17/Dec/2023 By: Fahad Abdullah Munir

Cybersecurity Consultant & Penetration Tester | Expert in Web Apps, API, Network VAPT
Focused On Cybersecurity Program & Penetration Testing Projects Management.

Vulnerability That Can Allow Attacker Bypass Your Login Page In A Blink Of Eyes

Summary

This report explores a critical security vulnerability in login pages, allowing attackers to swiftly bypass authentication mechanisms using the SQL injection technique. This vulnerability poses a significant threat by enabling unauthorized access to sensitive information, potentially compromising user accounts and exposing sensitive data.

Timely implementation of security best practices, such as input validation and parameterized queries, is essential to mitigate this vulnerability and safeguard against unauthorized access to sensitive information.

The Vulnerability:

This vulnerability, rooted in the arena of SQL injection, manifests as a concise yet potent threat capable of allowing attackers to effortlessly circumvent login page defenses. The deceptively simple payload, **'or 1=1--** becomes the key to unlocking unauthorized access with unparalleled speed, breaching authentication barriers in the blink of an eye.

This report delves into the mechanics of this exploit, exposing the imminent dangers it poses to user accounts, confidential information, and the overall integrity of systems. As we navigate the nuances of this vulnerability, my exploration will illuminate not only the intricacies of the threat but also chart a course toward robust mitigation strategies essential for safeguarding against this swift and stealthy incursion.

Decoding the 'or 1=1—Payload:

‘or 1=1--

This seemingly innocuous string conceals a potent threat that manipulates the logic of authentication processes, providing attackers with an expedited pathway to unauthorized access.

Understanding the 'or 1=1--' Payload:

- The payload is a SQL injection technique designed to manipulate the logical conditions of authentication queries.
- Consisting of the logical OR operator ('or') followed by the condition '1=1,' it ensures the query always evaluates to true.
- The appended double hyphen ('--') signifies a SQL comment, effectively neutralizing the remainder of the query, preventing errors and evading detection.

Functionality and Impact:

- Swift Bypass: The payload exploits the inherent vulnerability in login page input handling, enabling attackers to rapidly bypass authentication mechanisms.
- Logical Manipulation: By forcing the query to always return true, the payload subverts the need for valid credentials, granting access without raising alarms.
- Minimal Footprint: The concise nature of 'or 1=1--' facilitates stealthy execution, allowing attackers to breach security measures in a fraction of a second.

Underpinnings of the 'or 1=1--

Weaknesses in Input Validation:

One of the primary factors contributing to the 'or 1=1--' vulnerability is inadequate input validation in login page forms. When user inputs are not rigorously screened and sanitized, malicious actors can inject specially crafted payloads, such as the 'or 1=1--,' to manipulate the intended behavior of SQL queries. In the absence of robust input validation mechanisms, the door is left ajar for unauthorized access, enabling attackers to exploit the system's trust in the integrity of user-provided data.

Flaws in Source Code Implementation:

At the heart of this vulnerability lies the source code governing the functionality of login pages. A meticulous examination often reveals instances where developers inadvertently overlook or mismanage user input processing. In the context of the 'or 1=1--' exploit, a lack of parameterized queries and failure to employ prepared statements in the source code allow for the direct insertion of user inputs into SQL queries. This oversight enables attackers to manipulate query logic, as the payload is processed as part of the SQL statement, leading to the compromised authentication process.

Example Login Page Code (JavaScript and SQL):

```
// Sample JavaScript code for login page authentication

function authenticateUser(username, password) {

  var query = "SELECT * FROM users WHERE username = '" + username + "' AND password = '" + password + "'";

  // The above code is vulnerable to SQL injection


  // Rest of the authentication logic

}


// Sample SQL query vulnerable to 'or 1=1--' payload

// Original query in the source code

// SELECT * FROM users WHERE username = '[user_input]' AND password = '[user_input]'


// Exploited query with 'or 1=1--'

// SELECT * FROM users WHERE username = '' or 1=1--' AND password = '[user_input]'
```

In the provided example, the absence of parameterized queries in the JavaScript code and the direct incorporation of user inputs into the SQL query create a fertile ground for the 'or 1=1--' vulnerability. Addressing these weaknesses demands a comprehensive approach, involving stringent input validation practices and adherence to secure coding standards. By unraveling the intricacies of source code flaws, we pave the way for a fortified defense against the exploitation of login pages through SQL injection.

Securing Login Pages Against 'or 1=1--

To mitigate vulnerabilities such as the insidious 'or 1=1--' exploit, developers must prioritize robust input validation and diligently employ secure coding techniques. By securing the

foundations of login page architecture, organizations can create an impervious barrier against unauthorized access attempts and secure sensitive user data.

1. Implementing Parameterized Queries:

A pivotal strategy in securing login pages involves the adoption of parameterized queries. Developers should refrain from concatenating user inputs directly into SQL statements, as demonstrated in the vulnerable code snippet. Instead, parameterized queries separate user input from the query itself, preventing the injection of malicious payloads like 'or 1=1--.' By using placeholders for dynamic values and binding parameters, developers can thwart SQL injection attempts, ensuring that user inputs are treated solely as data and not as executable code.

```
// Secure JavaScript code with parameterized query

function authenticateUser(username, password) {

  var query = "SELECT * FROM users WHERE username = ? AND password = ?";

  // Use a secure method to bind parameters, depending on the programming language

  // The use of parameterized queries prevents SQL injection

  // Rest of the authentication logic

}
```

2. Input Validation and Sanitization:

Strengthening the defense against SQL injection necessitates comprehensive input validation and sanitization. Developers should scrutinize user inputs, ensuring they adhere to expected formats and lengths. Employing regular expressions, white-listing, and black-listing techniques can help filter out potentially malicious inputs. By enforcing strict validation standards, developers can reduce the attack surface and impede attempts to inject harmful payloads into login page parameters.

3. Regular Security Audits and Education: In tandem with code fortification, organizations should conduct regular security audits to identify and address vulnerabilities proactively. Additionally, fostering a culture of security awareness among development teams is paramount. Continuous education on emerging threats, secure coding practices, and the

significance of input validation equips developers with the knowledge to preemptively secure login pages against evolving exploits like 'or 1=1--.' By combining robust coding methodologies with ongoing vigilance, organizations can build resilient defenses that withstand the dynamic landscape of cybersecurity threats.

P1-Banks&Other Sectors:

"Championing Security: My Track Record of Success in Mitigating 'or 1=1--' Exploits"

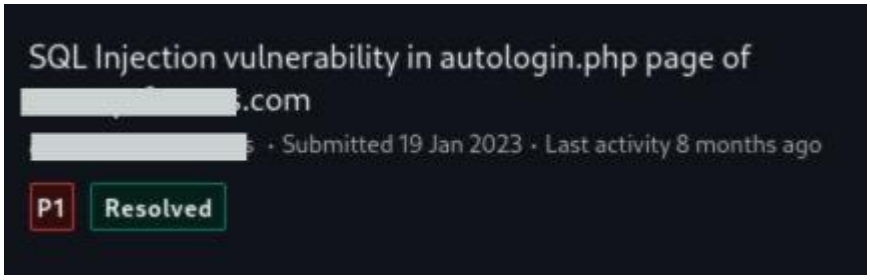
In my journey thus far, I've been privileged to fortify numerous organizations against insidious bypasses, particularly those associated with the notorious 'or 1=1--' SQL injection exploit. My efforts have extended across diverse sectors, including Banks, Hospitals, and Educational Institutions, where I've undertaken comprehensive security assessments. Notably, during these evaluations, the 'or 1=1--' vulnerability emerged as a recurring threat, especially in the banking and government sectors.

Combatting Common Vulnerabilities in Critical Sectors:

The 'or 1=1--' exploit, once a pervasive weakness in login pages, posed a significant challenge for organizations entrusted with safeguarding sensitive information. Through meticulous assessments, I've played a pivotal role in mitigating this vulnerability, contributing to the enhanced security posture of financial institutions and government entities. The proactive identification and remediation of such vulnerabilities not only fortify digital ecosystems but also instill confidence in stakeholders who rely on the integrity of these systems.

A Glimpse into Success: Bugcrowd P1 Report Screenshot:

Attached is a snapshot from one of my recent Bugcrowd submissions, where I reported a critical 'or 1=1--' vulnerability earlier this year. This P1 report exemplifies the tangible impact of proactive security measures, showcasing how timely detection and mitigation can thwart potential exploits. It serves as a testament to the collaborative efforts needed to uphold the security of digital landscapes, and the ongoing commitment to staying ahead of evolving threats. Together, we can continue to fortify organizations, creating resilient defenses that stand strong against the ever-changing landscape of cybersecurity challenges.



By sharing knowledge, implementing robust security practices, and conducting thorough assessments, we can collectively build resilient defenses that withstand the relentless evolution of cyber threats.

I invite you to join the mission of securing digital landscapes against vulnerabilities. Whether you're looking to enhance your organization's security posture or seeking an experienced ally in the fight against cyber threats, let's connect and explore how we can collaborate.

Feel free to reach out on LinkedIn for discussions, insights, or to share experiences in the ever-evolving field of cybersecurity. Together, we can forge a more secure digital future.

Connect with me:

LinkedIn Profile: https://www.linkedin.com/in/fahad-abdullah-8799b7205/