX **11** (2020)

[71] Verizon: 2020 data breach investigations report (2020), accessed on 09/10/2023

[72] Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques with java implementations. Acm Sigmod Record **31** (2002)

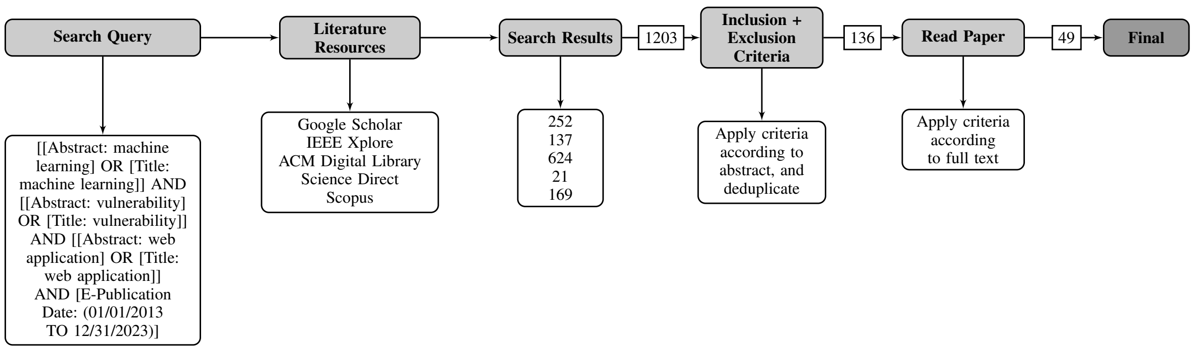
[73] Yao, Y., He, J., Li, T., Wang, Y., Lan, X., Li, Y.: An automatic xss attack vector generation method based on the improved dueling ddqn algorithm. IEEE Transactions on Dependable and Secure Computing (2023)

[74] Zhang, M., Zhu, H., Chang, X., Wang, Z., Wang, H.: Astnn-based system for auditing php code. In: ICEIB. IEEE (2023)

[75] Zheng, J., Li, J., Li, C., Li, R.: A sql blind injection method based on gated recurrent neural network. In: DSC. IEEE (2022)

Appendix

A. Paper selection methodology



Search Query
Literature Resources
Search Results
1203
Inclusion + Exclusion Criteria
136
Read Paper
49
Final
[[Abstract: machine learning] OR [Title: machine learning]] AND [[Abstract: vulnerability] OR [Title: vulnerability]] AND [[Abstract: web application] OR [Title: web application]] AND [E-Publication Date: (01/01/2013 TO 12/31/2023)]
Google Scholar
IEEE Xplore
ACM Digital Library
Science Direct
Scopus
252
137
624
21
169
Apply criteria according to abstract, and deduplicate
Apply criteria according to full text

B. List of Acronyms

TABLE III
Algorithm acronyms and other abbreviations used.

Acronym	Full Name
A2C	Advantage Actor-Critic
Adaboost	Adaptive Boosting
ASTNN	Attention-based Spatial-Temporal Neural Network
ASTE	Attention-based Spatial-Temporal Encoder
Bagging	Bootstrap Aggregating
BiGRU	Bidirectional Gated Recurrent Unit
BiLSTM	Bidirectional Long Short-Term Memory
BigARTM	Big Additive Regularization on Topic Models
BP NN	Back Propagation Neural Network
CNN	Convolutional Neural Network
DAA	Deterministic Arithmetic Automata
DDQN	Double Deep Q-Network
DNN	Deep Neural Network
DQN	Deep Q-Network
DT	Decision Tree
FSM	Finite State Machine
GA	Genetic Algorithm
GAN	Generative Adversarial Network
GB	Gradient Boosting
GBDT	Gradient Boosting Decision Tree
Gated RNN	Gated Recurrent Neural Network
Graph CNN	Graph Convolutional Neural Network
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
ID3	Iterative Dichotomiser 3
J48	J48 Decision Tree
JRip	Repeated Incremental Pruning
KNN	K-Nearest Neighbors
LR	Logistic Regression
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
MLP	Multi-Layer Perceptron
NB	Naive Bayes
PAA	Probabilistic Arithmetic Automata
PPO	Proximal Policy Optimization
RF	Random Forest
RNN	Recurrent Neural Network
RT	Random Tree
SCW	Soft Confidence-Weighted Learning
SGD	Stochastic Gradient Descent
SGBT	Stochastic Gradient Boosting Tree
SVM	Support Vector Machines
TAN	Tree Augmented Naive Bayes
XGBoost	eXtreme Gradient Boosting