Penetration Testing on Gaming Portal

***Abstract*—This report presents the results of a penetration test conducted on the Hacme Casino gaming portal, an online platform intentionally designed with vulnerabilities for educational purposes. Using various testing techniques, we identified several critical security flaws, including SQL Injection, CSRF, improper session handling, logic vulnerabilities, and verbose error messages. Each vulnerability is described in detail, including its impact on the application, exploitation steps, and potential countermeasures. This report aims to demonstrate the importance of addressing these vulnerabilities to secure web applications.**

I. Introduction

The objective of this practical is to assess the security of an online gaming portal, Hacme Casino, by conducting a penetration test. Gaming applications often handle sensitive user data and financial transactions, making them attractive targets for attackers. Identifying and addressing vulnerabilities in such platforms is crucial to prevent unauthorized access, data theft, and manipulation of game outcomes. The following sections detail the vulnerabilities identified and highlight their implications.

II. Vulnerabilities Identified

The penetration test revealed five critical security vulnerabilities on the Hacme Casino platform. Each vulnerability is described below, along with its impact, exploitation method, and potential mitigation.

A. 1. Blind SQL Injection

Blind SQL Injection occurs when an attacker can manipulate SQL queries indirectly, often without visible error messages, by sending specific input to exploit weaknesses in the application's database interaction. This vulnerability can lead to unauthorized access, data exfiltration, or administrative control over the application.

Impact: An attacker can bypass the authentication mechanism and log in as a privileged user, granting unauthorized access to sensitive user data and application functionalities.

Exploit Steps:

1) Enter a malicious SQL statement in the username field, such as `' OR 1=1--`.
2) Leave the password field blank.
3) Submit the login form; the application grants access without needing valid credentials.

Mitigation: Use prepared statements with parameterized queries to prevent SQL injection. Additionally, validate user inputs on both client and server sides and limit database error feedback to prevent attackers from inferring database structure.

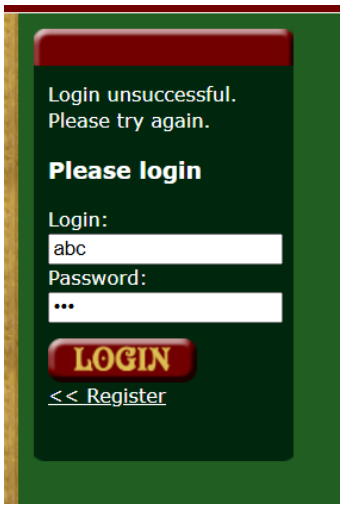


Fig. 1. Using Simple login and password

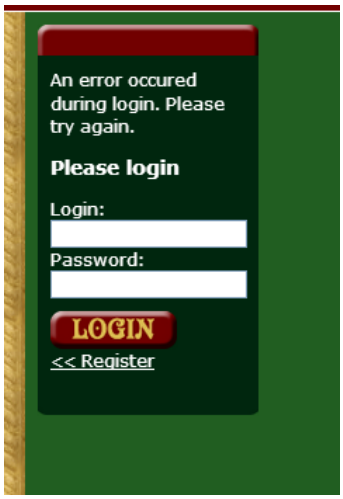


Fig. 2. Adding Single inverted Comma Causes Different Error

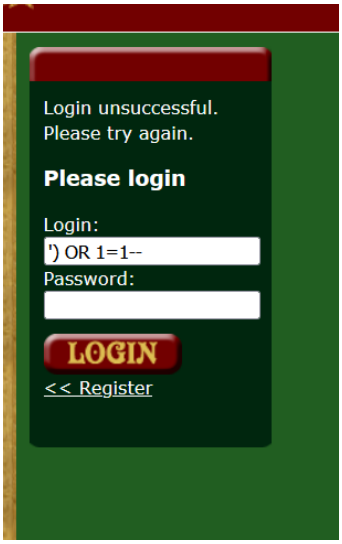


Fig. 3. Blind SQL Injection exploited to bypass login authentication

Fig. 4. Login Successfull

B. 2. Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery allows attackers to trick authenticated users into performing actions on behalf of the attacker, without the user's intent or knowledge. In this scenario, attackers could exploit the CSRF vulnerability to initiate unauthorized transactions on behalf of users.

Impact: Attackers can transfer chips or perform other transactions without user consent, potentially leading to financial losses or manipulation of game assets.

Exploit Steps:

1) Create a URL with embedded commands to transfer chips, such as `http://localhost:3000/account/transfer_chips?transfer=1000&login[]=attacker_account`.
2) Send the link to a target user. When clicked, the transaction completes, transferring funds to the attacker.

Mitigation: Implement anti-CSRF tokens in forms and sensitive actions, and validate the presence of these tokens with each request. Also, restrict sensitive actions to POST requests and require re-authentication for high-risk transactions.

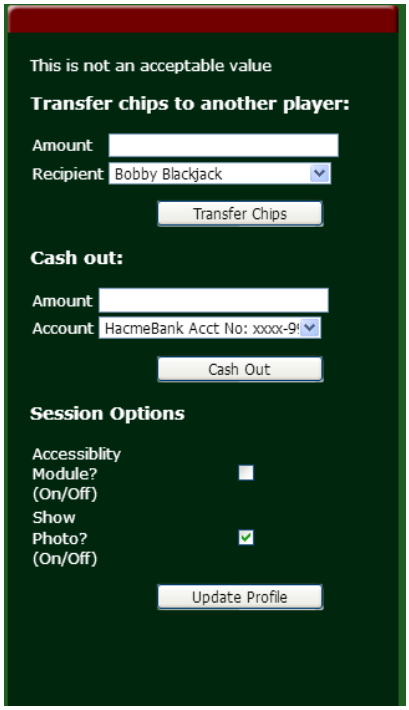


Fig. 5. Transfer Chip to anther player causes error

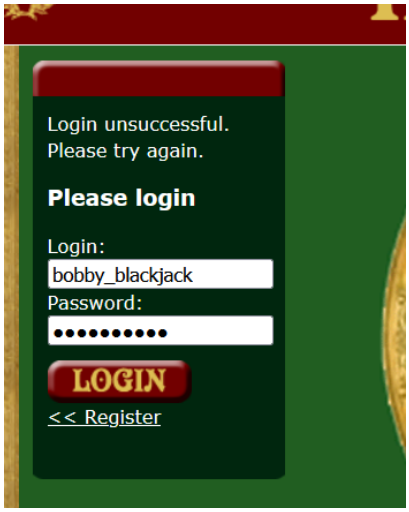


Fig. 6. Login with bobby blackjack Player Id

Fig. 7. CSRF Exploit: Unauthorized transfer of chips using malicious link

Fig. 8. 1000 chips transfered to our account

Fig. 9. Opening BlackJack Game from lobby Tab

Fig. 10. Playing the game to get an idea of it

C. 3. Improper Session Handling

Improper session handling occurs when applications do not correctly manage user sessions, allowing users to manipulate game outcomes by prematurely ending sessions. This allows users to avoid losses in games like blackjack by simply leaving the game mid-round.

Impact: Players can avoid losses by leaving the game before unfavorable outcomes are registered, impacting the platform's revenue and the fairness of gameplay.

Exploit Steps:

1) Start a game and place a bet.
2) If the player's hand is weak, exit to the lobby without completing the round.
3) Re-enter the game and observe that the player's chip count remains unchanged.

Mitigation: Ensure that session data is consistently updated on the server side to reflect game states accurately. Implement a server-side mechanism to complete any incomplete rounds, applying penalties if necessary for prematurely ended sessions.

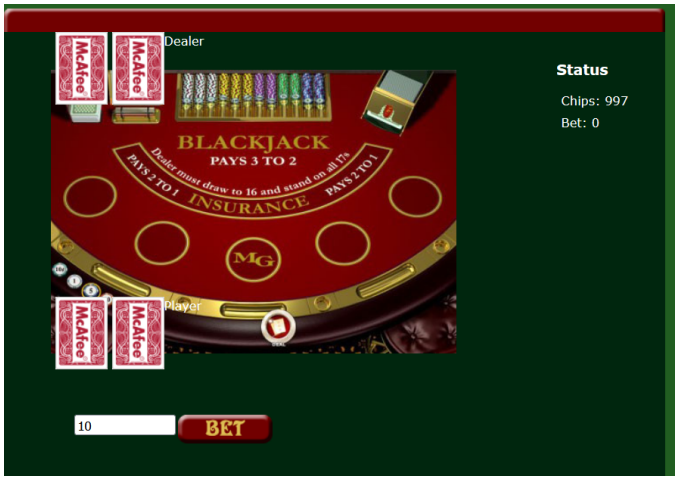


Fig. 11. Now Betting 10 chips on it

Fig. 12. On the verge of loosing the game

Fig. 14. Checking the logic of the bet endpoint

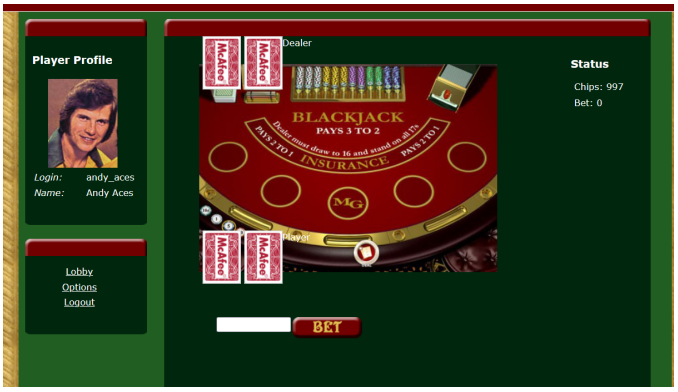
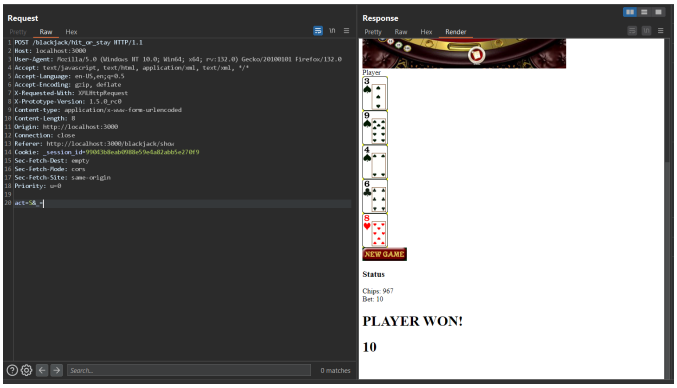


Fig. 13. Improper Session Handling: Chip count remains unaffected after leaving a game mid-round



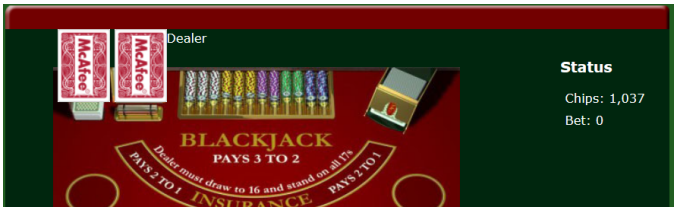


Fig. 16. Application Logic Exploit: Repeating a winning request to inflate chip count

D. 4. Application Logic Vulnerability

An application logic vulnerability arises when the flow of operations allows actions that go against the intended purpose of the application, potentially allowing users to manipulate game outcomes. In this case, users can replay a successful outcome repeatedly by resending a previous winning request.

Impact: This vulnerability allows users to repeatedly trigger a winning payout, leading to inflation of chip counts, which affects the game's integrity and financial stability.

Exploit Steps:

1) Play a hand in the blackjack game and win.
2) Capture the last "win" request using a proxy tool (e.g., Paros) and replay it.
3) Each replay of the request results in an additional payout.

Mitigation: Enforce unique transaction identifiers to prevent replay attacks. Track game states to ensure requests are only processed once, especially for actions tied to game outcomes.

E. 5. Detailed Error Messages

Detailed error messages reveal sensitive information about the application's backend, which attackers can exploit to gain insights into its structure, session data, and even card order in a game. This vulnerability arises when debug information is exposed to end users.

Impact: Attackers can use information from detailed error messages to predict future game outcomes or craft targeted attacks against the application's infrastructure.

Exploit Steps:

1) Access a non-public URL, such as `http://localhost:3000/video_poker`

`/test_deuces_wild`,
to trigger a debug error.

2) Examine the error output to view session data, including card order in the deck.

Mitigation: Limit error message detail in production environments to avoid exposing sensitive information. Use centralized logging systems for detailed error tracking, and disable debug messages in production settings.

Fig. 17. Playing the poker video game

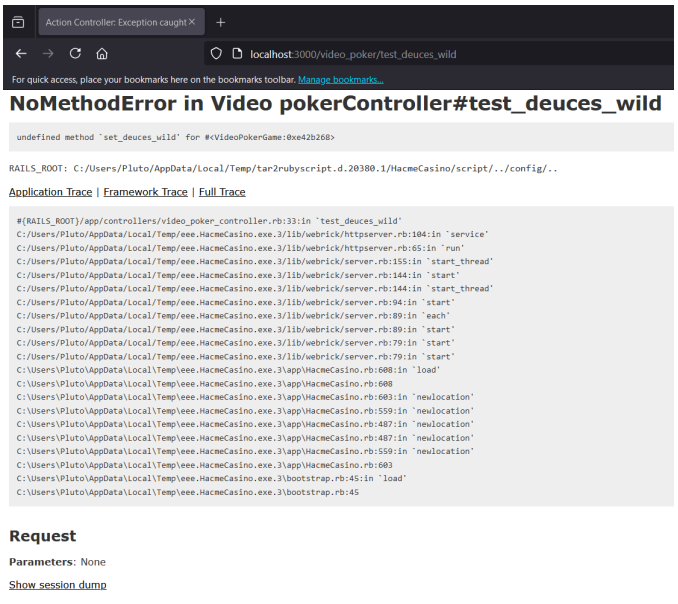


Fig. 18. Detailed Error Messages: Revealing session data and card order with the test-deuces-wild endpoint

Fig. 19. Checking the Session Dump from the error message

Alert type	Risk	Count
Absence of Anti-CSRF Tokens	Medium	2 (15.4%)
Content Security Policy (CSP) Header Not Set	Medium	5 (38.5%)
Missing Anti-clickjacking Header	Medium	4 (30.8%)
Cookie No HttpOnly Flag	Low	6 (46.2%)
Cookie without SameSite Attribute	Low	6 (46.2%)
Possible SQLi	Low	1 (7.7%)
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	26 (200.0%)
Timestamp Disclosure - Unix	Low	7 (53.8%)

Fig. 20. Report Generated in ZAP

III. Conclusion

This penetration test on the Hacme Casino gaming platform identified several critical vulnerabilities, such as SQL Injection, CSRF, improper session handling, application logic flaws,

and verbose error messages. Each of these weaknesses compromises the security and fairness of the gaming application, potentially leading to data breaches, financial loss, or an unfair advantage for attackers. Securing gaming portals is crucial to protecting users, maintaining the integrity of game outcomes, and ensuring that the platform adheres to best practices in application security.