

Assessing Web Security in E-Learning Systems

Denisa-Nicoleta MIHALACHE

Faculty of Electronics, Telecommunications and Information Technology,
National University of Science and Technology POLITEHNICA Bucharest, Romania
denisa.mihalache@stud.etti.upb.ro

Abstract

The exponentially evolution of the internet and the increasing sophistication of cyber threats have made securing web servers and web applications a critical concern in today's digital landscape. This research explores the security vulnerabilities of e-learning platforms, particularly Moodle, and demonstrates practical exploitation methods to highlight the risks. A key focus is the development and deployment of a custom script to create a trojan virus leveraging the Right-to-Left Override (RLO) technique. This malware, disguised as a legitimate e-learning material, infiltrates the platform, lists system files, and injects malicious code into Python files, showcasing a high-impact threat vector.

Index terms: cyber-attacks, penetration testing, malicious software, trojan, e-learning security

1. Introduction

The rise of e-learning platforms has transformed the education landscape, providing students and educators with flexible, accessible, and scalable learning environments. By leveraging technologies such as Learning Management Systems (LMS), institutions can deliver courses, assignments, and interactive content seamlessly across the globe. Among these platforms, Moodle has emerged as one of the most widely adopted open-source solutions, enabling customizable and collaborative educational experiences.

However, the growing reliance on web-based platforms has made them attractive targets for cyberattacks. Threat actors exploit vulnerabilities in these systems to compromise sensitive user data, disrupt operations, or propagate malware. Security risks are further amplified as institutions integrate third-party plugins, rely on cloud hosting, and manage large volumes of user-generated content.

This research investigates the vulnerabilities inherent in e-learning platforms, with a focus on Moodle, and demonstrates the implications of a trojan virus attack. By simulating real-world exploitation techniques, such as leveraging the Right-to-Left Override (RLO) character for file obfuscation, the study highlights critical gaps in the security of web servers hosting such platforms. The findings aim to raise awareness about the importance of robust security practices and provide actionable insights to mitigate potential risks.

2. Security testing

Security testing is a critical process in evaluating the security of an application, identifying vulnerabilities and threats, and mitigating them effectively. It plays a key role in the Software Development Life Cycle (SDLC), allowing teams to discover security flaws before they escalate into serious attacks or breaches. By focusing on the integrity, confidentiality, and availability of systems, security testing helps to ensure that applications remain resilient against potential risks.

Security testing provides the advantage of early identification of flaws during the development process, ensuring that vulnerabilities are addressed before they can become expensive or complex to fix. It also prevents the delivery of insecure software that could lead to significant business risks such as compliance violations, legal issues, and reputational damage. In production environments, security testing enables organizations to quickly identify and address vulnerabilities, reducing the chances of exploitation. Furthermore, it encourages continuous improvement by uncovering new threats and enabling organizations to enhance their security measures [1].

Ethical hacking, or penetration testing, involves simulating the tactics of malicious hackers with full authorization to identify weaknesses and improve security defenses. Ethical hackers use the same tools and techniques as cybercriminals, but with the intent of identifying and fixing vulnerabilities rather than exploiting them [2].

White hat hackers, also known as ethical hackers, focus on helping organizations by identifying and fixing vulnerabilities to prevent attacks. Black hat hackers, on the other hand, exploit weaknesses for malicious purposes, often for financial gain or data theft. Gray hat hackers operate in an ethical gray area; although they identify vulnerabilities without permission, they typically notify the system owners of the risks, sometimes requesting a fee in exchange for details of the exploit [3].

Vulnerability scanning tools are widely used to identify weaknesses in operating systems and software, providing organizations with a clear picture of potential security risks. Penetration testing, or simulated attacks, is another valuable technique, allowing organizations to see how their systems would fare under real-world attack conditions and identify areas for improvement. Ethical hacking, often considered the cornerstone of proactive security, involves authorized attempts to break into systems to discover vulnerabilities before malicious actors can exploit them. Security auditing offers a systematic review of system configurations, software, practices, and environments to ensure compliance with industry standards and improve overall security [4].

3. Penetration testing approaches for Moodle

The objective of the penetration testing phase was to evaluate the security posture of the Moodle installation within an Apache2 web server environment. The testing environment was carefully constructed to ensure a secure and controlled setup for the research on improving web server security for Learning Management Systems (LMS). VMware Workstation 17 PRO for Linux was chosen as the platform for virtualization, offering enhanced security features such as snapshots for system restoration. Kali Linux was used as the operating system to leverage penetration testing tools, while Apache2 and MariaDB were installed as core components to host web applications and manage databases securely. Moodle, the selected LMS, was deployed to provide a practical testing environment, with a focus on securing both the server and application to mitigate potential security threats. The methodology involved systematic enumeration and analysis of the application's endpoints, utilizing various reconnaissance tools. The testing process commenced with the reconnaissance phase using open-source tools such as Dirb, Gobuster, Ferox, and Burp Suite. These tools were employed to generate a comprehensive map of the web application and identify potential attack vectors.

The reconnaissance phase provided valuable insights into the application's surface area and potential attack vectors, establishing a foundation for further vulnerability analysis and exploitation efforts.

Next, the analysis of the website, without authentication, revealed an anomaly: course names and numbers were displayed without restrictions, and the author names of these courses were visible to any visitor. From a security perspective, the only information that should be visible to any visitor is the name of the learning platform. Exposing such details could facilitate various attacks, including

social engineering and user impersonation. Additionally, Moodle should restrict the use of default usernames, such as "admin," as this information can be exploited for targeted attacks.

At this point, a brute-force attack attempt was carried out using Hydra, a parallelized login cracker that supports various attack protocols (Figure 1). To test this, a list of commonly used passwords for the "admin" username was required. Along with this, a list of common admin usernames was also created. Using Burp Suite's interception capabilities, I analyzed the behavior of the server when arbitrary data was entered into the login form. The intercepted HTTP form post revealed that a login token was assigned to each session, but this token was not crucial as a new one would be generated for each submission. The error message from the server helped in identifying valid credentials.

Out of 106 tested passwords, 16 were found to be valid, with "Administrator1@" being the correct one. This password, although meeting the minimum complexity requirements (8 characters, including upper and lower case letters, numbers, and special characters), was deemed too simplistic. This breach increases the severity of the vulnerability significantly.

```

kali@kali:~/Desktop$ hydra 127.0.0.1 -l admin -P ~/Desktop/password-list.txt http-post-form "/moodle/login/index.php:anchor=username^USER^&password^PASSWORD":F=Invalid login, please try again"
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-20 01:55:47
[DATA] max 16 tasks per 1 server, overall 16 tasks, 106 login tries (l:1/p:106), ~7 tries per task
[DATA] attacking http-post-form://127.0.0.1:80/moodle/login/index.php:anchor=username^USER^&password^PASSWORD:F=Invalid login, please try again
[80][http-post-form] host: 127.0.0.1 login: admin password: password
[80][http-post-form] host: 127.0.0.1 login: admin password: root
[80][http-post-form] host: 127.0.0.1 login: admin password: wubao
[80][http-post-form] host: 127.0.0.1 login: admin password: 1234
[80][http-post-form] host: 127.0.0.1 login: admin password: admin
[80][http-post-form] host: 127.0.0.1 login: admin password: Administrator1@
[80][http-post-form] host: 127.0.0.1 login: admin password: 123
[80][http-post-form] host: 127.0.0.1 login: admin password: 12345
[80][http-post-form] host: 127.0.0.1 login: admin password: 123456
[80][http-post-form] host: 127.0.0.1 login: admin password: PublishThisListPlease
[80][http-post-form] host: 127.0.0.1 login: admin password: p@ssw0rd
[80][http-post-form] host: 127.0.0.1 login: admin password: 1
[80][http-post-form] host: 127.0.0.1 login: admin password: test
[80][http-post-form] host: 127.0.0.1 login: admin password: root123
[80][http-post-form] host: 127.0.0.1 login: admin password: jiamima
[80][http-post-form] host: 127.0.0.1 login: admin password: l@
1 of 1 target successfully completed, 16 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-20 01:55:48
kali@kali:~/Desktop$

```

Fig. 1. Password cracking using Hydra

Several vulnerabilities were identified, including Cross-Site Scripting (XSS) [5] in various parts of the Moodle interface and the clear-text transmission of passwords. These issues could further compromise the security of the platform and should be addressed promptly. Specific attack paths, such as those found in the "Grader Report," "Group Search," and "Announcements" sections, allow for the injection of malicious JavaScript, leading to potential theft of session cookies or redirection to malicious websites. The lack of encryption for password submission in the login and signup pages is another critical vulnerability, as it exposes passwords to interception, especially in unsecured network environments. Furthermore, the "username persistence" vulnerability was also noted on the login page, where the last valid username entered was retained, even if the user was invalid. This could be exploited by attackers to gain insights into valid usernames on the platform.

To demonstrate the severity of the security vulnerability, I opted for an attack strategy involving the insertion of a malicious virus into the system. Utilizing the administrative privileges previously obtained, I created a virus using the Python programming language. This virus was then uploaded onto the platform in the form of a trojan, disguised as a PDF course material, to illustrate the potential risks associated with unrestricted file upload vulnerabilities.

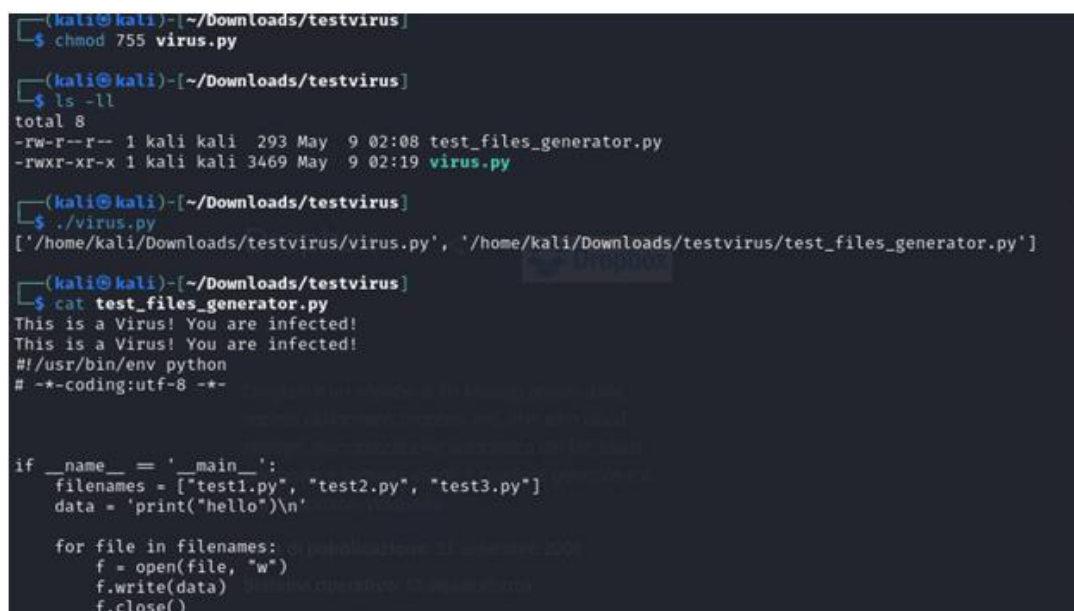
A virus is a type of malicious software (malware) [6] designed to infect files by injecting malicious data or code into them. The virus attempts to enumerate all files across directories and then injects the malicious payload into those files. Unlike worms, which propagate autonomously, this virus does not replicate by itself but continuously infects all files within the specified directories.

In this specific example, once the virus is executed it recursively searches through directories and identifies files with a specific extension (such as .py). For each file, it checks if it has already been infected by looking for a predefined infection string. If the file is not infected, the virus appends the malicious string at the beginning of the file's content. This payload does not replicate itself like a worm but instead continues to infect any new files it encounters within its directory search. The virus can infect numerous files, potentially compromising system integrity and security by modifying and spreading across the file system.

The virus specifically targets .py files because higher privilege accounts, such as system administrators, typically rely on Python for various administrative tasks and scripts. These accounts often have elevated access rights, which makes them a prime target for malware designed to exploit their permissions. By focusing on Python files, the virus takes advantage of the fact that these files are commonly present and frequently executed by privileged users. Once the virus infects these files, it ensures that it operates under the radar, leveraging the trust placed in administrative tasks that use Python scripts. As a result, the virus spreads quietly among the system's critical administrative operations, potentially allowing further exploitation of the compromised environment (Figure 2).

To execute the virus, a timer or a specific execution time can be defined, but in this demonstration, a timer was used for quicker testing. When the virus runs, it will enumerate files and infect those that meet the specified criteria.

To test the functionality of the virus, a controlled environment was created with several Python files in the same directory. Upon execution, the virus successfully inserted the infection message into the files, indicating that it was operating correctly.



```
(kali@kali)~(/Downloads/testvirus)
$ chmod 755 virus.py

(kali@kali)~(/Downloads/testvirus)
$ ls -ll
total 8
-rw-r--r-- 1 kali kali 293 May  9 02:08 test_files_generator.py
-rwxr-xr-x 1 kali kali 3469 May  9 02:19 virus.py

(kali@kali)~(/Downloads/testvirus)
$ ./virus.py
['/home/kali/Downloads/testvirus/virus.py', '/home/kali/Downloads/testvirus/test_files_generator.py']

(kali@kali)~(/Downloads/testvirus)
$ cat test_files_generator.py
This is a Virus! You are infected!
This is a Virus! You are infected!
#!/usr/bin/env python
# -*-coding:utf-8 -*-

if __name__ == '__main__':
    filenames = ["test1.py", "test2.py", "test3.py"]
    data = 'print("hello")\n'

    for file in filenames:
        f = open(file, "w")
        f.write(data)
        f.close()
```

Fig. 2. Repeat access virus testing

The next phase involved turning the virus into a trojan. Trojans are highly dangerous forms of malware due to their ability to load other malware types, execute arbitrary commands, and compromise victim systems without detection. The virus, when deployed as a trojan, can operate covertly while spreading malware or executing harmful actions on the compromised system.

The trojan was disguised as a legitimate PDF file containing course materials. This method, known as **file obfuscation**, involves modifying the appearance of a file to make it seem like an innocuous document (e.g., a PDF) while in reality, it is an executable file.

To achieve this, the .exe file was uploaded to a public document upload platform (e.g., Dropbox), and the file's extension was manipulated using the **Right-to-Left Override (RLO)** technique [7].

The use of Dropbox in the attack is essential for distributing the malicious file while bypassing security filters. Dropbox offers public, easy-to-share file hosting, making it harder for security systems to flag the file as suspicious. By uploading the disguised .exe file with a manipulated name using the Right-to-Left Override (RLO) technique, the attacker masks its true nature, making it appear as a benign document. The public, trusted nature of Dropbox increases the file's legitimacy, and the attacker can easily distribute the link via email or other platforms. This approach allows the attacker to reach multiple victims with minimal risk of detection.

Right-to-Left Override (RLO) technique exploits the way certain languages, like Arabic, are written from right to left. In most systems, file names are displayed left to right, but when the RLO character is inserted into a file name, it causes the system to interpret and display the file name in reverse order.

For example, the attacker might rename a file trojan.exe to something like trojan.pdf by inserting an RLO character before the .exe extension. This character effectively "reverses" the order in which the file name is displayed, making it appear as though the file ends with .pdf instead of .exe. In this case, the RLO character forces the system to read the file extension as .pdf, while the true file extension .exe is still present but hidden from view.

This manipulation makes the file appear as a benign PDF document when viewed by users or security software, but in reality, it is an executable file that can execute malicious code when opened. Since the file's extension is disguised, it bypasses user suspicion and may evade basic security checks that rely on extension names to determine the file's type.

When accessing a course material on Moodle, it appears as a file with a path such as file.pdf, which allows for the simultaneous loading of two URLs: one legitimate (for the course content) and one malicious (containing the virus). This is achieved through an AutoIt script that swaps the two URLs, redirecting the user to the malicious content while still displaying a legitimate appearance (Figure 3).

	BENEFICE ATINGERII OBIECTIVELOR	PUN ÎN PERICOL ATINGEREA OBIECTIVELOR
	Strengths (puncte tari)	Weaknesses (puncte slabe)
SURSĂ INTERNĂ (ORGANIZATIA)	<ul style="list-style-type: none"> - Resurse financiare adecvate ale UPB - Infrastructură IT (hardware) dezvoltată - Disponibilitatea managementului UPB de a susține un proces amplu de transformare digitală - Parteneriatele existente între UPB și firme de IT din București 	<ul style="list-style-type: none"> - Lipsa unei strategii de dezvoltare digitală a UPB - Proces administrativ netransparent și greu de urmărit - Sistemele informatice sunt neadaptate realităților și nu foarte prietenoase cu utilizatorul

Fig. 3. Vulnerable URL

The script effectively manipulates the request, making the malicious file appear to be a harmless course material file. The AutoIt script automates this URL redirection, enhancing the attack's effectiveness and stealth. The script, saved as `trojan.au3`, automates this redirection. Additionally, to make the malicious file appear legitimate, the attacker converts the PDF icon into a `.ico` format, giving it a harmless appearance. The final executable, now appearing as a PDF, is then delivered to the victim, masking its true nature and increasing the likelihood of execution (Figure 4).

The trojan was renamed and uploaded to a platform where users could download it, such as an online course platform (Moodle), disguised as a legitimate course file. Upon downloading and executing the file, the trojan would begin its operation, infecting the system and potentially executing further malicious payloads, such as backdoors or additional malware.

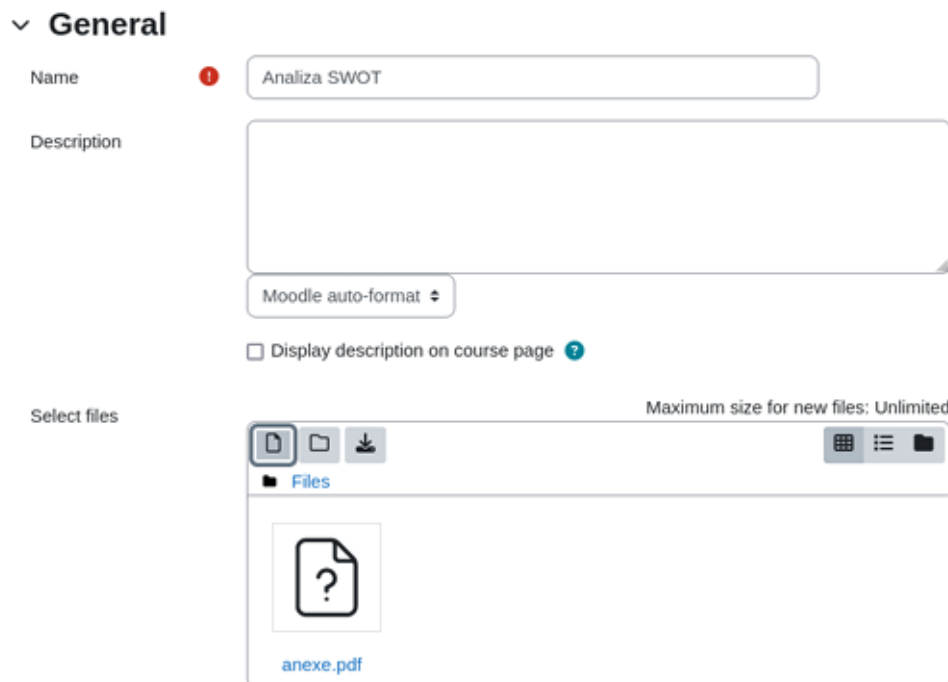


Fig. 4. Uploading the malicious file "*anexe.pdf*"

4. Conclusion

The domain of web server and web application security is vast and continuously evolving, with new threats emerging regularly as technology advances.

Ethical hacking is vital because it simulates real-world attacks, helping organizations uncover vulnerabilities early in the process. By proactively testing systems, networks, and applications, organizations can address weaknesses before they are exploited by malicious actors, ensuring a stronger defense against potential threats. Integrating security testing into the development and operational phases is a strategic approach to maintaining the trust of customers and stakeholders while safeguarding data and systems from unauthorized access.

Malware threats continue to evolve, becoming more sophisticated and harder to detect. The research highlighted the limitations of current malware detection solutions, which often struggle to keep pace with the rapid development of new malware variants. By employing dynamic analysis techniques and behavior-based detection, the time required to identify and mitigate malware infections can be reduced. Sandbox environments, in particular, proved valuable for observing malware behavior in a controlled environment, allowing for more precise identification and prompt response. All software, including web servers and applications, must be updated with the latest security patches.

Password security is a fundamental aspect of web application security, yet many systems still rely on weak password storage mechanisms and inadequate password policies. The brute-force password cracking attack was demonstrated practically. Findings underscored the importance of adopting complex password storage and encryption mechanisms and implementing multi-factor authentication (MFA) to enhance security. Educating users on secure password creation practices is also crucial to reduce the risk of password cracking.

In conclusion, this research focused on the most prevalent attack vectors targeting widely utilized e-learning platform, addressing key security challenges. Progress in testing methods, attack replication, and comprehensive risk assessments offers a strong basis for enhancing the security of web applications. As cyber threats continue to evolve, sustained research and innovation will be crucial to preserving a secure and resilient web infrastructure.

References

- [1]. S. Qadir and S. Quadri, "Information Availability: An Insight into the Most Important Attribute of Information Security," *Journal of Information Security*, 2015.
- [2]. M. Walker, "Certified Ethical Hacker Exam Guide," SYBEX, 2012, pp. 48-55.
- [3]. D. Ghimiray and O. Buxton, "Hacker Types: Black Hat, White Hat, and Gray Hat Hackers," 03 11 2023. [Online]. Available: <https://www.avast.com/c-hacker-types>. Accessed September 7, 2024.
- [4]. R. Messier, C|EH - Certified Ethical Hacker - Study guide, SYBEX, 2023.
- [5]. J. Grossman, R. Hansen, P. Petkov, A. Rage and S. Fogie, "XSS Attacks: Cross Site Scripting Exploits and Defense," in *XSS Attacks: Cross Site Scripting Exploits and Defense*, Syngress, 2007, pp. 67-75.
- [6]. Malwarebytes, "Malware," [Online]. Available: <https://www.malwarebytes.com/malware>. Accessed October 15, 2024.
- [7]. M. ATT&CK, "Masquerading: Right-to-Left Override," 14 10 2021. [Online]. Available: <https://attack.mitre.org/techniques/T1036/002/>. Accessed October 9, 2024.