CrawLLMentor: An LLM-Powered Tool to Assist Pen Testers in Identifying Business Logic Vulnerabilities

Nicolò Romandini[1,*], Luca Capacci[2] and Rebecca Montanari[1]

[1]Alma Mater Studiorum - University of Bologna, Bologna, Italy
[2]CryptoNet Labs s.r.l., Milano, Italy

Abstract

Web applications play a vital role in various critical aspects of daily life, from social platforms to healthcare and banking systems. Their widespread use and the massive volume of data flowing through them make them an attractive target for cyberattacks. While numerous tools exist to identify vulnerabilities, most focus primarily on technical issues. Business logic vulnerabilities, however, are often overlooked due to the challenges associated with their automated detection.

In this paper, we present CrawLLMentor, a novel black-box framework designed to assist penetration testers in identifying business logic vulnerabilities. The framework acts as an intelligent assistant for penetration testers, leveraging Large Language Models (LLMs) to analyze the semantics of web pages and enabling a deeper understanding of HTML element functionalities. By providing insights into the website's structure and behavior, the tool helps testers uncover potential flaws in business logic. We implemented and tested the tool on several web applications, demonstrating its effectiveness in real-world scenarios. This innovative approach enhances the security of web applications, addressing a critical gap in cybersecurity.

Keywords

Business Logic Vulnerabilities, Large Language Models, Machine Learning, Web Applications, Penetration Testing

1. Introduction

Web applications are a cornerstone of today's digital ecosystem, facilitating billions of interactions daily across various domains such as finance, healthcare, and social networking. This extensive usage makes them a prime target for cyberattacks. Consequently, ensuring their security is a top priority, with significant attention to identifying vulnerabilities. Among these, business logic vulnerabilities stand out as particularly critical. These vulnerabilities arise from flaws in an application's fundamental workflows or processes. When exploited, they can lead to severe outcomes, including data breaches and operational disruptions. Given the increasing complexity of cyber threats and modern web applications, detecting and mitigating business logic vulnerabilities has become crucial.

Unlike conventional technical vulnerabilities often addressed by automated tools, business logic vulnerabilities are deeply embedded within the application's functional design. Their detection largely depends on manual penetration testing, which, while effective, suffers from limitations such as high costs, scalability issues, and an inability to keep up with the dynamic evolution of digital environments. Furthermore, the intricate nature of business logic vulnerabilities frequently results in their oversight [1], and existing literature offers limited solutions due to the challenges of automated detection. The field of cybersecurity also faces a notable shortage of skilled professionals [2], particularly in the domain of penetration testing, where demand outpaces supply. This gap is especially evident for less experienced testers who may lack the expertise required for comprehensive vulnerability assessments. Manual penetration testing, while effective, often relies on advanced skills, making it difficult to scale and introducing inconsistencies in testing quality.

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT
*Corresponding author.
nicolo.romandini@unibo.it (N. Romandini); luca.capacci@cryptonetlabs.it (L. Capacci); rebecca.montanari@unibo.it (R. Montanari)
0000-0002-2820-5978 (N. Romandini); 0000-0002-3687-0361 (R. Montanari)

This paper addresses these challenges by introducing CrawLLMentor, a novel black-box framework that leverages Large Language Models (LLMs) to assist penetration testers. Acting as an intelligent assistant for testers, this framework enables the semantic analysis of web pages, enhancing the understanding of HTML element functionalities and their relationships. Beyond serving as an efficiency booster, CrawLLMentor also aims to standardize the penetration testing process, providing a consistent baseline of insights for testers of all experience levels. By leveraging the tool, even less experienced penetration testers can start with a common foundation of results, focusing their efforts on expanding beyond the findings of the framework. For expert testers, CrawLLMentor minimizes repetitive tasks, enabling them to concentrate on complex vulnerabilities. This dual functionality helps address the scalability and consistency challenges in penetration testing while making the field more accessible to newcomers.

The main contributions of this work include:

- **A semantic mapping tool**: This tool utilizes a custom-built crawler powered by LLMs to generate a detailed semantic map of the website, providing insights into the functionality of web elements and their relationships.
- **A risk evaluator tool**: This component evaluates the risk associated with the application's elements and HTTP requests. The risk is determined based on the identified functionalities and their potential to bypass or manipulate the application's logic.
- **A visualizer tool**: This tool provides a Graphical User Interface (GUI) to display the semantic map and risk visually. It allows penetration testers to easily explore and interact with the information, facilitating the identification of vulnerabilities and a clearer understanding of how web elements and their interactions contribute to the application logic.

The proposed framework aims to enhance the efficiency and accuracy of logic vulnerability detection in web applications, addressing a critical need in cybersecurity. The rest of the paper is organized as follows: Section 2 explores the nature of business logic vulnerabilities and reviews related work in the field, highlighting existing solutions and their limitations. Section 3 presents the architecture of our tool, and Section 4 shows its implementation and the experimental evaluation, demonstrating its effectiveness in helping the pen tester. Finally, Section 5 concludes the paper.

2. Business Logic Vulnerabilities

Business logic vulnerabilities refer to weaknesses or flaws in a software application's logical flow and decision-making processes, particularly those related to business rules and processes. Attackers can potentially exploit these vulnerabilities to manipulate the application in unintended ways. Unlike traditional security vulnerabilities, business logic vulnerabilities are more subtle and often stem from misunderstandings or oversights in designing and implementing the application's core functionality.

Common business logic vulnerabilities [3] include:

1. **Parameter Manipulation**: Attackers compromise the application's behavior by manipulating input parameters, violating semantic restrictions. For instance, modifying the product price in an online shopping app in an HTTP request allows attackers to purchase items for free. The absence or incorrect implementation of business logic and improper input validation on the server side contribute to these attacks, known as parameter manipulation/tampering attacks.
2. **Access-Control Bypass**: Implementing flaws that fail to incorporate Access Control Policies (ACPs) enable attackers to gain unauthorized access to restricted resources. Authentication and authorization bypass attacks can occur, such as accessing pages meant for logged-in users without proper authentication or accessing a restricted resource by directly pointing to its URL (authorization bypass or forceful browsing attack).
3. **Application Flow Bypass**: These attacks enable attackers to circumvent an application's intended workflow, as seen in scenarios where an attacker receives order confirmation without

paying in an online shopping application. This vulnerability, termed application flow bypass vulnerability, leads to workflow bypass attacks.

Mitigating business logic vulnerabilities requires a comprehensive understanding of the application's intended behavior, thorough testing of various scenarios, and implementing proper security controls throughout the development lifecycle. Regular security assessments, code reviews, and ongoing monitoring are essential to identify and address potential business logic vulnerabilities before malicious actors can exploit them.

2.1. Related Work

Numerous tools, both commercially available and in the literature, are designed to detect vulnerabilities in web applications. However, the majority of these tools concentrate solely on technical vulnerabilities, such as Cross-Site Scripting [4] or SQL Injection [5], completely overlooking those associated with business logic. Some solutions [6, 7, 8] have been proposed based on the analysis of server source code, known as white-box approaches. Only a limited number of approaches adopt a black-box approach. LogicScope, as proposed by Li and Xue in [9], identifies logic vulnerabilities by modeling the web application's behavior using a Finite State Machine (FSM). Vulnerabilities are identified when disparities are observed between the anticipated and actual FSMs. In [10], Li et al. propose BATMAN, a prototype that exposes access-control vulnerabilities in web applications. It achieves this by constructing attack vectors that violate the user-level and role-level policies inferred during the learning phase of the application. Pellegrino and Balzarotti in [11] focus on generating test cases to discover application vulnerabilities involving integrations with third-party APIs. Meanwhile, Scout, proposed by Wen et al. [12], specializes in identifying access-control vulnerabilities in applications that utilize MongoDB as the back-end. All the solutions presented have limitations, such as working only for a specific type of web application or requiring information modeled with human support.

CrawLLMentor addresses the gaps in finding business logic vulnerabilities outlined above. To achieve this, it leverages LLMs to understand the semantics of websites and the relationships between their elements and pages without relying on external resources such as the website's source code or previously conducted manual scans.

3. CrawLLMentor Architecture

This section introduces the architecture of CrawLLMentor. As depicted in Fig. 1, the framework consists of three main components:

- Semantic Crawler (1). Its task is to navigate the website by simulating user interaction. Each visited page is processed to extract global semantic information and information about individual elements using LLM.
- Risk Evaluator (2). It identifies potentially vulnerable or otherwise interesting HTTP elements and requests. This allows assigning a value representing the risk of each page on the site by aggregating the risk assessments of individual elements and requests.
- Visualizer (3). This GUI presents the semantic map and risk. It helps penetration testers to intuitively explore and understand the website's structure, functionalities, and associated risks, enhancing their ability to identify and analyze vulnerabilities.

The system's modular architecture allows individual modules to be used independently, without the need for all of them to be present simultaneously for operation. For instance, one could perform manual crawling and use the Risk Evaluator module to evaluate the risk of pages. Alternatively, one might only want to generate the site map without assessing the risk.

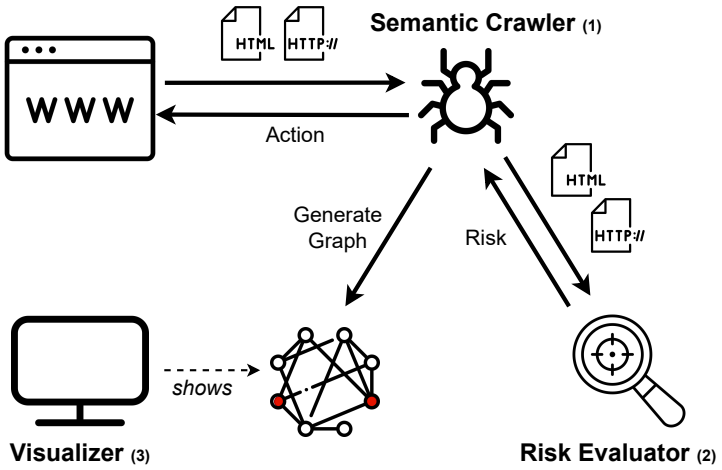


Figure 1: CrawLLMentor Architecture

3.1. Semantic Crawler

Traditional crawling techniques rely solely on HTTP interactions, neglecting the semantic aspects of various web pages. By doing so, they overlook entirely how multiple elements are related. In contrast, a human user disregards the underlying technicalities and perceives the application from a logical and semantic standpoint. They understand the flows and connections between different parts, the constraints, and the functionalities expressed by each page or element. Although analyzing the more technical aspects of the application is essential for uncovering many classes of vulnerabilities, those related to business logic surpass these. As explained in Section 2, these vulnerabilities are not based on technical flaws but on logical gaps, comprehensible only through a semantic analysis of the application. For this reason, the Semantic Crawler is one of the main components of CrawLLMentor. Its objective is to navigate the site as a human user would, not only through HTTP requests but also to generate a semantic map. We can model the web application as an FSM as in other works [9, 13]. An FSM is defined by states representing specific conditions or modes the system can assume at a given moment. Transitions, triggered by events or inputs, determine how the machine moves between these states, shaping its dynamic behavior. The crawler moves between various web pages, the states of the FSM, through elementary actions, such as button clicks and text field entries, or through complex sequences of elementary actions, such as filling out forms or completing purchases. This allows reaching parts of the site otherwise inaccessible, such as a user profile. The actions represent inputs to the FSM that can produce a transition from one state to another. The key point of this tool is its integration with LLM. Each web page is processed to build a state, a node in the semantic map, by comprehensively analyzing it and processing the individual HTML elements that compose it. Through the presentation of crafted prompts derived from the HTML of the page and its elements, the LLM can respond by providing its interpretation of the given input. This way, the semantic meaning of each element, such as a text field for entering a username, a button for accepting cookies, and the entire page, such as a login page, is extracted. At this point, the crawler can continue exploring the site, choosing the next action based on the interpretation given by the model of the page elements. At the end of the exploration, the tool generates a semantic map of the application that highlights the logical connections between various states. It showcases the high-level actions performed to transition from one state to another and the semantic interpretation of the pages and individual elements.

3.2. Risk Evaluator

While semantic analysis can assist in navigating and understanding the site, a more in-depth analysis is necessary to identify potential critical flows and vulnerabilities. This analysis relies on interpreting web

page elements and individual HTTP requests, including their parameters. This task is performed by the Risk Evaluator component, which leverages the LLM to assess the risk associated with each element and request. Specifically, the focus is primarily on the potential presence of sensitive or exploitable information for an attack. For instance, a login form or a shopping cart on an e-commerce site should increase the risk value associated with that page. A parameter referring to an identifier or a quantity should be flagged as a potential risk element. It is evident that identifying such information requires a semantic understanding of the context and cannot solely rely on a technical perspective. The LLM can provide crucial details about potential elements or parameters of interest in the page or HTTP request through appropriate prompts. The information output by this module is essential for understanding which parts of the application need to be tested or analyzed in detail.

3.3. Visualizer

The final component of CrawLLMentor is a visualization tool designed to enhance user experience by offering a comprehensive graphical representation of the data collected during the analysis. This tool empowers penetration testers to interactively explore the semantic map of the website, highlighting the risk levels associated with various pages and HTTP requests/responses. Starting with the site map generated by the Semantic Crawler, the Visualizer integrates risk evaluation from the Risk Evaluator. The map includes the states of visited web pages, the actions enabling transitions, and the corresponding HTTP requests and responses. Each state and HTTP element is visually represented based on risk level, allowing testers to pinpoint vulnerabilities or critical areas quickly. The interface supports filtering, enabling users to isolate specific state-action paths or focus on high-risk sections, such as pages vulnerable to parameter manipulation or access control bypass. The tool offers three levels of detail for map exploration. The general view simplifies the visualization by focusing on high-level transitions, providing a clear site structure overview. However, this view lacks detailed insights into specific HTTP elements. The intermediate view balances detail and clarity, summarizing HTTP requests and responses to facilitate quick identification of risky or interesting elements. While this view helps testers quickly spot potential vulnerabilities efficiently, it may aggregate information, potentially hiding issues in individual requests. The detailed view offers complete transparency, displaying every HTTP request and response for in-depth analysis. Although this level provides the most comprehensive insights, the vast amount of data can make it challenging to identify critical elements. The Visualizer streamlines the identification of business logic vulnerabilities and equips testers with insights to prioritize and address issues effectively.

4. Implementation and Experimental Evaluation

This section presents the implementation details of CrawLLMentor[1]. The tool was developed using Python and leverages OpenAI's GPT-4o mini, updated to July 2024, for semantic support.

4.1. Semantic Crawler

The Semantic Crawler uses the Selenium framework [14] to automate web interactions and dynamically analyze website behavior. It is designed to navigate web pages, simulate user interactions, and gather detailed information about the states and actions observed during crawling. The tool uses the Selenium Wire [15] library to capture HTTP requests and responses, enabling a deeper analysis of back-end interactions and potential vulnerabilities. The crawler is configured to work with Google Chrome and employs WebDriver Manager to install the browser driver automatically. It supports customization through command-line arguments, allowing users to specify URLs for websites to analyze, domain whitelists, output directories, and support files. These support files enable users to define arbitrary actions to execute in specific contexts, such as when recognizing a login page or a cookie banner.

[1] https://github.com/MMw-Unibo/CrawLLMentor

Additionally, parameters like cookies or headers can be specified to bypass actions such as authentication. As explained in Section 3.1, the crawler generates a site map represented as an FSM, primarily composed of states and actions. The first action is always loading the initial URL the user provided, corresponding to visiting the first state. Each state is represented by extracting the page's text, body, and HTML elements. The crawler utilizes the LLM to generate a semantic description with prompts such as:

```
Describe the functionality and the semantic meaning of the visible HTML page.
```

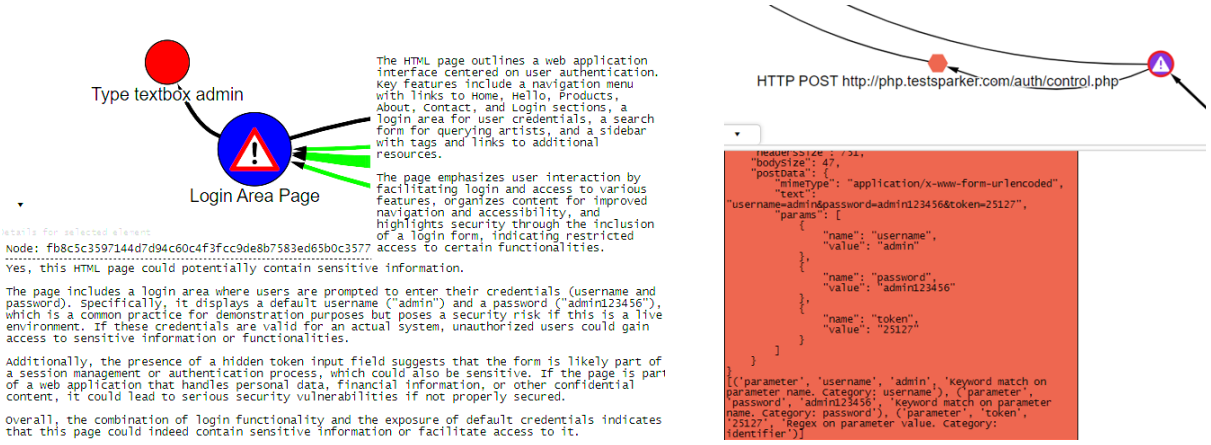
followed by the extracted HTML body. The model interprets the structure and content of the page, providing a meaningful summary. The crawler also identifies a list of possible actions for each state, such as clicking a button, typing into a field, selecting an option (`<select>`), or following a link. The currently supported actions include Click, Type, Select, Follow, and MultipleActions (sequences of the previous types of actions). The framework is designed to be extensible and can support new actions in the future. Once the actions are generated, they are sorted based on customizable criteria, such as the `z-index` value or a preference for Type over Click actions. The crawler also captures all HTTP requests and responses associated with each action. These data and information about states and actions are sent to the Risk Evaluator for a risk analysis. After the analysis, the crawler executes the first action in the list and repeats the cycle. It populates a graph using the NetworkX library [16] during this process, progressively building an FSM representation of the website.

4.2. Risk Evaluator

The Risk Evaluator analyzes the information extracted by the Semantic Crawler to identify potentially critical, high-risk, or vulnerable states. This module relies on two main tools: the LLM and support files provided by domain experts. The LLM helps understand the high-level semantics of the pages, identifying sensitive elements such as login forms or shopping carts in an e-commerce context. This is particularly useful given the lack of standardization in web page designs, making such analyses challenging without advanced semantic support. Support files, on the other hand, are used to analyze HTTP requests and responses, which tend to follow more standardized patterns. For instance, a numerical value might represent a user ID, while parameter names might indicate quantities of purchased items or authentication tokens. While an LLM could also be used for this analysis, leveraging support files reduces errors in these more technical tasks. After analyzing page elements and HTTP requests, the Risk Evaluator returns its results to the Semantic Crawler, which integrates them into the site map and enriches it with risk data.

4.3. Visualizer

At the end of the crawling process, the Semantic Crawler transfers the site map to the Visualizer to generate a visual representation. As described in Section 3.3, the Visualizer supports three levels of detail, allowing users to focus on specific aspects quickly. For each level, an HTML page is generated containing the FSM graph created by the Semantic Crawler, using the Gravis library [17]. The graph uses different colors to distinguish between nodes, including blue for states, dark green for out-of-scope states, red for actions, orange for the start action, and yellow and green for HTTP requests/responses. This color-coding allows users to differentiate between various elements at a glance quickly. Additionally, states change to a distinct color if they contain a form, as forms are typically high-risk elements. This feature enables users to identify such elements immediately. Finally, components identified as high-risk, such as web pages containing sensitive data or HTTP requests with exploitable parameters, are marked with an exclamation mark icon to draw attention to their sensitivity. Moreover, the tool incorporates screenshots with semantic annotations for each state, detailing if the state could be considered vulnerable or sensitive and why. For actions, the screenshots emphasize the specific elements involved in interactions. Additionally, when hovered, all nodes of the map display LLM-generated semantic descriptions, offering context and further enhancing the understanding of their functionality. This representation provides an interactive site visualization with semantic annotations, information



(a) Login page detected and LLM explanation.

(b) Login HTTP request detected.

Figure 2: Snapshots from the graph generated on `php.testsparker.com`

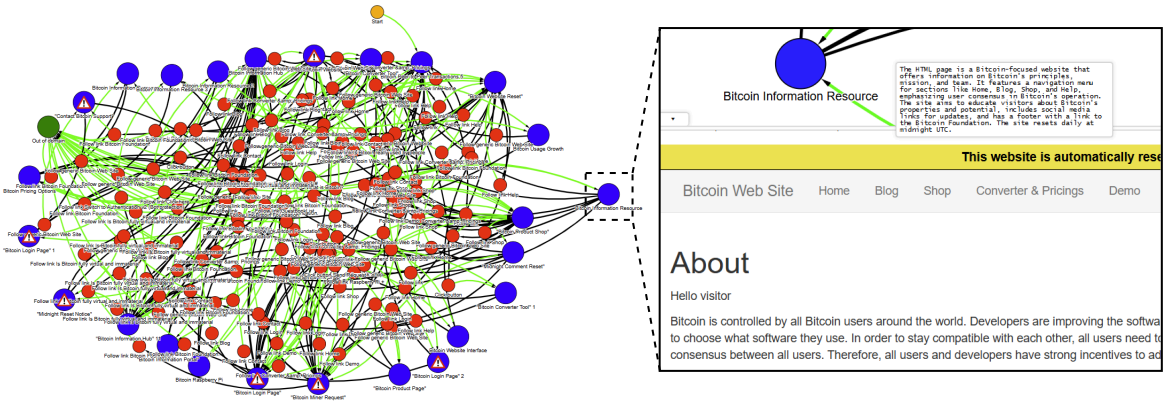


Figure 3: State-Action graph generated on `aspnet.testsparker.com`, with a zoomed-in view of a state, showing its screenshot.

on states and actions, and highlighted risk indicators. This approach facilitates penetration testers' analysis and understanding of the site's dynamics.

4.4. Experimental Evaluation

To evaluate our tool's capabilities, we test it on multiple websites. Due to privacy constraints, we cannot disclose the names of some of the sites used in our evaluation. Therefore, we present visual examples and results exclusively from publicly accessible websites, including test environments such as those provided by Netsparker [18]. These tests enable the tool to generate site maps, highlighting potential vulnerabilities and critical areas. Figure 2 illustrates a graph section generated after a simulation on `php.testsparker.com`. Figure 2a shows a state containing a login form, correctly highlighted with an exclamation mark, indicating it as a sensitive state. Additionally, when hovered over, the semantic description generated by the LLM is displayed. Moreover, a red node is also visible, indicating the tool's next action: enter the username into the text field to complete the login process. Figure 2b shows a highlighted HTTP request in red with an exclamation mark, indicating the presence of potentially manipulable parameters. In this instance, the parameter is tied to clear-text user credentials.

Figure 3 illustrates the graph obtained from `aspnet.testsparker.com`. The nodes follow the color scheme presented earlier. Although condensed due to space limitations, each node can be moved and rearranged, and the graph can be zoomed in or out, all for a clearer view. This visualization is essential for providing a quick overview of the website's structure and identifying interesting states. On the

right, the figure shows a zoomed-in view of a state displayed after clicking on its node, highlighting a screenshot captured by the tool and its semantic explanation generated by an LLM. In this case, the state is not considered sensitive or vulnerable, so the exclamation mark is not displayed.

A penetration tester beginning the website analysis can leverage this information to focus on specific areas of the site or potential requests. By identifying sensitive states, such as login forms or areas with manipulable parameters, the tester can prioritize these high-risk elements for deeper investigation. This approach allows for more efficient vulnerability identification, helping the tester direct their efforts toward parts of the site most likely to reveal logic vulnerabilities, such as forms or HTTP requests susceptible to parameter manipulation.

5. Conclusion

Web applications have become one of the most widely utilized tools by online users for various purposes. Ensuring their security is, therefore, a mandatory and primary goal. While many tools analyze web applications to find security vulnerabilities, most focus on specific classes of vulnerabilities. Due to their complex nature, these tools often overlook business logic vulnerabilities. However, they represent one of the major risks for web applications. This work aims to fill the gap in current tools by introducing CrawLLMentor, a framework that helps pen testers understand the semantics of the website and discover business logic vulnerabilities. A crucial and innovative aspect of research in this field is leveraging LLM. This allows for a more ”human” perspective in the analysis, comprehending the connections between elements and their semantic functionality.

Acknowledgments

This work was partially supported by the SERICS (PE00000014) project under the MUR National Recovery and Resilience Plan program funded by the European Union - NextGenerationEU.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] G. Stergiopoulos, B. Tsoumas, D. Gritzalis, On business logic vulnerabilities hunting: The app_loggic framework, in: International Conference on Network and System Security, 2013. URL: https://api.semanticscholar.org/CorpusID:31772021.

[2] Gartner, Gartner predicts nearly half of cybersecurity leaders will change jobs by 2025, 2024. URL: https://www.gartner.com/en/newsroom/press-releases/2023-02-22-gartner-predicts-nearly-half-of-cybersecurity-leaders-will-change-jobs-by-2025, [Online; accessed 2024-12-15].

[3] G. Deepa, P. S. Thilagam, Securing web applications from injection and logic vulnerabilities: Approaches and challenges, Information and Software Technology 74 (2016) 160–180.

[4] M. Mohammadi, B. Chu, H. R. Lipford, Detecting cross-site scripting vulnerabilities through automated unit testing, in: 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2017, pp. 364–373.

[5] W. G. Halfond, J. Viegas, A. Orso, et al., A classification of sql-injection attacks and countermeasures, in: Proceedings of the IEEE international symposium on secure software engineering, volume 1, IEEE, 2006, pp. 13–15.

[6] D. Balzarotti, M. Cova, V. V. Felmetsger, G. Vigna, Multi-module vulnerability analysis of web-based applications, in: Proceedings of the 14th ACM conference on Computer and communications security, 2007, pp. 25–35.
[7] V. Felmetsger, L. Cavedon, C. Kruegel, G. Vigna, Toward automated detection of logic vulnerabilities in web applications, in: 19th USENIX Security Symposium (USENIX Security 10), 2010.
[8] F. Sun, L. Xu, Z. Su, Detecting logic vulnerabilities in e-commerce applications., in: NDSS, 2014.
[9] X. Li, Y. Xue, Logicscope: Automatic discovery of logic vulnerabilities within web applications, in: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, 2013, pp. 481–486.
[10] X. Li, X. Si, Y. Xue, Automated black-box detection of access control vulnerabilities in web applications, in: Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, 2014, pp. 49–60.
[11] G. Pellegrino, D. Balzarotti, Toward black-box detection of logic flaws in web applications., in: NDSS, volume 14, 2014, pp. 23–26.
[12] S. Wen, Y. Xue, J. Xu, L.-Y. Yuan, W.-L. Song, H.-J. Yang, G.-N. Si, Lom: Discovering logic flaws within mongodb-based web applications, International Journal of Automation and Computing 14 (2017) 106–118.
[13] G. Deepa, P. S. Thilagam, A. Praseed, A. R. Pais, Detlogic: A black-box approach for detecting logic vulnerabilities in web applications, Journal of Network and Computer Applications 109 (2018) 89–109.
[14] Selenium, Selenium, 2024. URL: https://www.selenium.dev/, [Online; accessed 2024-12-15].
[15] S. Wire, Selenium wire, 2024. URL: https://github.com/wkeeling/selenium-wire, [Online; accessed 2024-12-15].
[16] NetworkX, Networkx, 2024. URL: https://networkx.org/, [Online; accessed 2024-12-15].
[17] Gravis, Gravis, 2024. URL: https://robert-haas.github.io/gravis-docs/, [Online; accessed 2024-12-15].
[18] Netsparker, Vulnerable web apps - testsparker, 2024. URL: http://testsparker.com/, [Online; accessed 2024-12-15].