Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.5521

EXSS: an Educational Emulator for Cross-Site Scripting Attacks

Bianca Guarizi [**Centro Federal de Educação Tecnológica Celso Suckow da Fonseca** | ***bianca.guarizi@aluno.cefet-rj.br***]
Isabela Alves [**Centro Federal de Educação Tecnológica Celso Suckow da Fonseca** | ***isabela.alves@aluno.cefet-rj.br***]
Júlia Fernandez e Souza [**Centro Federal de Educação Tecnológica Celso Suckow da Fonseca** | ***julia.fernandez@aluno.cefet-rj.br***]
Guilherme Pimentel [**Universidade Federal Fluminense** | ***guilherme_pimentel@id.uff.br***]
João André Watanabe [**Universidade Federal Fluminense** | ***jwatanabe@id.uff.br***]
Dalbert Mascarenhas [**Centro Federal de Educação Tecnológica Celso Suckow da Fonseca** | ***dalbert.mascarenhas@cefet-rj.br***]
Ian Bastos [**Universidade do Estado do Rio de Janeiro** | ***ian.bastos@eng.uerj.br***]
Marcelo Rubinstein [**Universidade do Estado do Rio de Janeiro** | ***rubi@uerj.br***]
Igor Moraes [**Universidade Federal Fluminense** | ***igor@ic.uff.br***]

Institute of Computing, Universidade Federal Fluminense, Av. Gal. Milton Tavares de Souza, s/n, São Domingos, Niterói, RJ, 24210-590, Brazil.

Received: 15 February 2025 • **Accepted:** 21 November 2025 • **Published:** 04 May 2026

Abstract This article proposes a Cross-Site Scripting (XSS) attack emulator for learning in cybersecurity. The emulator allows users to identify a website vulnerable to XSS attacks in a controlled environment. The identification of vulnerabilities is achieved through activities that consist of a theoretical introduction to the topic, followed by practical procedures for conducting XSS vulnerability tests on a Web server running on a virtual machine. Activities are developed for different levels of knowledge. The particularity of the proposed emulator is its educational approach and its goal is to raise awareness among undergraduate students and professionals to develop less vulnerable websites.

Keywords: Cybersecurity, XSS Attacks, Emulation, Education

1 Introduction

Internet users have become increasingly dependent on services offered through Web applications, such as e-commerce, Internet Banking, hotel reservations, and online payments. A report produced by CyCognito (CyCognito [2023]) shows that 70% of Web applications are developed with severe security gaps and 30% are vulnerable to one of the 10 most common attack categories identified by the Open Worldwide Application Security Project (OWASP) (Grossman [2007]; Gupta and Gupta [2017]; Rodríguez *et al*. [2020]; Liu *et al*. [2019]). Cross-Site Scripting (XSS) is one of the most common attacks, and its first identification dates back to 1996 (Grossman [2007]), when the JavaScript programming language emerged. XSS attacks belong to the category of attacks known as code injection attacks. In such attacks, a malicious user injects malicious code through the input fields available in the system. In the case of XSS attacks, the malicious user exploits the input fields of the Web page to inject malicious code, usually JavaScript, into a legitimate Web application because the application does not correctly encode or validate the user input data (Kaur *et al*. [2023]).

XSS is one of the leading causes of data theft and breaches in large organizations. In 2018, British Airways suffered the theft of approximately 380,000 records because of an XSS vulnerability in its payment module (Reuters [2018]; BBC [2018]). In 2019, a popular video game developed by Epic Games called Fortnite was a potential target of XSS attacks because of a vulnerability found on the Web page of one of the company's subdomains (Latest Cyber Security News [2019]). Although the company had approximately 350 million registered users in 2020, there are no records of credential theft because of the XSS vulnerability. In 2020, the event organization platform Meetup may have had some of the funds raised through the PayPal platform transferred to malicious users (Latest Cyber Security News [2020]). The discussion message area, enabled by default on the platform, allowed the persistent storage of malicious JavaScript code.

XSS attacks can be classified into three categories: (i) reflected or non-persistent XSS, (ii) stored or persistent XSS, and (iii) DOM (Document Object Model) based XSS. In reflected XSS attacks, the attacker generates a Uniform Resource Locator (URL) containing malicious code as part of the HTTP (HyperText Transfer Protocol) request. The attacker inserts malicious codes into input fields of the Web application, such as search bars, comment sections, and authentication text boxes. When the legitimate user receives a URL from the served Web page and executes it in the Web browser, the user carries out the malicious code present in the URL sent by the attacker. In stored XSS, the attacker generates malicious code and persistently stores it on a vulnerable server, typically through a database. We can find an example of this type of attack in Web applications that serve as discussion forums or social networks. A stored XSS attack

does not require a URL and can affect many users, as any user browsing the Web application may execute the malicious code served by the compromised page. In DOM-based XSS, the attacker uses a URL to insert malicious code. However, in this attack, the URL dynamically modifies the structure of the objects that build the HTML (HyperText Markup Language) page to insert two new structures, a source and a sink. The modified object model in the HTML page only occurs when the legitimate user's Web browser processes the page, with the HTML page server being unaware of the modification. With the altered Web page, the inserted source element retrieves sensitive information from the legitimate user and directs it to the sink element, where the attacker collects the sensitive information.

This article proposes an XSS attack emulator to promote awareness and practical learning in cybersecurity[1]. The proposed emulator, EXSS, allows users to identify a website vulnerable to XSS attacks in a controlled environment. The identification of vulnerabilities occurs through activities performed by users, as the primary goal of EXSS is to provide an educational approach to handling XSS attacks. The activities consist of a theoretical introduction to the topic of the activity, followed by practical procedures for performing XSS vulnerability tests on a Web server running on a Virtual Machine (VM). The laboratory environments emulate a small e-commerce site containing vulnerabilities the user must exploit. The EXSS users check for vulnerabilities by conducting insertion tests of characters and scripts. The emulator guides the user step-by-step during the execution of these tests. We define activities for different levels of knowledge, addressing the three types of XSS attacks. We develop the EXSS within the context of the "Hackers do Bem" Working Groups program.

Similar solutions, such as PortSwigger (PortSwigger [2024b]), Google XSS Game (Google [2024]), OWASP Juice Shop (OWASP [2024]), OWASP WebGoat (OWASP [2023]), and TryHackMe (TryHackMe [2024]) are provided. Section 4 describes those frameworks. The distinguishing features of our solution are that it is completely free, runs without Internet access, and supports Brazilian Portuguese.

We organize the remainder of this article as follows. Section 2 presents the architecture of the proposed emulator, mentioning the chosen technologies for implementing the architecture modules. Section 3 describes the EXSS functionalities. Section 4 describes tools similar to ours. Section 5 concludes the article and highlights the implications and strategic recommendations for the emulator.

2 The EXSS emulator architecture

The EXSS emulator has four modules, as Figure 1 illustrates: User Interface, Activities Catalog, Vulnerability Analysis, and Technical Report.

The User Interface module is a user-friendly Web interface focusing on usability and accessibility. The interface

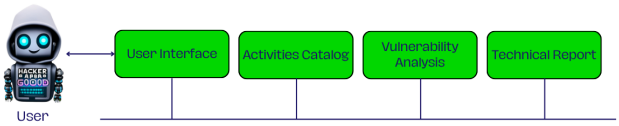

Figure 1. A block diagram from the EXSS modules.

is intuitive and aims to serve users of different skill levels, offering clear guidance and explanatory resources. It has enhanced usability through user-centered design techniques and incorporates gamification. We provide different Web pages depending on the activity context. The graphical interface includes a main page where users select their level. Additionally, an expandable sidebar menu facilitates user navigation through all the proposed activities.

The Activities Catalog module is responsible for defining the activities. Each activity consists of different tasks: a theoretical introduction to the activity topic, procedures for performing XSS vulnerability experiments, and a quiz test to assess the user's learning of the activity topic. This module communicates with the User Interface module to present the list of available activities and load the activity, as we show in the Figure 2. The Activities Catalog module identifies specific XSS vulnerabilities by querying an internal database and dynamically executes corresponding activities based on the user progress. The EXSS emulator provides three activities that address the most common types of XSS attacks with the third activity including good practices training in secure development. The activities are: 1: Reflected XSS, 2: Stored XSS, and 3: DOM-based XSS.

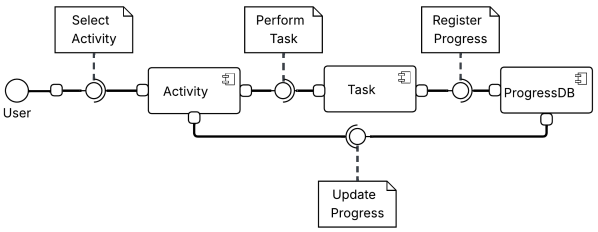

Figure 2. Components of the Activities Catalog module.

It is important to emphasize that each activity provides a theoretical foundation for understanding the different types of XSS attacks, followed by laboratory-format tasks. We guide the users in applying the theoretical concepts through tutorials, aiming to facilitate the resolution of the practical task. To perform each activity, the user needs a required knowledge level. The activities are spread across three levels: beginner, intermediate, and advanced. At the beginner level, the goal is to familiarize users with the EXSS emulator and the concepts of the different types of XSS attacks. Users with intermediate and advanced knowledge can experiment with scripts that exploit XSS vulnerabilities. Advanced users can perform mitigation techniques directly to the Web page's code to observe how the techniques work on the XSS vulnerability. Figure 3 shows a typical flow for a user to perform an activity. The practical application of attacking the Web pages and mitigating the attacks provides fundamental knowledge for real-world cybersecurity scenarios.

The Vulnerability Analysis module contains the emulator core, as it is responsible for the practical tasks. This mod-

[1]This work is based on our paper published in Portuguese in the Proceedings of the VIII Salão de Ferramentas (SF) available at `https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/30121/29929`

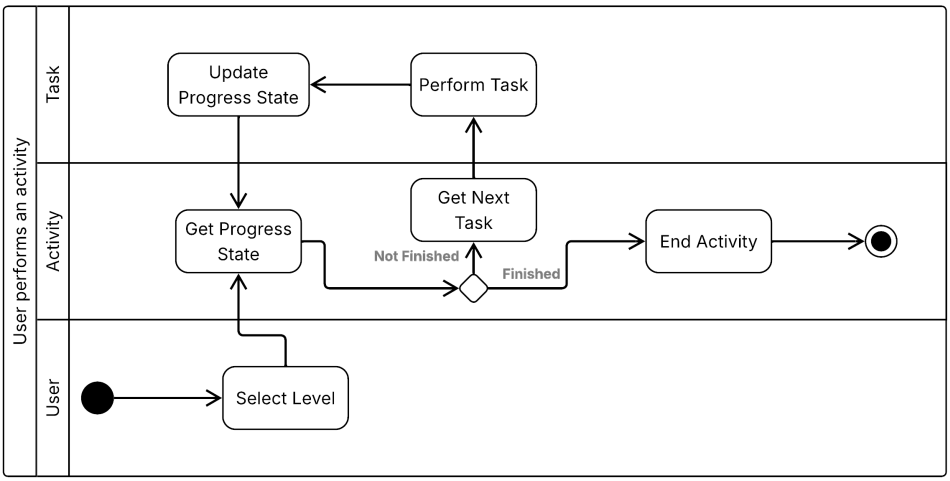

Figure 3. The flow to perform an activity in EXSS.

ule provides Web hosting services, using Apache 2 as the underlying Web server technology. The emulator integrates the environments for the activities into a small e-commerce explicitly developed for this emulator, as shown in Figure 4.

The Technical Report module has a gamified approach to improve user learning. Users receive points upon completion of tasks. The emulator incorporates other gamification elements, such as awarding medals, to enhance user engagement. Users earn one medal upon finishing one level. The immediate feedback and the gamification elements provide an interactive and didactic learning experience.

2.1 Deployment Model

A key goal for our tool is to provide a self-contained, reproducible, and easy-to-use environment for learning and training. To achieve this goal, we opted to provide the environment via a VM. The VM offers complete isolation, encapsulating the operating system, all required libraries, and preconfigured dependencies. This method ensures that the user's host system remains unaltered and guarantees that the tool runs as intended, regardless of the user's local configuration. Furthermore, a significant advantage of the VM is its ability to operate entirely offline after the initial download, which is beneficial for environments with restricted or no Internet access. Therefore, EXSS is a monolithic server-side application packaged within a VM. This approach simplifies deployment and guarantees a consistent operating environment, avoiding the setup complexities of distributed architectures, such as microservices, which would be counterproductive in an educational context.

Despite its pros, the VM also has its cons. The primary drawback is the substantial file size of the VM image, which can make the download process challenging and time-consuming, particularly for users with slower Internet connections. Additionally, VMs incur higher resource overhead in terms of CPU and RAM compared to more lightweight virtualization technologies.

In this context, containerization, particularly using Docker, presents a compelling alternative that we are actively exploring. Containers offer a more lightweight solution, with significantly smaller image sizes and faster start-up times. They provide process-level isolation while sharing the host's kernel, leading to lower resource consumption. A container-based distribution would simplify the download and setup process and promote integration with modern cloud-based research workflows and Continuous Integration/Continuous Delivery (CI/CD) pipelines.

Despite these advantages, the initial choice of a VM was deliberate to maximize out-of-the-box usability for a broad audience, some of whom may be less familiar with container command-line interfaces. Future work will focus on providing a parallel distribution based on Docker containers. This dual approach will consider different user needs: the stability and simplicity of the VM for initial exploration, and the efficiency and scalability of a container for advanced use cases and cloud deployment.

EXSS operates as a client-server model. The four modules described in Section 2 represent a logical separation of concerns within the PHP codebase. The communication protocol between the user's browser and the server relies on standard HTTP GET/POST requests. For instance, when a user submits a payload, the Vulnerability Analysis module processes the raw HTTP request to emulate the vulnerability.

The persistence layer is managed by MySQL and has tables for progress tracking and pedagogical content. Nonetheless, it is a critical layer to implement the Stored XSS labs. The users' input are intentionally stored in raw format, without server-side sanitization upon insertion. This decision is fundamental to the emulator's purpose, as it ensures that stored malicious payloads remain active and executable when retrieved, thus allowing students to observe the mechanics of a persistent XSS attack in a controlled manner.

3 EXSS Emulator Features

From the Client-side perspective, EXSS offers a user interface composed of an emulated e-commerce environment developed using a combination of HTML, CSS3, and JavaScript (ES6+). HTML provides the semantic structure for the Web pages, defining the Document Object Model (DOM) that is the primary target of XSS attacks. CSS3, implemented via the Bootstrap framework, ensures a responsive and visually con-

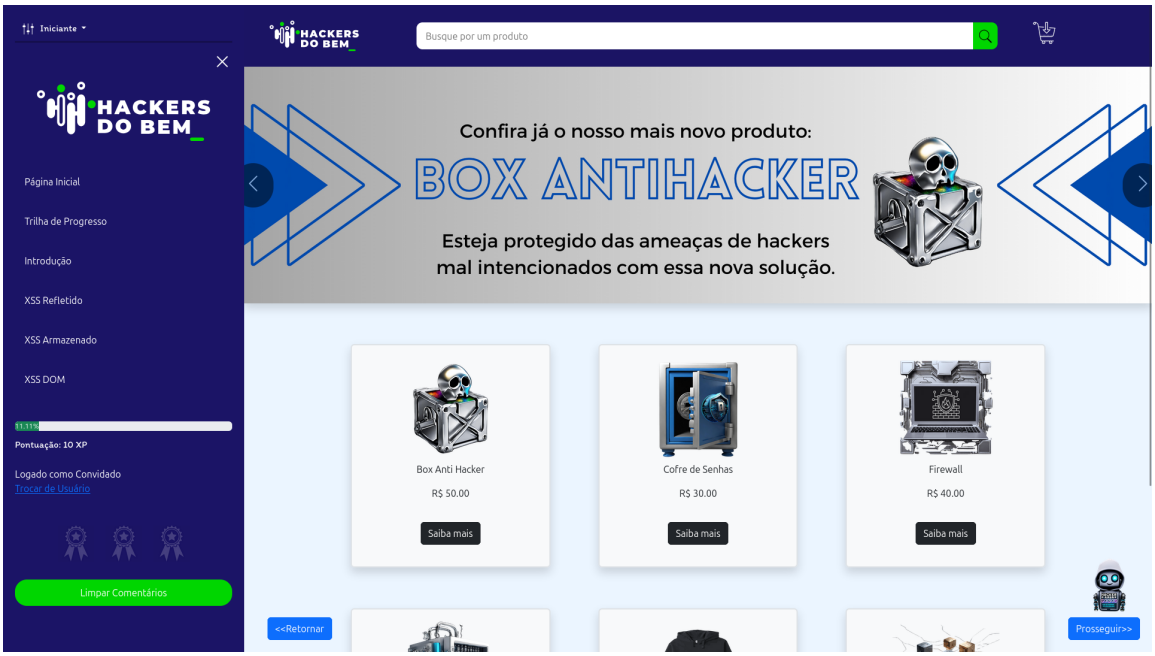

Figure 4. The developed EXSS e-commerce.

sistent user interface, which is critical for user engagement. EXSS uses JavaScript and JQuery to implement the vulnerable client-side logic in the laboratory scenarios (e.g., insecure DOM manipulation with ".innerHTML") and to power the emulator's interactive features, such as the step-by-step guidance and gamification elements.

In the Server-side, the backend logic is implemented in PHP and is responsible for managing user sessions, activities, and serving the vulnerable Web pages. PHP is widely adopted in Web development and makes the emulated vulnerabilities representative of real-world scenarios. It handles HTTP requests and interacts with the database. A MySQL database is used for data persistence, storing user progress and, in the case of Stored XSS labs, the malicious payloads. The entire stack is served by an Apache 2 Web Server, which is also widely adopted in real-world scenarios.

The main functionality of the EXSS emulator is to carry out activities for different types of XSS attacks, considering various user knowledge levels. Figure 5 illustrates the emulator's main page after logging in. The main page greets the user with Hacker Good, an avatar developed for the emulator that assists the users throughout their activities. The user can select the activity level they will engage in. On the left side of the page, the user can find a level indicator, a menu to access tasks for each type of XSS attack, and a progress bar for the tasks.

The user sees a page with explanatory documentation by clicking the button corresponding to the desired activity. Furthermore, the user has the option to follow the progress of the level tasks using the "Trilha de Progresso" ("Progress Trail") as shown in Figure 6. This trail represents the last task the user already performed by the "Hacker Good" position, all tasks already performed by the green token, the task the user is currently performing by the orange token, and the remaining level tasks by the gray tokens. The Web page also shows in the top-left corner the user's relative progress in the percentage progress bar and in the top-middle the user's current level. For each level, each task performed contributes equally to the percentage progress, and this progress reaches 100% when the user finishes all tasks of the corresponding level.

The beginner level serves as an entry point for first-time users. Users at this level study the XSS vulnerability, and the EXSS exposes the importance of Web security for users who are new to this type of vulnerability. Furthermore, the beginner level provides a comprehensive definition of XSS attacks, categorizing and explaining the various XSS types in a didactic manner. To proceed to the next task, the user must click the "Prosseguir" ("Continue") button.

In Activity 1, we present Reflected XSS in detail, as this type of XSS is easier for a beginner user to understand. Then the Hacker Good explains, step by step, what the user needs to perform. Figure 7 illustrates a step-by-step screen where Hacker Good "speaks", explains, and presents the emulated environment of the e-commerce site. The user can move forward or backward by clicking on the available buttons. Thereafter, the user performs the laboratory tasks. These tasks include introductory texts followed by demonstrations of the laboratory executions, as we assume the user has not performed any XSS attack before. In the case of Laboratory 1, the user has to insert a script that generates an alert message presented as a pop-up into a product search field. The result of the attack is shown in Figure 8. To provide a clearer methodological structure, we detail Laboratory 1 as follows:

- **Objective:** To demonstrate a classic Reflected XSS attack where non-sanitized user input is executed by the browser.
- **Setup:** The laboratory presents a product search page. The backend logic (emulated via PHP) is configured to take the query parameter from the URL and directly embed it into the HTML of the results page without any output encoding.
- **Procedure**: The user is guided to input the payload

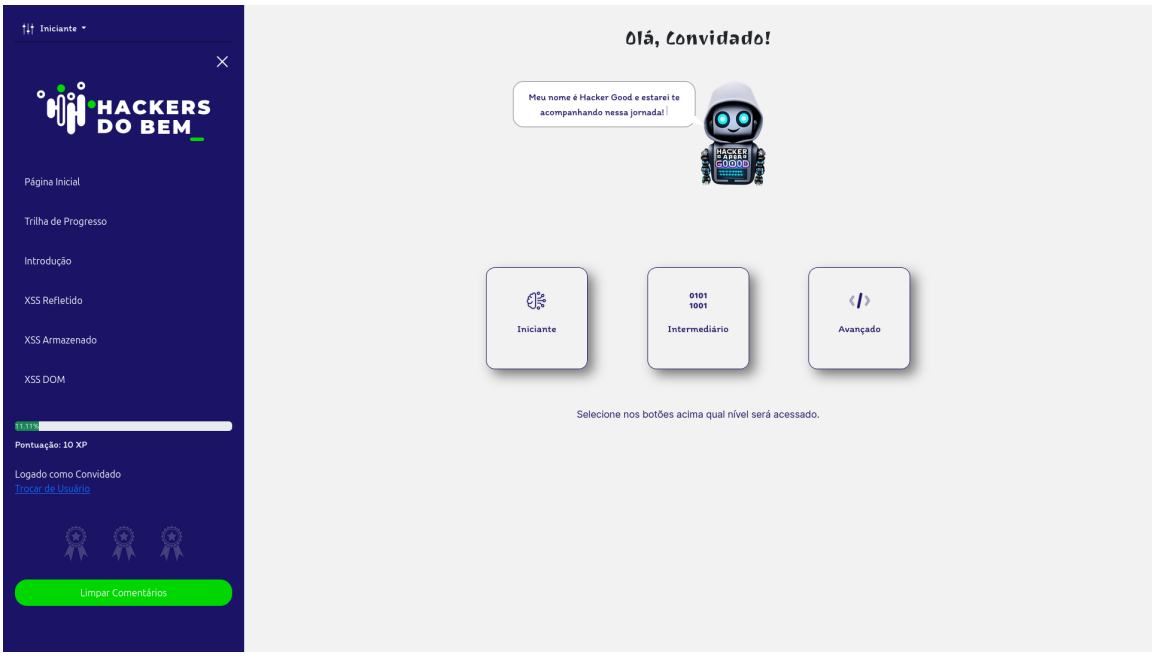

Figure 5. The EXSS's main page.

Figure 6. The Progress Trail page.

`<script>alert('XSS')</script>` into the search field. Upon submission, the application generates a URL such as `/search.php?query=<script>alert('XSS')</script>` and reflects the payload back into the page.

- **Expected Outcome:** The browser parses the reflected HTML, encounters the `<script>` tag, and executes the contained code, resulting in the JavaScript `alert()` dialog box shown in Figure 8.
- **Pedagogical Analysis:** This outcome provides tangible proof of the vulnerability. The emulator reinforces the learning by visually highlighting how unescaped characters (<, >) can break out of the data context and be interpreted as executable code by the browser.

A second laboratory involves the redirection of a Web page as the attack effect. The user inserts another script into the product search field. At last, simple multiple-choice questions are provided to verify the user's learning. Similar tasks are provided for the other two types of XSS attacks.

The intermediate level starts with a summary of the level. For the Reflected XSS activity, the emulator introduces the HTTP protocol, including its message format. Then the practical tasks are presented. As shown in Figure 9, the first laboratory involves an electronic voting process that simulates a competition between two products to receive a discount coupon. The user manipulates the voting in favor of their favorite product "Box Anti Hacker" ("Anti Hacker Box") by inserting a script into the "Buscar Comentário" ("Search for Comment") field. When the user votes for the "Cofre de

Figure 7. A step-by-step explanatory page regarding Reflected XSS.

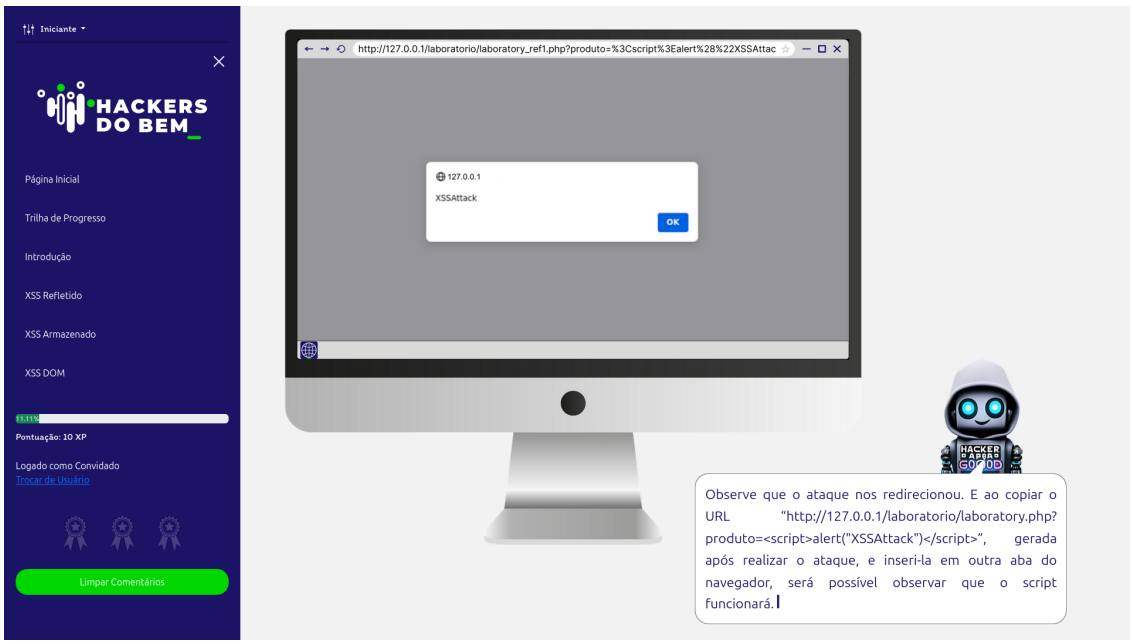

Figure 8. A feedback result from a Reflected XSS laboratory.

Senhas" ("Password Safe") product, the script counts the vote for both products, ensuring a scoring advantage for the Anti Hacker Box product. We have also implemented the voting process laboratory considering the other two types of XSS attacks.

Figure 10 shows the cookie theft laboratory for the Reflected XSS. The emulated e-commerce environment contains a vulnerability in the "Buscar CEP" ("Search ZIP Code") field within the checkout process. This field serves as an attack vector. The task aims to demonstrate how a malicious user can inject scripts into vulnerable fields to capture and redirect the session cookie of another user completing a purchase.

The cookie theft laboratory highlights the real-world implications of XSS vulnerabilities, demonstrating how attackers can obtain unauthorized access to user accounts and potentially steal sensitive data. Users understand the compromising risks of XSS and the importance of implementing security measures when performing the task. This laboratory is also implemented for the Stored XSS.

The cookie theft laboratory, a more advanced scenario, is structured as follows:

- **Objective:** To demonstrate how an XSS vulnerability can be escalated to achieve session hijacking by stealing a user's session cookie.
- **Setup:** The scenario uses a vulnerable "Search ZIP Code" field during a checkout process. A simple listener (emulated attacker server) is provided to display any data sent to it.
- **Procedure:** The user crafts a payload designed to read the browser's cookie and exfiltrate it. A typical payload

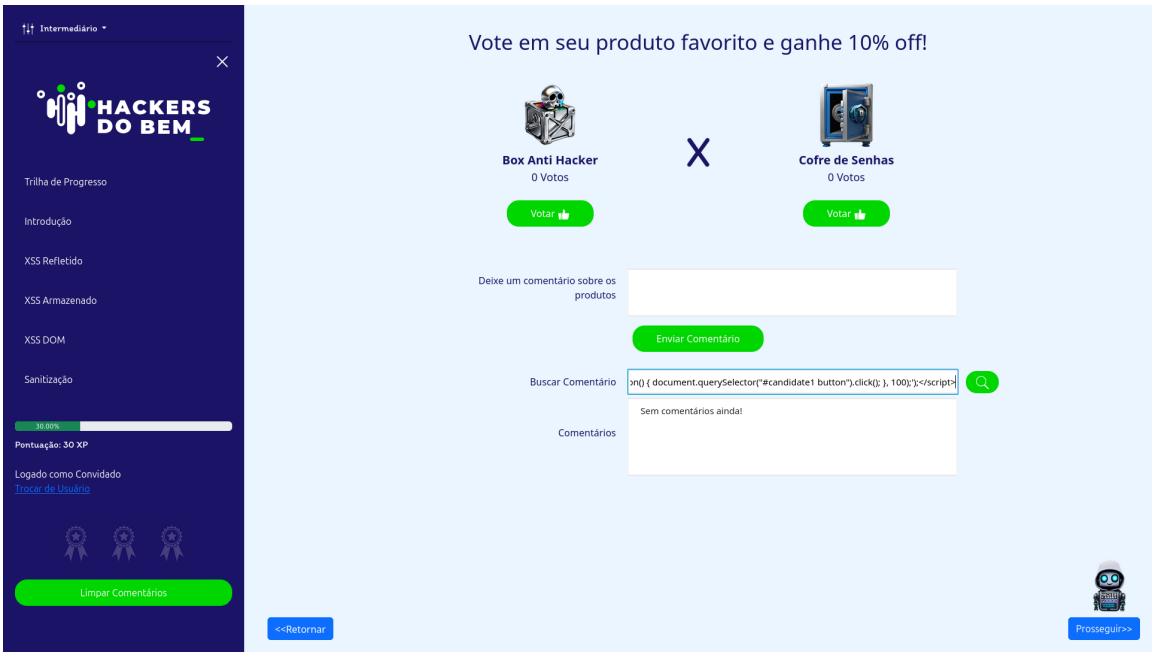

Figure 9. Electronic voting laboratory page for Reflected XSS.

is `<script>document.location= 'http://attacker.com/log?cookie=' + document.cookie</script>`. This script, when injected, redirects the user's browser to the attacker's server, appending the session cookie to the URL.

- **Expected Outcome:** The user's session cookie is captured by the attacker's server, demonstrating a successful session hijacking. The emulator shows the captured cookie in a simulated attacker terminal.
- **Pedagogical Analysis:** This lab illustrates the direct security impact of XSS, moving beyond a simple alert box to a tangible data breach. It highlights the critical importance of setting the `HttpOnly` flag on session cookies as a defense-in-depth mechanism.

At the end of the Reflected XSS activity, another questionnaire is applied.

The intermediate level also deals with sanitization, one of the main ways to prevent XSS attacks. This activity includes introductory material and a sanitized version of the voting laboratory, in which the same attack described before is not successful. Two versions of the laboratory are implemented for Reflected and DOM-based XSS.

The user can learn about Graphical User Interface (GUI), HTML, and element inspection from the Reflected XSS activity at the advanced level. EXSS then provides an inspection laboratory where users explore a session hijacking attack (Figure 11). EXSS challenges the users to craft a malicious URL containing a JavaScript code. The code aims to exploit a Reflected XSS vulnerability within the simulated EXSS Web page. When the user clicks on the maliciously crafted URL, the JavaScript code executes in the user's Web browser, giving access to the "document.cookie" object. This object contains the user's session cookie, which could allow a malicious user to impersonate the user.

The inspection laboratory emphasizes the technical details of Reflected XSS attacks and their impact on session management. By successfully stealing the session cookie, users understand how attackers can bypass authentication mechanisms and compromise user accounts. This laboratory reinforces the importance of input validation, output encoding, and proper HTTP header configuration to mitigate XSS vulnerabilities. After the laboratory, the user answers a questionnaire.

The Stored XSS at the advanced level initially introduces JavaScript, DOM, and PHP. After that, the user can perform an emulated keylogger attack laboratory designed to provide a practical experience of vulnerabilities related to sensitive information theft (Figure 12). This laboratory emulates a realistic e-commerce scenario where users input credit card data into sensitive fields. The injected malicious code captures the user's input, demonstrating how attackers can steal sensitive information. At the end of the laboratory, the user understands the importance of secure coding practices in Web application development and the potential consequences of XSS vulnerabilities. At last, the user answers a few questions.

At the end of the advanced level, EXSS provides a sanitization activity to teach users effective protection methods against XSS attacks, including information about CSP (Content Security Policy) (West and Sartori [2025]) and the DOMPurify library (DOMPurify [2025]). Then another questionnaire is applied, followed by a dedicated laboratory environment (Figure 13). The sanitization laboratory focuses on practical mitigation techniques, including input sanitization, output encoding, and CSP. The sanitization laboratory guides the user through tasks demonstrating how these techniques can prevent XSS vulnerabilities and protect Web applications. More specifically, the user can try different solutions to sanitize a Web page, which is rendered when the user clicks a button. EXSS provides the user with an attack and defense emulation training approach to comprehensively understand XSS vulnerabilities and their mitigation.

Listing 1: A PHP function for output encoding.

```
<?php
```

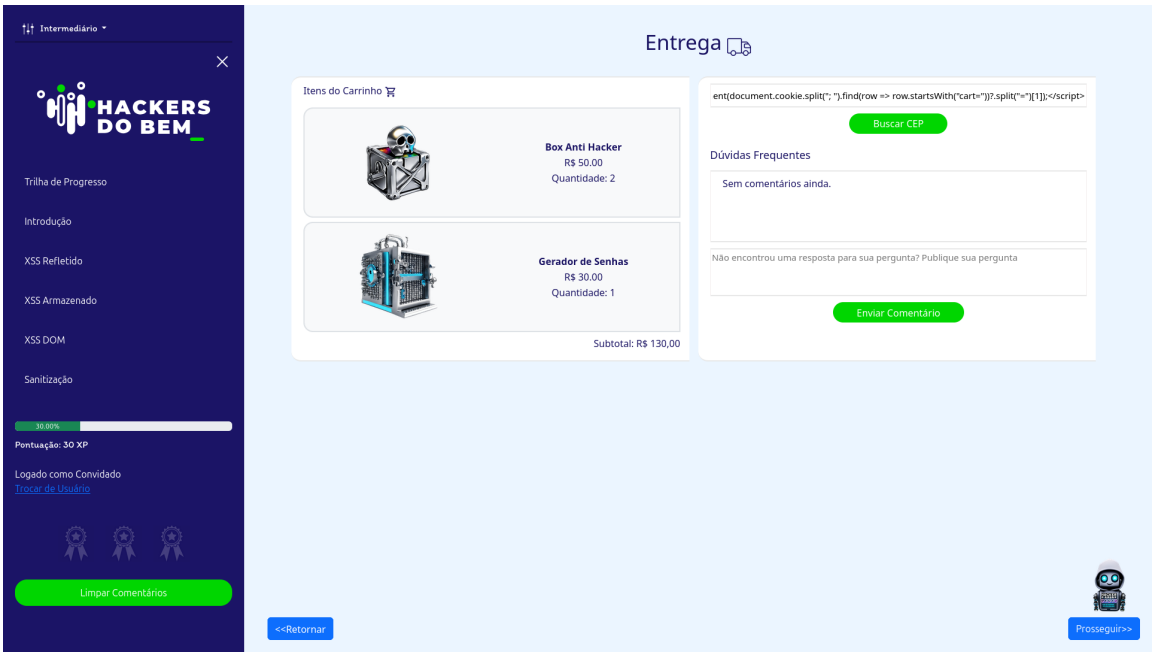

Figure 10. Cookie theft laboratory page for the Reflected XSS.

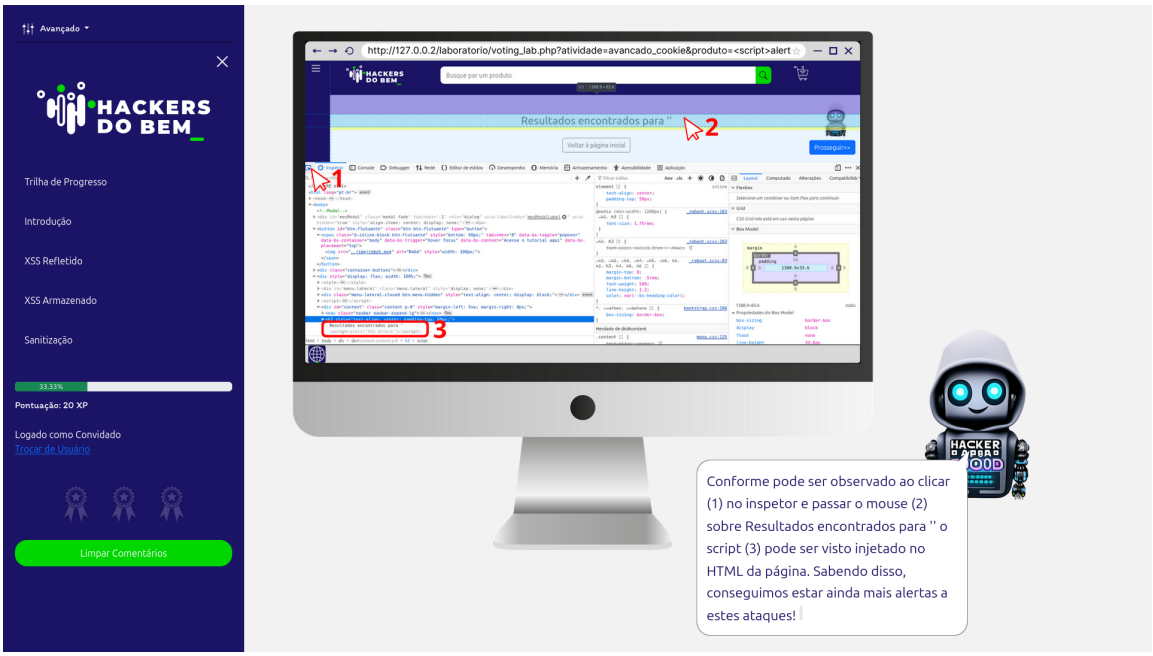

Figure 11. Inspection laboratory page for the Reflected XSS.

```php
/**
 * Sanitizes user input to be safely rendered in HTML.
 * This prevents XSS by converting special HTML
 * characters into their corresponding entities.
 *
 * @param string $input The raw user input.
 * @return string The sanitized string.
 */
function sanitize_output($input) {
  // Convert special characters using htmlspecialchars.
  // ENT_QUOTES converts both double and single quotes.
  // UTF-8 ensures correct character encoding.
  return htmlspecialchars(
    $input,
    ENT_QUOTES,
    'UTF-8'
  );
}

// --- Example Usage ---

// Unsafe:
// echo "<h4>Search results for: " . $_GET['query'] .
     "</h4>";

// Safe:
$safe_query = sanitize_output($_GET['query']);
echo "<h4>Search results for: " . $safe_query . "</h4>";
?>
```

To illustrate a fundamental server-side defense mechanism taught within the emulator, Listing 1 presents a simple PHP function for output encoding. This technique ensures that user-supplied data is treated as literal text rather than executable code, effectively neutralizing XSS payloads.

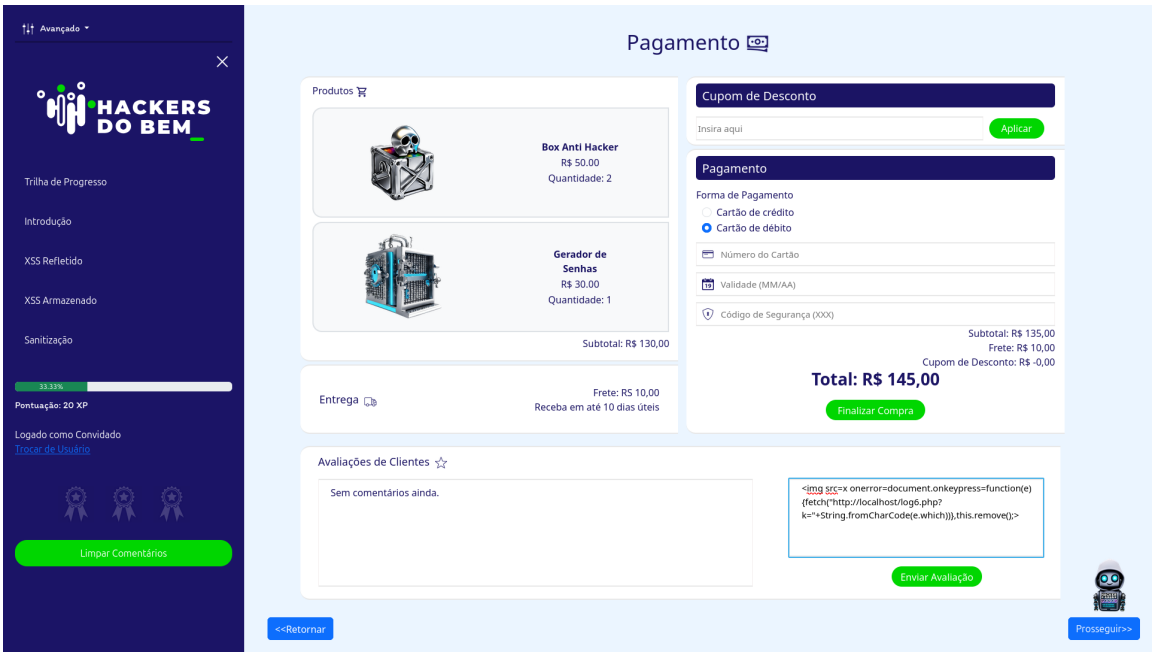

Figure 12. KeyLogger laboratory page for the Stored XSS.

Figure 13. Sanitization laboratory page.

4 Related Frameworks

We can find frameworks with the same purpose as the EXSS emulator in the literature (PortSwigger [2024b]; Google [2024]; OWASP [2024, 2023]; TryHackMe [2024]). PortSwigger, through its Web Security Academy (PortSwigger [2024b]), provides a series of laboratories that involve performing different types of attacks, including XSS. The laboratories provide generic texts covering basic concepts such as exploiting vulnerabilities and techniques to prevent attacks. Users can access the activity through an Internet connection and a simulated Web page containing the associated vulnerability. The PortSwigger framework does not provide material for performing the activity, except for a brief description and the solution. The dependency on the paid version of the Burp Suite tool to perform some activities is PortSwigger's major drawback (PortSwigger [2024a]).

Google XSS Game (Google [2024]) contains six activities that involve performing different types of online XSS attacks on vulnerable Web pages. In each activity, the framework provides the description, the objective, the page source code, and up to three hints for solving the activity. The user has to click on links to show the page source code and the hints. The framework does not offer more comprehensive support material to facilitate the completion of the activities. The absence of supporting material makes executing the last activities challenging for those with no previous background in XSS. Moreover, the user interface is not very attractive and there are no scoring systems or awarding medals to engage users.

Table 1. Comparison of related work.

Solution	Open-source	Tasks	Technical Feedback	PT-BR support	Ease of Use	Offline
PortSwigger	No	Theoretical and practical	Only laboratory solutions	No	Payed auxiliary tools	No
Google XSS Game	No	Practical	Tips for activity resolution	No	Provided Web page	No
OWASP Juice Shop	Yes	Practical	Score and reward system	No	Docker containers	No
OWASP Webgoat	Yes	Theoretical and practical	Tips to mitigate vulnerabilities	Yes	Docker containers	No
TryHackMe	No	Theoretical and practical	Allows environments to follow user progress	No	Environments with VMs	No
EXSS	Yes	Theoretical and practical	Constructive reports to mitigate vulnerabilities	Yes	Environments with VMs	Yes

The OWASP Juice Shop (OWASP [2024]) is an application where the users can explore vulnerabilities from the OWASP Top Ten (OWASP [2021]) and other real-world application security flaws, including XSS. The pedagogical model of the OWASP Juice Shop is effective for hands-on training in vulnerability exploitation but presents limitations for a comprehensive education on XSS. Its primary focus is gamification and does not provide a strong theoretical foundation for XSS vulnerabilities. The platform offers limited guidance on preventive measures and good practices for secure Web development. Furthermore, the framework requires Windows, Mac, or Linux installation using node.js, Docker, or Vagrant, which may pose a challenge for inexperienced users. Therefore, while the Juice Shop provides foundations for developing practical skills, it necessitates supplementary tools for a holistic educational experience.

OWASP WebGoat (OWASP [2023]) provides another intentionally insecure application designed for performing attacks commonly found in Java-based applications that use popular and common open-source components, including XSS attacks. Users interact with the system through the Web browser interface with a local host via containers to perform activities and learn about different vulnerabilities. The lessons follow a simple teaching pattern: a theoretical explanation of the content followed by laboratory practice, and at the end, there is a short questionnaire. The framework offers support in Portuguese. However, during the execution of the application, the user machine becomes vulnerable to attacks, and OWASP recommends that it be disconnected from the Internet.

Finally, TryHackMe (TryHackMe [2024]) is an online platform focused on learning and development in cybersecurity, including the study of XSS. The platform includes reading and assessment materials, challenges, and laboratories. The platform employs a gamified approach to track the user progress, awarding badges as rewards. The activities are carried out in classroom libraries, which are virtual machines configured with simulated scenarios of real vulnerabilities so that users can test their knowledge. The platform restricts part of its content to subscribed users.

Table 1 summarizes and compares the EXSS emulator with the five previously presented frameworks. The user can perform all EXSS activities free of charge. EXSS specifically addresses XSS vulnerabilities, aiming to provide the user with training from basic to advanced activities. EXSS employs an educational approach to teach about XSS attacks, including a theoretical introduction to the topic of the activity, followed by practical procedures, primarily guiding the beginner user. The users perform the tasks in a controlled environment with the Web server running on a local virtual machine, with no need of Internet access during the activity. Additionally, EXSS supports the Brazilian Portuguese language (PT-BR), which is an important feature for many young students in high school or undergraduate programs in Brazil.

5 Conclusion and Future Work

This article presents the EXSS emulator to teach about XSS attacks. The proposed emulator presents the main concepts of Web applications and allows users to identify websites vulnerable to XSS attacks in a controlled environment. EXSS activities provide a theoretical introduction and experimental laboratories that cover the main types of XSS attacks (reflected, stored, and DOM-based) and consider users with different knowledge levels. The key features of EXSS are that it is entirely free, operates without Internet access, and supports the Brazilian Portuguese language. The proposed emulator implements beginner-, intermediate- and advanced-level tasks. Nonetheless, EXSS has tasks at the advanced level to make users aware of the importance of secure development.

Next steps include a study on the use of Docker containers. As detailed in Section 2.1, our future work will address the primary limitations of the VM distribution — large download size and high resource overhead. We will develop a Docker container version to provide a lightweight, scalable, and efficient alternative for users. By incorporating new lab activities and backend scenarios, such as insecure SQL queries, unsafe command executions, and outdated libraries, the emulator can maintain a gamified learning approach while broadening its educational scope. Furthermore, the planned container-based and cloud-hosted distributions will facilitate the deployment of these new activities, enhancing accessibility and scalability

for users and instructors.

Moreover, we intend to make EXSS available in the cloud to provide a global ranking for a more gamified experience. This ranking can include different metrics, such as the average time to exploit the vulnerabilities and the user hit ratio. Our target audience can be extended to people who work for RNP (the Brazilian network for education and research) clients, such as higher education and research institutions, museums and cultural institutes, healthcare facilities, and development agencies.

Declarations

Authors' Contributions

All the authors have contributed to the conception of this study. BG, IA, JFS, GP, JAW, and DM have implemented the emulator. IB, MR, and IM are the main writers of this manuscript. All authors have read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

We would like to thank all the users who tested our emulator providing meaningful feedback.

Funding

This research has been funded in part by RNP, CNPq, CEFET/RJ, CAPES, FAPERJ, and PGC/UFF.

Availability of data and materials

The EXSS emulator, its installation and usage documentation, and the configuration and demonstration videos are available on the working group landing page[2]. Alternatively, we provide separately the virtual machine image with the installed emulator[3], the configuration and demonstration videos[4], and the documentation and source code[5].

References

BBC (2018). British airways faces record £183m fine for data breach. Available at:https://www.bbc.com/news/business-48905907 (01/17/2025).

CyCognito (2023). Web apps are leaving pii exposed state of external exposure management report. Technical report, CyCognito.

DOMPurify (2025). DOMPurify. Available at:https://github.com/cure53/DOMPurify (05/14/2025).

Google (2024). Xss game. Available at:https://xss-game.appspot.com/ (01/17/2025).

Grossman, J. (2007). *XSS Attacks: Cross Site Scripting Exploits and Defense*. Syngress. Book.

Gupta, S. and Gupta, B. B. (2017). Cross-site scripting (XSS) attacks and defense mechanisms: Classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, 8:512–530. DOI: 10.1007/s13198-015-0376-0.

Kaur, J., Garg, U., and Bathla, G. (2023). Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review. *Artificial Intelligence Review*, 56(11):12725–12769. DOI: 10.1007/s10462-023-10433-3.

Latest Cyber Security News (2019). Fortnite hack could have compromised many gamers accounts. Available at:https://latesthackingnews.com/2019/01/17/fortnite-hack-could-have-compromised-many-gamers-accounts/amp/ (01/17/2025).

Latest Cyber Security News (2020). Vulnerabilities in event service meetup.com could allow group takeovers. Available at:https://latesthackingnews.com/2020/08/09/vulnerabilities-in-event-service-meetup-com-could-allow-group-takeovers/amp/ (01/17/2025).

Liu, M., Zhang, B., Chen, W., and Zhang, X. (2019). A survey of exploitation and detection methods of XSS vulnerabilities. *IEEE Access*, 7:182004–182016. DOI: 10.1109/ACCESS.2019.2960449.

OWASP (2021). Owasp top 10. Available at:https://owasp.org/Top10/ (01/17/2025).

OWASP (2023). OWASP webgoat | OWASP foundation. Available at:https://owasp.org/www-project-webgoat/ (01/17/2025).

OWASP (2024). OWASP juice shop | OWASP foundation. Available at:https://owasp.org/www-project-juice-shop/ (01/17/2025).

PortSwigger (2024a). Burp suite - application security testing software - PortSwigger. Available at:https://portswigger.net/burp (01/17/2025).

PortSwigger (2024b). Web security academy: Free online training from PortSwigger. Available at:https://portswigger.net/web-security (01/17/2025).

Reuters (2018). Ba apologizes after 380,000 customers hit in cyber attack. Available at:https://www.reuters.com/article/us-iag-cybercrime-british-airways/ba-apologizes-after-380000-customers-hit-in-cyber-attack-idUSKCN1LM2P6/ (01/17/2025).

Rodríguez, G. E., Torres, J. G., Flores, P., and Benavides, D. E. (2020). Cross-site scripting (XSS) attacks and mitigation: a survey. *Computer Networks*, 166:106960. DOI: 10.1016/j.comnet.2019.106960.

TryHackMe (2024). TryHackMe | cybersecurity training. Available at:https://tryhackme.com/ (01/17/2025).

West, M. and Sartori, A. (2025). Content Security Policy level 3. Available at:https://www.w3.org/TR/CSP3/ (05/14/2025).

[2]https://gtexss.uff.br/

[3]https://drive.google.com/drive/folders/1Dpi8TWZnwcbUaoh97W8wuYgVQ6S_fajf

[4]https://drive.google.com/drive/folders/1P-I-tWrgNDREgSetLyICa3e28z4hrqby

[5]https://github.com/bguarizi/emuladorXSS-HackersdoBem