Secure Code Recommendation Based on Code Review Result Using OWASP Code Review Guide

1st Venia Noella Nanisura Damanik
Crypto Software Engineering
Politeknik Siber dan Sandi Negara
Bogor, Indonesia
venia.noella@student.poltekssn.ac.id

2nd Septia Ulfa Sunaringtyas
Department of Cyber Security
Politeknik Siber dan Sandi Negara
Bogor, Indonesia
septia.ulfa@poltekssn.ac.id

***Abstract*—Sistem Informasi Akademik dan Pengasuhan (SIAP) is an application web that facilitates the academic community in accessing information about academics and student care. However, in its development, if the code system is poorly written, a web application might have vulnerabilities. This study aims to determine the vulnerabilities found in SIAP using OWASP guide and how to fix these vulnerabilities. OWASP is the best choice because it can be obtained free of charge and freely on the internet and periodically up to date. Before vulnerability testing, analysis about security level of SIAP is done using OWASP ASVS. Then, OWASP Testing Guide is used for vulnerability testing. In the tests conducted, it was found that SIAP is vulnerable to injection, broken authentication, and broken access control. Injection is one of the vulnerabilities that has the highest risk value. After testing, a code review using OWASP Code Review Guide is performed to find the location of the vulnerability in the source code. For the last, a secure code recommendation will be given that can overcome the vulnerabilities found from the used guidelines.**

Keywords— code review (1), OWASP (2), recommendation (3), secure code (4), vulnerability (5), web application (6)

I. INTRODUCTION

A. Background

Web applications have become an integral part of the daily life since they are freely available and accessible from any machine through the internet [1]. Web application is also a computer software application coded in a language supported by web browser engines and relies on the browser engine to display the application. In recent decades, there has been a significant increase in existing web applications. The existence of dynamic websites, payment systems, social media, web portals, etc., is possible because of the progress of the functional coding system. In 2016, Alenezi and Yasir tested several open source web applications against common security vulnerabilities and the results point to the fact that hasty programming or lack of developer knowledge about security is a major cause of vulnerability [2]. The fact that many vulnerabilities stem from relatively negligible coding errors and many research findings have shown that the majority of vulnerabilities are related to programming errors that are fairly well understood [3].

In the context of information technology, the term 'vulnerability' pointes out a defect that allows the violation of security policy in a system [4]. Coding disadvantages are very high. A large number of vulnerabilities is followed by errors in the coding. To avoid these vulnerabilities, vulnerability or risk management can be done. One of the efforts to overcome the vulnerability is to conduct a test, namely the penetration test. In addition to the penetration test code review can be done.

Sistem Informasi Akademik dan Pengasuhan (SIAP) is a tool whose main purpose is to become a source of information for the academic community, to provide data and information about academic and student care that can be accessed directly (real-time) when needed. Information provided by the SIAP includes student biographical data, student grades, medical history, psychological history, as well as other information about academics and care. SIAP was developed since November 2018 by third parties using Laravel

Based on an interview with one of the internal development teams of the SIAP, they received a security report regarding SIAP. The security report received is from the user's side. After further checking, the information system under construction has several vulnerabilities in its development, one of which has not been identified security requirements and there is no proper system of documenting input and output requirements. OWASP is an open community dedicated to enabling organizations in developing, purchasing and maintaining applications that can be trusted [7]. OWASP provides several web application development guidelines. To overcome the vulnerabilities found, a study was conducted on efforts to overcome vulnerabilities in SIAP. Efforts to overcome vulnerabilities are carried out by conducting code reviews to find vulnerabilities and recommend secure codes according to the standards used.

II. LITERATURE REVIEW

A. Web Application

According to Garousi et.al [7] a web application such as traditional software requires a development process that involves a collection of requirements and programming through different languages, which leads to diversity in development. Therefore, it is possible to consider web applications as a system that generally consists of databases (or back-end) and web pages (front-end), with users interacting through a network through a search engine.

B. OWASP Application Security Verification Standard (OWASP ASVS)

The Application Security Verification Standard is a list of application security requirements or tests that can be used by architects, developers, testers, security professionals, tool vendors, and consumers to define, build, test and verify secure applications [8].

C. OWASP Testing Guide

OWASP Testing Guide is a complete guide that is available free and open. The OWASP Testing Guide facilitates technical understanding that is used to test security

issues, and helps understand what is dangerous to the SIAP if it is not handled.

D. OWASP Code Review Guide

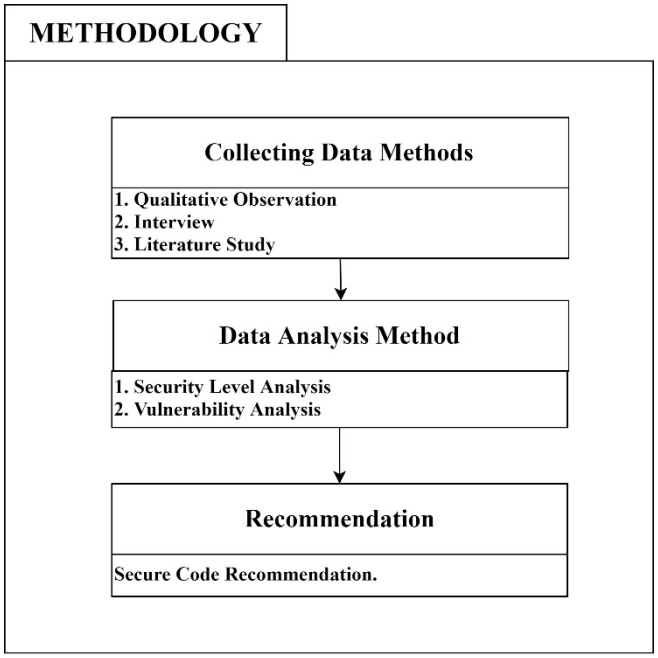
Code review is the process of auditing an application's source code to verify that logical controls function as they should, according to where they are [5]. Code review is a way to ensure applications have been developed so that they can protect themselves in their environment. Code review aims to identify security flaws in applications related to features and design, and those related to the main causes..

E. Penetration Testing Strategies

There are 3 types of penetration testing approaches in software, namely black box testing, white box testing, and gray box testing [10]. According to Infosec, to conduct a safe coding review in general there are 2 techniques, namely automatic based (black box testing) and manual (white box testing). In this research, we will use black box testing and white box testing.

III. Methodology

In this research, there are 2 methods used to find the recommendation.



A. Collecting Data Methods

This research is done with collecting data by 3 steps:

1) Qualitative Observation

Observation is done by finding documents about web application vulnerability.

2) Interview

Interview were conducted with head of subsection Administrasi Akademik dan Kemahasiswaan as well as the internal development team.

3) Literature Study

Finding the literature study about testing and code review.

B. Data Analysis Methods

1) Security Level Analysis

SIAP Security Level Analysi used OWASP Application Security Verification Standard (OWASP ASVS) as guide.

- V1.1 Software Development Life Cycle
- V1.2 Authentication Architectural Requirements
- V1.5 Input and Output Architectural Requirements

2) Vulnerability Analysis

a. Black-box Testing
   Threat guide: OWASP Top 10 – 2017
   - A1: 2017 – Injection
   - A2: 2017 – Broken Authentication
   - A5: 2017 – Broken Access Control

   Attack method: OWASP Testing Guide
   - Testing for Bypassing Authentication Schema (OTG-AUTHN-004)
   - Testing for SQL Injection (OTG-INPVAL-005)
   - Testing for Default Credentials (OTG-AUTHN-002)

   Result: Vulnerability in SIAP found.

b. White-box Testing
   Method: OWASP Code Review Guide
   Result: Vulnerable and impactful source code.

C. Secure Code Recommendation

Secure code recommendation is arranged from guides used and/or documents about web application vulnerability.

IV. Description

A. Security Level Analysis

The development of the SIAP began in November 2018 until now using the Laravel framework version 5.4.36. The language used is PHP and the server used is Apache / 2.4.29. Laravel uses the application of the Model-View-Controller design or commonly abbreviated as MVC. Until the latest development, SIAP consisted of approximately 10,000 lines of source code. The information contained in the SIAP such as academic information, care, student psychology, and control from the subsection Administrasi, Akademik, dan Kemahasiswaan, as well as information supporting education and care activities.

Based on the information stored, referring to the OWASP Application Security Verification Standard (OWASP ASVS), SIAP is included in the level 2 category. Level 2 ASVS obtained from the internal development team of SIAP:

From the identification regarding to V1.1 Secure Development Life Cycle (SDLC) Requirements results, it was found that 5 out of 7 general software development requirements have not been fulfilled. The five points are: the documentation from the stages of SDLC not verified, no verification about using threat modelling of design changes, no verification about architecture and remote services, no verification about centralization, and no verification about requirements, guidelines, or even policies to developers and testers. This allows an injection attack on SIAP.

From the identification regarding to V1.2 Authentication Architectural Requirements, it was found 3 out of 4 architectural authentication requirements have not been fulfilled. The three points are: no verification about using special privilege in all application components, no verification about communications authentication, no verification about authentication mechanism. This allows the existence of authentication vulnerabilities in SIAP.

Based on the identification regarding to V1.5 Input and Output Architectural Requirements, found 3 out of 4 architectural requirements for input and output have not been

identified and are well documented by the developer. The three points are: no verification requirements about handling and processing data, no verification about serialization, no documentation about input validation, no verification about the interpreter place. This has an impact on the vulnerability of access control that may occur in SIAP.

B. Testing

After getting the results of the analysis from the previous stage, the next stage is to test to prove the vulnerability that has been identified. Proof of the vulnerability uses OWASP Top 10 - 2017 as a reference to the attack on the SIAP web page.

1) Injection

Injection testing uses the black box testing method. The steps used are Testing for Bypassing Authentication Schema (OTG-AUTHN-004) and Testing for SQL Injection (OTG-INPVAL-005) from OWASP Testing Guide 4.0 and A1: 2017 - Injection from OWASP Top 10 - 2017 as a reference for launching attack.

One of the stages of Testing for Bypassing Authentication Schema (OTG-AUTHN-004) is done with SQL Injection.

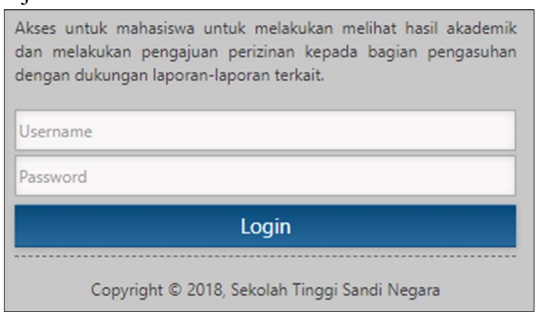


Fig 1. Login Page

The malicious code injected in the login form is the admin username 'OR' 1 '=' 1 and random input for the password. The type of injection is Blind SQL Injection, the attacker enters a malicious code and gets access rights to access the SIAP.

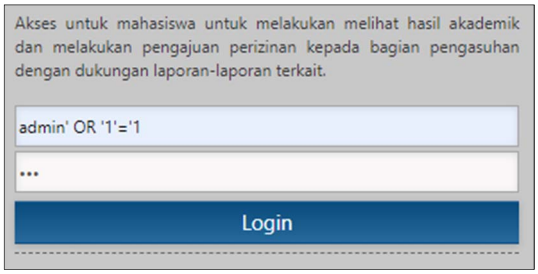


Fig 2. Login Page Injected by Malicious Code

In Figure 4 indicates the user has successfully logged in as a Super User. Super User are user with the highest permissions / privileges in a web application. Users who successfully log in with Super User access rights can modify data on SIAP. Modified data can affect all levels of access rights at both organizers and students. This proves that SIAP is vulnerable, can be penetrated by using malicious code injection attacks on the login page.

When the user login successfully, it indicates the user has successfully logged in as a Super User. Super User is a user with the highest permissions/rights in a web application. Users who have successfully logged in with Super User access rights can modify data on SIAP. Data that can affect at all levels of rights both in organizers and students. This proves that SIAP is vulnerable, it can be penetrated using malicious code injection attacks on the login page.

2) Broken Authentication

After finding a vulnerability at injection, another vulnerability was found on the same page. The location of the broken authentication vulnerability at SIAP is on the login page. This is based on the history of user usernames and passwords that have logged into SIAP before. It shows the user login history view on SIAP.

Testing for Default Credentials (OTG-AUTHN-002) was tested by entering a username which has the first character of a username similar to an account that has logged in before, history of accounts that have entered appears on the login page, complete with username and password. This should not happen, because if an attacker wants to log in, he can use an account that was previously logged in and the password is stored. So it can be said that SIAP is vulnerable to Broken Authentication attacks.

3) Broken Access Control

Robots.txt is a .txt format file that must be owned by a site. SIAP has robots.txt which functions to organize and control which pages or directories search engines can display or index. When accessed, SIAP displays as shown in Figure 4

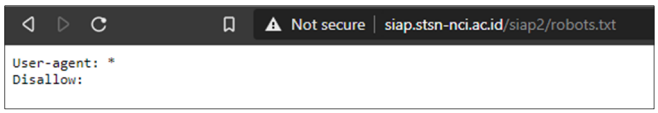


Fig 3. Robots.txt

This should not happen because if you do this, the web page crawler robot can access pages that should not be generally accessible.

C. Code Review

At this stage, a code review is carried out. A manual safe source code review (code review) provides insight into the risk of coding defects. The white-box testing approach is one of the important values in testing. With white-box testing, researchers can understand the relevance of bugs or vulnerabilities in source code. The author understands the relationship between source code in order to be able to estimate risk, take into account possible attacks, and their impact. The categorization of the level of vulnerability helps mitigation efforts and increases the effectiveness of research.

SIAP was developed using the help of a web application framework called Laravel. Laravel version used is Laravel 5.4.36. Judging from the existing developments, the version used by SIAP hasn't used the latest version.

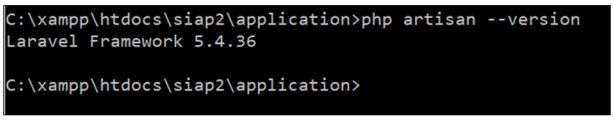


Fig 4. SIAP Version

1) SQL Injection

Untrusted SQL command input is not protected by the machine from attackers. The engine cannot distinguish the SQL command between code and data entered.

TABLE I. LOGINSYSTEMCONTROLLER.PHP

LoginSystemController.php

```
public function LoginAction(Request
$request){
        $userName = request("user_name");
        $password = md5($this-
>stringSec.request("password"));
        $notif = "Login fail, please check
your authentication";
        $type = "warning";

        $user = DB::select(DB::raw("SELECT
* from mst_user where status=1 and
user_name='".$userName."' and
password='".$password."'"));
        if(count($user) > 0) {
            $user = (object) $user[0];
            $id = $user->related_id;
            $appConfig =
Appconfig::find(1);
```

The pseudocode above illustrates the location of the injection vulnerability in SIAP. If there is a change on one page it will affect the other pages because Laravel uses the MVC design.

2) Broken Authentication

Broken Authentication at SIAP occurs when the coding system stores a history of accounts that have accessed the page. Table II is a pseudocode from the source code on the login page.

TABLE II. LOGIN.PHP

Login.php

```
public function __construct($user,
$remember)
    {
        $this->user = $user;
        $this->remember = $remember;
    }
```

Users can log into SIAP by entering their username and password. SIAP will read and save account history and enter users according to the privilege level they have. However, storing the history of accounts that have been entered into SIAP allows attackers to access SIAP at any time.

3) Broken Access Control

When access to SIAP is done by adding robots.txt input, information is obtained that this web setting allows the web page crawler robot (web crawler bot) to know the entire directory list contained in the web application. Table III shows the source code in robot.txt

TABLE III. ROBOTS.TXT

robots.txt

```
User-agent: *
Disallow:
```

User-agent is filled to determine which robot is allowed to index. In the source code above, using '*' in the User-agent means that all robots are allowed to index all pages. Disallow is filled to determine restrictions on pages or files that are allowed to be indexed by the robot into SIAP. From the picture above, it can be concluded that SIAP gives permission to any robot to index and does not limit the pages that can be accessed.

D. Secure Code Recommendation

1) Injection

SQL Injection attacks occur when injecting SQL queries on a back-end database system, in this study included in the SIAP login page. So, to avoid SQL injection in PHP, use addslashes() and mysql_real_escape_string() functions.

a) addslashes()

addslashes() is used by adding quotes to the query string

b) mysql_real_escape_string():

mysql_real_escape_string() is slightly stronger than addslashes() which calls the MySQL library function which adds backslashes to the following characters: \x00, \n, \r, \, ‘," and \x1a. Just like addslashes(), mysql_real_escape_string() will only work if the query string is enclosed in quotation marks.

2) Broken Authentication

To fix the vulnerability of SIAP which stores user history, one of the ways is the built-in session manager which generates a random session ID with high entropy after login. PHP does not have a session time limiting mechanism. Then the developer needs to create its own session time limitation. The ID of each session is not stored in the URL, but is safely invalid after logout, idle, and absolute timeout.

```
if (isset($_SESSION['LAST_ACTIVITY']) &&
(time() - $_SESSION['LAST_ACTIVITY'] >
1800)) {
    // last request was more than 30
minutes ago
    session_unset();// unset $_SESSION
variable for the run-time
    session_destroy()// destroy session
data in storage
}
$_SESSION['LAST_ACTIVITY'] = time(); //
update last activity time stamp
```

3) Broken Access Control

From the results of the code review above, writing the source code should include restrictions on the robot and its pages. If there are no restrictions given, any web crawler can access all pages or files contained in SIAP. To avoid the existence of bots that can read the entire contents of a web application, restrictions can be given in the User-agent and Disallow sections. If you allow web crawlers to access SIAP, restrictions can be made in Disallow by adding Directories or Files that should not be accessed. As:

TABLE IV. ROBOTS.TXT

robots.txt

```
User-Agent: *

# Directories

Disallow: /app/
Disallow: /bin/
Disallow: /documentation/
Disallow: /home/
Disallow: /main/
Disallow: /plugin/
Disallow: /tests/
Disallow: /vendor/

# Files
Disallow: /license.txt
Disallow: /README.txt
Disallow: /.htaccess
```

V. Conclusions and Suggestions

Based on the analysis of the tests and code reviews that have been done, the results show that SIAP is vulnerable to injection attacks, broken authentication, and broken access control. Therefore, a secure code recommendation is given to overcome the vulnerabilities found, namely injection, broken authentication, and broken access control.

The secure code recommendations given are in the form of:

a. Adding addslashes() and mysql_real_escape_string() functions to the SIAP login system,
b. Provide a time limit on account login sessions,
c. Include restrictions on directories and files that can be accessed by web page crawler / web crawler.

For further research, suggestions that can be done when an organizations develop or build web applications, it's better to do identification and verification of security requirements to find whether the application is secure to used. In the other hand, when the research was doing, found others vulnerabilities that may affect the web application. Therefore, the future research may overcome and mitigate the discovered vulnerabilities.

References

[1] G. Deepa and S. Thilagam, "Securing Web Applications from Injection and Logic Vulnerabilities:," Information and Software Technology, 2016.

[2] M. Alenezi and Y. Javed, "Open Source Web Application Security: A Static," in Engineering & MIS (ICEMIS), 2016.

[3] A. Agrawal, M. Alenzi, R. Kumar and R. A. Khan, "A Source Code Perspective Framework To Produce Secure Web Application," Computer Fraud & Security, vol. 10, pp. 11-18, 2019.

[4] E. R. Russo, A. Di Sorbo, C. A. Visaggio and G. Canfora, "Summarizing Vulnerabilities' Description to Support Experts during Vulnerability Assessment Activities," Journal of Systems and Software, 2019.

[5] O. W. A. S. P. (OWASP), "OWASP Code Review Guide 2.0," The OWASP Foundation, 2017.

[6] O. W. A. S. P. (OWASP), "OWASP Top 10 - 2017: The Ten Most Critical Web Application Security Risk," The OWASP Foundation, 2017.

[7] V. Garousi, A. Mesbah, A. Betin-Can and S. Mirshokraie, "A systematic mapping study of web application testing," Information and Software Technology, vol. 55, p. 1374–1396, 2013.

[8] T. O. Foundation, "OWASP Application Security Verification Standard 4.0," Open Web Application Security Project, 2019.

[9] S. Shah and B. Mehtre, "An Overview of Vulnerability Assessment and Penetration Testing Technique," Springer, vol. XI, pp. 1-23, 2014.

[10] L. Blessing and A. Chakrabarti, DRM, A Design Research Methodology, New York: Springer, 2009.

978-1-7281-9098-3/20//$31.00