Semgrep Report

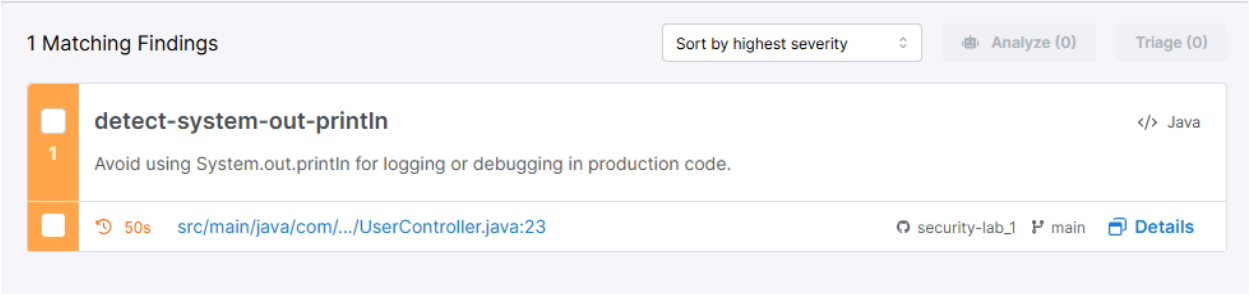
Introduction

In the API development process, I implemented several security measures to enhance the overall security of the application:

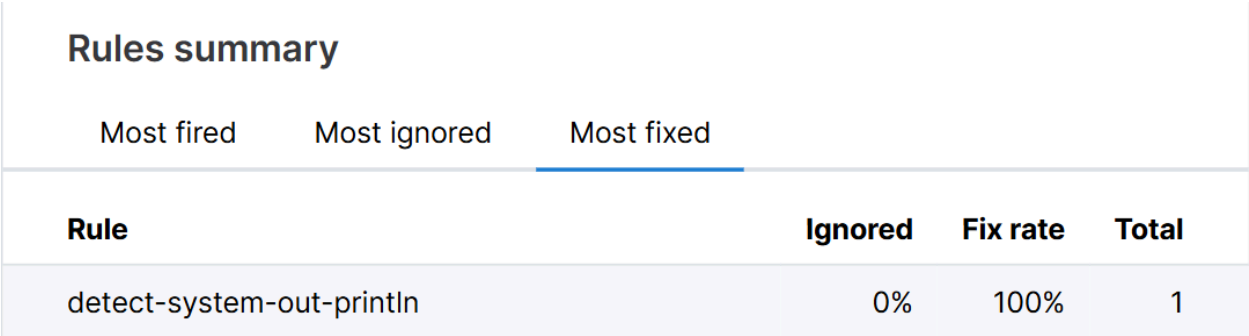
1. **Input Validation and Sanitization:** I used input validation to ensure that only valid and expected data is processed by the API. Additionally, I sanitized inputs by escaping HTML characters using utilities like **HtmlUtils.htmlEscape** to prevent cross-site scripting (XSS) attacks.
2. **Output Encoding:** To mitigate XSS vulnerabilities, I encoded user data before sending it as a response. This ensures that any potentially malicious code in the input is rendered harmless in the browser.
3. **Error Handling:** I implemented custom exception handling to provide clear and secure error messages. This prevents the exposure of sensitive information and helps in diagnosing issues without giving attackers insights into the system's inner workings.
4. **Validation with @Valid Annotation:** I used the **@Valid** annotation in controller methods to automatically validate incoming request data, ensuring that it meets the required constraints before further processing.
5. **Exception Handling with Custom Exceptions:** By creating custom exceptions like **UserNotFoundException,** I provided specific error responses, improving both security and user experience by giving clear and appropriate feedback when errors occur.
6. **Secure Data Handling:** I ensured that sensitive data such as user details were handled securely throughout the API, using secure methods for saving, retrieving, and processing data.
7. **Security Scanning with Semgrep:** I used Semgrep rules to identify potential security issues in the code, such as unescaped user input being returned in responses, and corrected those issues to meet security best practices.

Semgrep output

Vulnerability I have of system out println



After resolving vulnerability



How I resolved it:

Remove system out println on the line specified

As I conclude, my understanding of web security fundamentals has deepened through the practical implementation of secure coding practices in my projects. Specifically, I learned the importance of sanitizing and encoding user inputs to prevent vulnerabilities like cross-site scripting (XSS). By integrating proper output encoding in the service layer, I ensured that user data is safely processed before being sent to the client. I also recognized the significance of error handling to provide meaningful feedback while protecting sensitive information. These

experiences reinforced the necessity of adopting a security-first mindset in web application development.