Mitigating Insecure Outputs in Large Language Models(LLMs): A Practical Educational Module

MD Abdul Barek[1*], Md Mostafizur Rahman[2†], Mst Shapna Akter[3*], A.B.M Kamrul Islam Riad[4*], Md Abdur Rahman[4*], Hossain Shahriar[5‡], Akond Rahman[5§], Fan Wu[5¶],

[*] Department of Intelligent Systems and Robotics, University of West Florida, USA
[†] Dept. of Cybersecurity and Information Technology, University of West Florida, USA
[‡] Center for Cybersecurity, University of West Florida, USA
[§] Computer Science and Software Engineering, Auburn University, USA
[¶] Dept. of Computer Science, Tuskegee University, USA
{[*] barek2k2@gmail.com,[†] md.mostafizur.rn@gmail.com, [*] jannatul.shapna99@gmail.com, [*] kamrulriad75@gmail.com, [*] mdrahman.uwf@gmail.com, [‡] hshahriar@uwf.edu, [§] Azr0154@auburn.edu, [¶] Fwu@tuskegee.edu}

***Abstract*—Large Language Models (LLMs) have extensive ability to produce promising output. Nowadays, people are increasingly relying on them due to easy accessibility, rapid and outstanding outcomes. However, the use of these results without appropriate scrutiny poses serious security risks, particularly when they are integrated with other software, APIs, or plugins. This is because the LLM outputs are highly dependent on the prompts they receive. Therefore, it is essential to carefully clean these outputs before using them in additional software environments. This paper is designed to teach students about the potential dangers of contaminated LLM output within the context of web development through prelab, hands-on, and postlab experiences. Hands-on lab provides practical guidance on how to handle LLM vulnerabilities to make applications safe with some real-world examples in Python. This approach aims to provide students with a deeper understanding of the precautions necessary to ensure software against the vulnerabilities introduced by LLM output.**

***Index Terms*—Large Language Models, Cybersecurity, Insecure Output, Sanitization, Authentic Learning.**

I. Introduction

The Large Language Model has rapidly gained huge popularity worldwide and has already been widely adopted by a diverse range of users for generating human-like text outcomes. LLM has also recently been widely accepted in academia. The output of LLM is basically influenced by the input prompts, and it can be modified iteratively by changing the prompts. This means that although the LLM relies on a large amount of training data, the final output can still be controlled by carefully providing the prompts [1]. Although the LLM output is different in various contexts, this paper aims mainly to focus on some specific areas for students to learn possible vulnerabilities of the LLM output and to understand how to handle them properly in the real world. This paper also aims to equip students with the knowledge and skills necessary to critically evaluate and secure LLM output, ensuring safer integration into web development and other software applications.

Our approach utilizes authentic learning [2] concepts to equip students with the skills needed to identify and address security vulnerabilities in the LLM output. The reason we focus on hands-on learning is because it deepens understanding and proficiency by directly involving students with real-world problems and solutions. Engaging in practical tasks not only enhances active learning but also boosts a sense of competence and mastery, significantly improving knowledge and acquisition. Furthermore, by integrating authentic learning principles, we highlight the relevance and applicability of the skills taught, enhancing how effectively this knowledge can be transferred and applied in real-world situations [3, 4].

To address these things, we propose authentic learning modules meticulously designed to guide students about handling insecure outputs from Large Language Models (LLMs). These modules are carefully designed to include prelab orientations, hands-on activities, and postlab reflections, creating a comprehensive educational experience that prepares students for their future endeavors. To get an overall visual understanding, please follow Figure 1:

In Figure 1, we illustrate the complete process of cleaning the output of a large language model (LLM) to ensure that it is safe to use as input for other software services. We started with an arbitrary actor who intentionally offers a biased prompt to the LLM in order to generate vulnerable results that are passed to a tool to thoroughly filter out vulnerable parts from it. After proper cleansing, the final output then becomes safe and it can now be fed into the other software, API, or plugins.

We demonstrate to students with real-world scenarios illustrating the risks associated with directly rendering HTML and JavaScript in browsers without proper security measures. We discussed how image and iframe HTML tags, when src tags are improperly generated by the LLM, can initiate HTTP requests to external sites, potentially compromising browser security and leading to data privacy violations. Additionally, we cover the dangers of arbitrary executable JavaScript code generated by LLMs, which can be dangerously harmful when rendered directly in browsers without further scrutiny. To address these issues, we showed Python code how to clean HTML and JavaScript using the escape method, and applied regular expressions to thoroughly remove any JavaScript code from the HTML.

We also presented a case study on URL query strings that is produced by LLMs. If these query strings contain malicious data, they pose a significant risk to web servers. To mitigate this risk, we utilized Python's quote method to safely encode these query strings into URLs.

Moreover, we delved into the common issue of SQL injection, a well-known concern in database management systems. We explain how SQL queries, if partially generated by LLMs and used without scrutiny, can expose databases to attacks. To safeguard databases, we encouraged the use of parameterized query execution, which helps ensure the security and integrity of database operations.

The rest of the paper is structured as follows. Section II outlines a detailed labware setup, which is divided into subsections covering pre-lab, hands-on, and post-lab activities. Section III discusses related work, Section IV details the results of the student surveys, Section V explores potential directions for future research, Section VI concludes

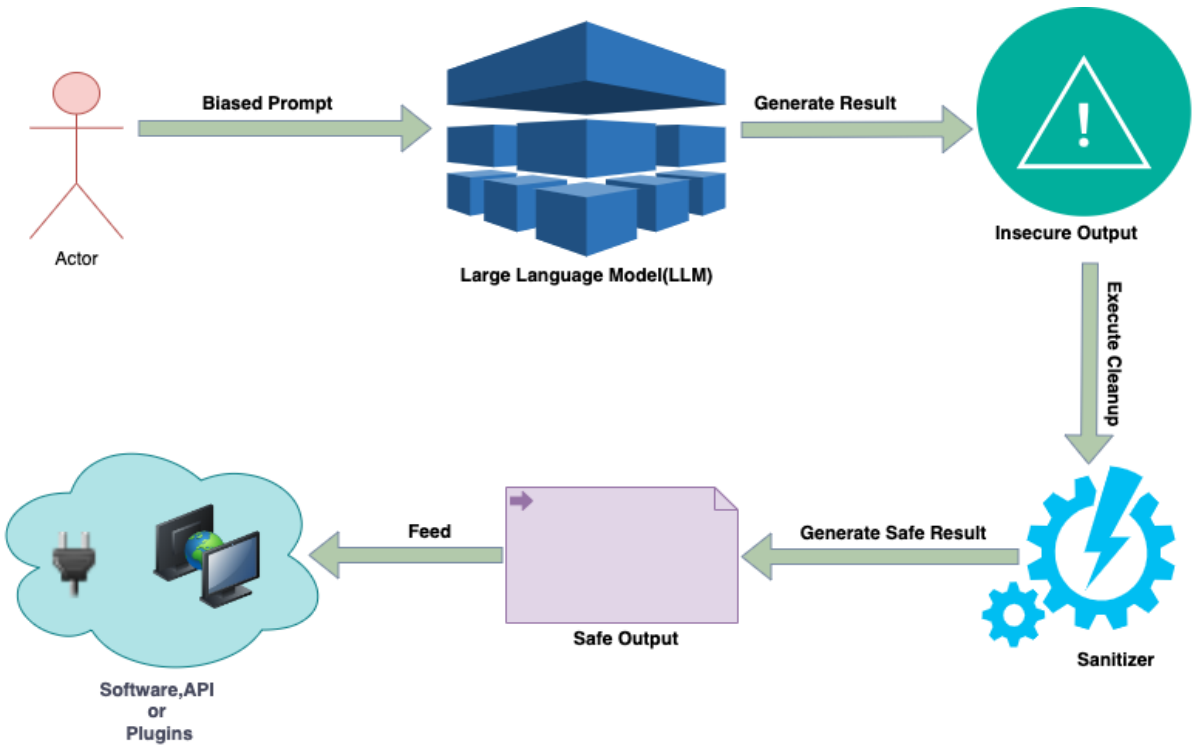


Fig. 1: Life cycle of Insecure output handling of LLM

the paper and finally Section VII contains acknowledgements.

II. Labware Setup

The labware is organized into three main sections: prelab, hands-on, and postlab. They are carefully structured to provide a thorough learning experience which offers a comprehensive learning experience, enabling students to progressively build their knowledge from the basic to advanced level. The pre-lab section equips learners with essential foundational knowledge to the topic. This is followed by the hands-on lab, which offers detailed insights and hands-on with Python examples, enhancing deeper understanding and skill. The post-lab section lastly inspires students to delve deeper into the subject, supporting further research. More importantly, we have provided URLs to the Google website for easy access from anywhere. Refer to the labware design as a learning pathway depicted in Figure 2: The base of the pyramid represents the foundational aspects, the middle section encompasses the hands-on experience, and the upper part, the post-lab, advances the learners to a higher level of expertise.

A. *Pre-lab*

The pre-lab section covers the basics of very common web vulnerabilities such as Cross Site Request Forgery(CSRF), Cross Site Scripting (XSS), Server Side Request Forgery (SSRF), and SQL Injection. It shows how these problems can show up in the outputs from Large Language Models (LLMs) when used in web development. It also explains why it is important to handle these issues carefully, especially when we use these outputs directly as inputs for other software services without checking them. This is essential to prevent security risks that could harm software and associated data.

B. *Hands-On*

In the hands-on lab section, students will dive deeper into the practical applications of handling outputs from Large Language Models (LLMs) to enhance security and vulnerabilities in software development workflows. This section provides real-world examples, complete with Python code snippets, detailed explanations, and screenshots, showing how to address vulnerabilities associated with LLM output. We demonstrated using an open-source LLM model from Hugging Face's Transformers library, and we developed a Python method that accepts a prompt and generates an output. The output of this Python method is flexible and smart, as it can handle any given instructions and deliver AI-powered results based on the prompts provided. By frequently calling this method, students have the opportunity to freely experiment and observe how the LLM might generate biased output.

Web browsers are the primary means through which users interact with the internet daily, but many users do not regularly update their browsers. As a result, their devices are at risk of being compromised when they visit vulnerable or insecure websites[5, 6]. The rendering of unfiltered content directly in a browser can pose serious security threats [7]. Likewise, accepting arbitrary content in APIs and plugins is risky because they might contain malicious data or executable code.

So, in the context of web development, we discussed how directly rendering this output in a browser could lead to serious security risks such as XSS [8] and CSRF [9], if not properly cleaned up. We demonstrated how these outputs can be cleaned up using techniques like Python provided methods named 'escape', 'quote', and regular expressions. We explain how a query string for an URL could potentially pose Server Side Request Forgery(SSRF) [10, 11] if it is generated from LLM and we show techniques such as URL encoding to mitigate this issue. Furthermore, we explain the risks of SQL injection attacks

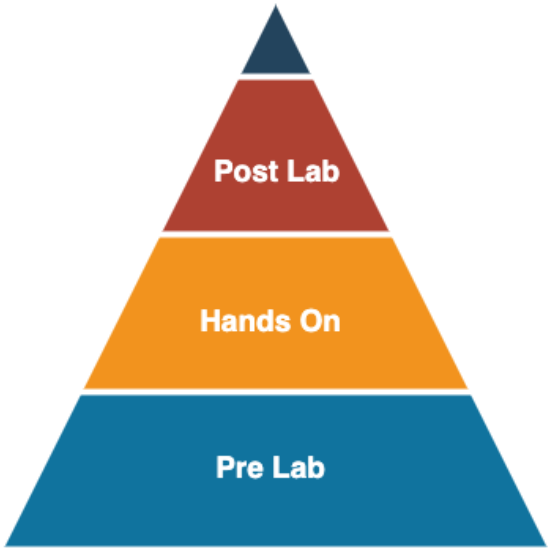


Fig. 2: Labware Setup

[12], a very well-known issue, in databases, particularly when parts of SQL queries are derived from LLM output, and showed how to eliminate such SQL injection attacks.

C. *Post-lab*

The post-lab section is designed to inspire students to dive deeper into the subject matter and engage in more thorough research. It encourages them to reflect on their hands-on experiences and apply their newly acquired knowledge in broader contexts.

III. Related works

Calò et al. [13] introduce a new way to use large language models (LLM) to help people generate websites by describing what they want in plain words through prompts. Their main idea is to guide the LLM to produce a website draft using a special setup that makes sure the LLM sticks to a certain template. They show how users describe website features, and the LLM generates the corresponding HTML and CSS in response. Le et al. [14] discuss the use of Large Language Models (LLMs) like ChatGPT and Bard for the automatic repair of vulnerabilities in JavaScript code. The authors also contribute to how LLM-generated outputs can be refined to reduce risks, especially in programming and web development scenarios where security is critical. Hong et al. [15] introduce the Knowledge-to-SQL framework, designed to improve SQL generation from text by integrating a specialized Data Expert Large Language Model (DELLM). Their innovation addresses the limitations of existing text-to-SQL models that often produce inaccurate SQL due to an incomplete understanding of the database schema and the context of the queries. The paper details the development and fine-tuning of DELLM, including a novel training strategy through natural language processing. Oh et al. [16] explore the real-world implications of poisoning attacks on AI-powered coding assistant tools like ChatGPT. They investigate how such attacks can literally introduce vulnerabilities into the code suggestions provided by LLM tools, potentially leading to security breaches in software development. Siddiq and Santos [17] introduce the security vulnerabilities in code produced by Large Language Models (LLMs). They highlight that while the functional correctness of code is frequently assessed, security aspects are often neglected. To address this, the authors introduce the SALLM framework, which includes a new dataset of security-focused Python prompts for testing generated code, along with innovative metrics specifically designed to evaluate the security of code generated by LLMs. This framework aims to reduce potential security risks in code that is generated automatically. He and Vechev [18] focus on enhancing the security of code generated by Large Language Models (LLMs) through a novel framework named SVEN which introduces controlled code generation, where code security is manipulated via binary properties that guide the LLM to produce secure code. Luo et al. [19] propose a novel method called Guide-Align, focusing on improving the safety and quality of outputs from Large Language Models (LLMs). Their approach involves creating a comprehensive library of detailed guidelines. Their proposed method uses a two-stage process where a safety-trained model initially identifies potential risks and formulates appropriate guidelines for different inputs. These guidelines are then used by a retrieval model to guide LLMs during response generation to ensure that safe, reliable outputs align with human values. Jesse et al. [20] investigate the susceptibility of Codex, AI-enabled Copilot by Github, to generate coding errors, particularly focusing on a type of bug known as single statement bugs (SStuBs). Codex, which is trained on public GitHub code that may contain bugs and vulnerabilities. This research finds that while Codex can help avoid some types of these simple bugs, they are twice as likely to produce known, exact copies of these bugs compared to correct code. The authors propose strategies to minimize the occurrence of these bugs while increasing the generation of accurate, bug-free code. Vörös et al. [21], introduces a cutting-edge method for URL categorization using Large Language Models (LLMs) aimed at enhancing web content filtering to protect organizations from legal and ethical risks. It restricts access to high-risk or dubious sites, and promotes a secure, professional workplace.

While these studies provide very useful information, it is important to note that none specifically address how to manage insecure LLM output for the academic domain. Although learning methods such as project-based, case study-based, and authentic learning have been extensively used in numerous fields, there is still a gap in the literature in focusing on the implementation of insecure LLM output handing in

the web development context. Therefore, our research seeks to bridge this gap by developing authentic learning modules to manage insecure LLM production in the educational sector.

IV. Students Survey

We conducted surveys for the prelab and postlab on large language model security by asking various questions. The surveys are offered in both quantitative and qualitative formats.

A. *Pre-lab Survey*

We asked students based on their prior work experience related to Computer and Artificial Intelligence and found different level of their roles such as Android Developer, Programmer, Software Developer and Software Engineer. The students were asked to identify their level of experience in the Large Language Model(LLM). According to the pre-lab survey, a majority of the students lacked experience in it. Among them, 33% of the students had no experience and 66% of students had limited themselves to a moderate level of knowledge about it. We also asked about their level of experiences at vulnerabilities in Large Language Model. It was surprising that almost 45% of the students had almost no experience in this area and the rest had very little knowledge about it. Furthermore, we asked students very important questions regarding their learning preferences with five different choices: Strongly Agree, Agree, Neutral, Disagree and Strongly Disagree. From the learning preference questions, it is observed that significant number of students are strongly agreed on learning better by hands on lab which are 78%. It proves that learning through hands-on is more effective for pupils.

TABLE I: Responses of students on their learning preferences

Question Title	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
I learn better by hands-on lab work	78%	22%	0%	0%	0%
I learn better by listening to lectures	22%	67%	11%	0%	0%
I learn better by personally doing or writing by examples	44%	33%	11%	11%	0%
I learn better by reading the material on my own	22%	44%	33%	0%	0%
I learn better by having a learning/tutorial system that provides feedback	44%	44%	11%	0%	0%

B. *Post-lab Survey*

In the post-lab survey, we collected responses from a total of 9 students. The survey was conducted after the students participated in the hands-on module. We asked various questions to the students about the benefits of the hands-on lab and also asked whether pre-lab session adequately prepared them to grasp the topic. The results indicated that authentic learning practices in the field of Cybersecurity issues in LLM outcomes are viewed promisingly positive. The survey included questions with five options: Strongly Agree, Agree, Neutral, Disagree, and Strongly Disagree. We observed some interesting facts from the post-lab survey. We found that the category "Strongly Agree" is between 78% and 89% on the benefits of the post-lab session concerning LLM security. Nobody disagreed or strongly disagreed our authentic learning process. Table II displays the results in percentages, while Figures 3, 4 and 5 illustrate the detailed responses from the nine students, presented as counts.

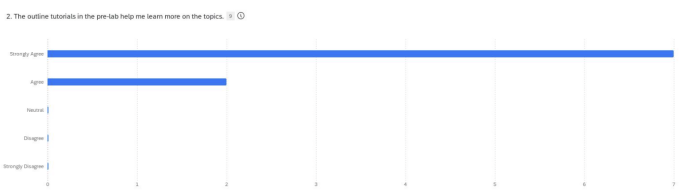
TABLE II: Responses of 9 students on LLM post lab survey

Question Title	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
I liked working with python and Colab	78%	22%	0%	0%	0%
The outline tutorials on pre-lab help me more on the pre-lab	78%	22%	0%	0%	0%
The hands-on on lab helps me understand better on Cybersecurity issues in Large Language Model during development and operation	89%	11%	0%	0%	0%
The hands-on lab helps me learning experience Cybersecurity on Large Language Model	89%	0%	11%	0%	0%
The real world relevant applications engage my learning on Cybersecurity in Large Language Model	78%	22%	0%	0%	0%
The learning modules help me apply learned knowledge to solve cybersecurity problems in Large Language Model	78%	22%	0%	0%	0%

It is important to note that students not only answered the questions we asked but provided insightful feedback on our another question. The question was "Please add any additional comments about this LLM security hands on project, either what you liked or disliked and make any suggestions for further improvement". Here are some positive comments we received: "The hands on module is great to build basic foundation on Software security on Large Language Model Output Concerns", "I liked the detail examples and code for demo on LLM vulnerabilities", "liked the process", "The labs are well designed and conveyed the concepts clearly with code and example", "Really helpful explanation of each line of code".

V. Future Research Directions

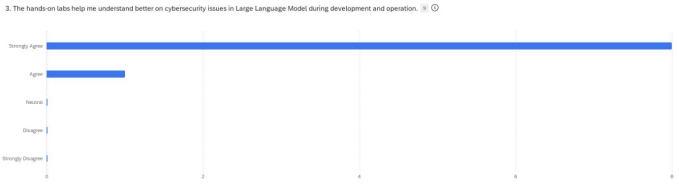
In our research, we concentrated exclusively on handling insecure LLM outputs within the context of web development. However, it is also essential to evaluate LLM responses across various use cases and further research is needed to thoroughly examine the results produced by LLM in other scopes. The LLM output needs thorough scrutiny in sectors such as Healthcare advice, legal and compliance, business and finance, public safety and emergency response, journalism and media, customer service and support, personal data and privacy, and Interactions with Children, etc. As we increasingly rely on AI-enabled services in our daily lives due to their promising results these days, it



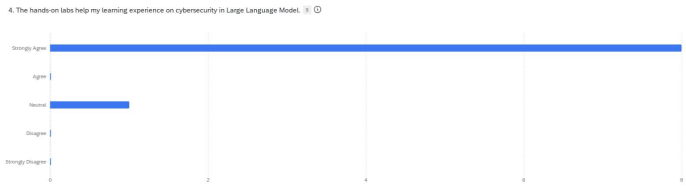
(a)

(b)

Fig. 3: Figure (a), and (b) displays the responses from the post-survey questions

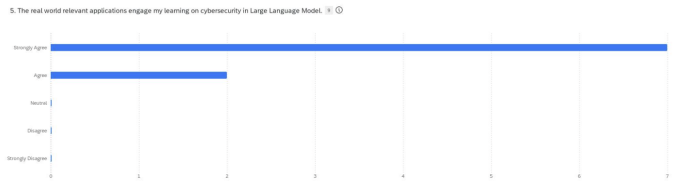


(a)

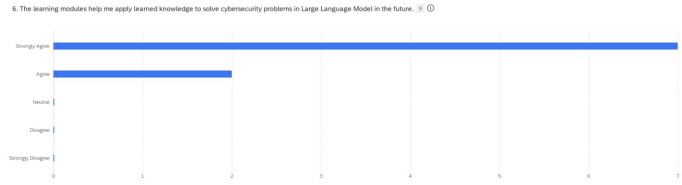


(b)

Fig. 4: Figure (a), and (b) displays the responses from the post-survey questions



(a)



(b)

Fig. 5: Figure (a), and (b) displays the responses from the post-survey questions

is also very crucial to carefully examine the outcome of LLM before making any decisions in these critical areas. Furthermore, we aim to disseminate our google site among various students, possibly more than 1500, to allow them to learn and get feedbacks from them and improve our lab based on the feedbacks.

VI. Conclusion

Our labware is designed to tackle the challenges associated with learning how to handle insecure outputs in Large Language Models (LLMs) effectively focused on software security through authentic learning. We have structured the labware progressively to guide students from basic to advanced levels. In order to overcome the gap between typical learning environments and engaging practical experiences, we have proposed an approach which provides both theoretical knowledge as well as practical coding-level training that are directly relevant to the authentic learning for the learners. The preliminary feedback from students has been very positive. They have not only been able to understand the concepts but have also actively applied these skills through the hands-on labs. This reinforces our belief that practical experience combined with theoretical understanding is crucial in educating students about insecure output handling of LLMs. Through this labware, we aim to equip students with the necessary skills to navigate and mitigate the vulnerabilities of LLM outputs in their future endeavors in the software development domain.

VII. Acknowledgments

The work is supported by the National Science Foundation under NSF Award #2100134, #2100115, #2310179, #2209637, #2421324 and #1946442 . Any opinions, findings, recommendations, expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] D. Glukhov, I. Shumailov, Y. Gal, N. Papernot, and V. Papyan, "Llm censorship: A machine learning challenge or a computer security problem?," *arXiv preprint arXiv:2307.10719*, 2023.

[2] M. M. Lombardi and D. G. Oblinger, "Authentic learning for the 21st century: An overview," *Educause learning initiative*, vol. 1, no. 2007, pp. 1–12, 2007.

[3] M. S. Akter, H. Shahriar, J. Rodriguez-Cardenas, M. M. Rahman, A. Rahman, and F. Wu, "Teaching devops security education with hands-on labware: Automated detection of security weakness in python," in *Proceedings of the ISCAP Conference ISSN*, vol. 2473, p. 4901, 2023.

[4] M. M. Rahman, A. S. Arshi, M. M. Hasan, S. F. Mishu, H. Shahriar, and F. Wu, "Security risk and attacks in ai: A survey of security and privacy," in *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1834–1839, IEEE, 2023.

[5] M. R. Mia, H. Shahriar, M. Valero, N. Sakib, B. Saha, M. A. Barek, M. J. H. Faruk, B. Goodman, R. A. Khan, and S. I.

Ahamed, "A comparative study on hipaa technical safeguards assessment of android mhealth applications," *Smart Health*, vol. 26, p. 100349, 2022.

[6] M. Dua and H. Singh, "Detection & prevention of website vulnerabilities: Current scenario and future trends," in *2017 2nd International Conference on Communication and Electronics Systems (ICCES)*, pp. 429–435, IEEE, 2017.

[7] P. S. Satish and R. Chavan, "Web browser security: different attacks detection and prevention techniques," *International Journal of Computer Applications*, vol. 170, no. 9, pp. 35–41, 2017.

[8] S. Gupta and B. B. Gupta, "Cross-site scripting (xss) attacks and defense mechanisms: classification and state-of-the-art," *International Journal of System Assurance Engineering and Management*, vol. 8, pp. 512–530, 2017.

[9] X. Lin, P. Zavarsky, R. Ruhl, and D. Lindskog, "Threat modeling for csrf attacks," in *2009 International Conference on Computational Science and Engineering*, vol. 3, pp. 486–491, IEEE, 2009.

[10] G. Pellegrino, O. Catakoglu, D. Balzarotti, and C. Rossow, "Uses and abuses of server-side requests," in *Research in Attacks, Intrusions, and Defenses: 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings 19*, pp. 393–414, Springer, 2016.

[11] B. Jabiyev, O. Mirzaei, A. Kharraz, and E. Kirda, "Preventing server-side request forgery attacks," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pp. 1626–1635, 2021.

[12] Z. S. Alwan and M. F. Younis, "Detection and prevention of sql injection attack: a survey," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 8, pp. 5–17, 2017.

[13] T. Calò and L. De Russis, "Leveraging large language models for end-user website generation," in *International Symposium on End User Development*, pp. 52–61, Springer, 2023.

[14] T. K. Le, S. Alimadadi, and S. Y. Ko, "A study of vulnerability repair in javascript programs with large language models," *arXiv preprint arXiv:2403.13193*, 2024.

[15] Z. Hong, Z. Yuan, H. Chen, Q. Zhang, F. Huang, and X. Huang, "Knowledge-to-sql: Enhancing sql generation with data expert llm," *arXiv preprint arXiv:2402.11517*, 2024.

[16] S. Oh, K. Lee, S. Park, D. Kim, and H. Kim, "Poisoned chatgpt finds work for idle hands: Exploring developers' coding practices with insecure suggestions from poisoned ai models," *arXiv preprint arXiv:2312.06227*, 2023.

[17] M. L. Siddiq and J. Santos, "Generate and pray: Using sallms to evaluate the security of llm generated code," *arXiv preprint arXiv:2311.00889*, 2023.

[18] J. He and M. Vechev, "Large language models for code: Security hardening and adversarial testing," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1865–1879, 2023.

[19] Y. Luo, Z. Lin, Y. Zhang, J. Sun, C. Lin, C. Xu, X. Su, Y. Shen, J. Guo, and Y. Gong, "Ensuring safe and high-quality outputs: A guideline library approach for language models," *arXiv preprint arXiv:2403.11838*, 2024.

[20] K. Jesse, T. Ahmed, P. T. Devanbu, and E. Morgan, "Large language models and simple, stupid bugs," in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, pp. 563–575, IEEE, 2023.

[21] T. Vörös, S. P. Bergeron, and K. Berlin, "Web content filtering through knowledge distillation of large language models," in *2023 IEEE International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pp. 357–361, IEEE, 2023.