International Journal of Advanced Smart Convergence Vol.10 No.4 22-29 (2021)
http://dx.doi.org/10.7236/IJASC.2021.10.4.22

IJASC 21-4-3

Security Measures by Diagnosing Vulnerabilities in Web Applications

Kim Hee Wan

[1]*Prof., Division of Computer Science & Engineering, Sahmyook Univ., Korea*
E-mail hwkim@syu.ac.kr

Abstract

For web application vulnerability diagnosis, from the development stage to the operation stage, it is possible to stably operate the web only when there is a policy that is commonly applied to each task through diagnosis of vulnerabilities, removal of vulnerabilities, and rapid recovery from web page damage. KISA presents 28 evaluation items for technical vulnerability analysis of major information and communication infrastructure. In this paper, we diagnose the vulnerabilities in the automobile goods shopping mall website and suggest security measures according to the vulnerabilities. As a result of diagnosing 28 items, major vulnerabilities were found in three items: cross-site scripting, cross-site request tampering, and insufficient session expiration. Cookie values were exposed on the bulletin board, and personal information was exposed in the parameter values related to passwords when personal information was edited. Also, since the session end time is not set, it was confirmed that session reuse is always possible. By suggesting security measures according to these vulnerabilities, the discovered security threats were eliminated, and it was possible to prevent breaches in web applications and secure the stability of web services.

Keywords: *web application, web vulnerability, cross-site scripting, cross-site request tampering, insufficient session expiration, security measures*

1. Introduction

Web vulnerability check is to find the vulnerabilities of web applications that exist in the target website. The white box test method finds vulnerabilities by analyzing the source code, and the black box test method checks the vulnerabilities by substituting data into the target web page. White box testing is mainly performed in the stage before software distribution or user use, and the black box testing method is often used for applications that have already been distributed. In actual industrial sites, black box inspection is mostly used rather than white method inspection, and web vulnerability inspection generally checks whether a web vulnerability exists in the web site to be inspected based on predefined web vulnerability inspection items. Web vulnerability check items are checked using the major information and communication infrastructure web vulnerability check items, where most companies legally provide check items clearly.

According to a study conducted by Cenzic, an application security service provider in the United States, 96% of the applications used in the test for the study had an average of 14 security vulnerabilities [1]. As a result

Manuscript Received: September. 22, 2021 / Revised: September. 25, 2021 / Accepted: September. 29, 2021
Corresponding Author: hwkim@syu.ac.kr
Tel: +82-2-3399-1783, Fax: +82-2-3399-1791
Professor, Division of Computer Science & Engineering, Sahmyook Univ., Korea

of web vulnerability inspection of 1,242 websites by the Korea Internet & Security Agency (KISA) in 2016, about 73% (895 sites) were found to be vulnerable [2]. In order to solve and defend against these web vulnerabilities,

many companies and institutions are spending huge amounts of money and effort, but various web vulnerabilities are still found in many websites, and damage caused by attacks that exploit them are frequently reported in the media [3].

For web applications, from the development stage to the operation stage, the system can be operated stably only when there is a policy that is commonly applied to each task for diagnosis of vulnerabilities, removal of vulnerabilities, and rapid recovery from damage. There are several vulnerabilities of web application as follows. Among them, it can be considered as vulnerable when the vulnerability diagnosis is neglected during application development, when the business logic of the entire application is completely separated from the security core process logic, or when vulnerability analysis and diagnosis is not performed periodically. Cases where the application is unexpectedly damaged include using code that is vulnerable to security, recognizing that basic operation and management are under the jurisdiction of an individual, or installing various files that the user does not recognize through frequent Internet access outside of work, or It can be seen that it occurs frequently when an external storage medium (USB Memory, Hard Disk, SD Memory, etc.) is easily used without any recognition, or when a user without basic knowledge of computer operation installs a program and makes a mistake or deletes a file. With the recent spread of various hacking technologies and the introduction of information protection and information security system projects in preparation for this, the improvement of the web application part, which is the most vulnerable to information security, is emerging as the most important issue in the government, public offices, and companies. As large-scale projects related to web application information protection and information security in corporations and public institutions constitute and integrate various information systems, the information security risk factors inherent in each application have increased exponentially. In general, hacking attacks or security breaches of web applications have different attack patterns for each web application, and there are cases where vaccines cannot detect them, and there are hardly any traces of the attack left in the log, making it difficult to take security measures or respond. Information protection and security are important in corporations and public institutions, and among them, web-related applications are the most vulnerable to security [4][5].

Therefore, in this paper, the possible path of intrusion accidents in the car goods shopping mall web application operated by CAMEL is identified and the impact of the threat on the service is analyzed. In addition, we intend to propose a method to prevent breaches and secure stability by removing the discovered threats by presenting security measures.

2. Related Work

2.1 Web vulnerability analysis evaluation items

A security vulnerability is a bug (wrong part) inherent in a system or program, and hackers exploit it to break into the system and cause information leakage and system destruction. Vulnerability checking refers to finding these security vulnerabilities inherent in a system or program. Vulnerability analysis and evaluation presented in the evaluation criteria for vulnerability analysis of major information and communication infrastructure are largely divided into administrative, physical, and technical, and technical vulnerability inspection is again performed on servers (Window, Unix), network equipment, security equipment, DBMS, PC, and web. Web vulnerability check checks the operation (response) of the web server and web application to the request from the client. Inspection of other systems (servers, network equipment, etc.) proceeds by checking the setting values (Config values) of the target system [6][7]. Security threats in these content areas can lead to vulnerability in most applications. Vulnerabilities in the application can bypass authentication and view unauthorized information [8]. Evaluation items for web vulnerability analysis consist of a total of 28 items based on KISA's evaluation items for technical vulnerability analysis of major information and communication infrastructure [9]. According to statistical surveys, most of the security problems are related to web applications, and their vulnerabilities are increasing significantly. It is predicted that the majority of companies and government offices will continue to have problems related to web application security. We believe that policies to gradually improve web application vulnerabilities through static analysis and dynamic analysis are necessary [10]. In this paper, the purpose of this paper is to suggest security measures through vulnerability diagnosis and results for cross-site scripting, cross-site request tampering, and insufficient session expiration, and to provide stable web services by preventing breaches.

2.2 Cross-site scripting

It was created with the purpose of blocking the execution of malicious scripts by removing the cross-site scripting vulnerability in the web page. If the filtering of the user input argument value is not performed properly in the web application, a malicious script (JavaScript, VB script, ActiveX, Flash, etc.) Security threats can be imposed by stealing the reader's cookie (session) or distributing malicious code (URL redirect). It is mainly checked for source code or web firewall, and it can be said that it is good when verification and filtering of user input argument values are performed. However, if validation and filtering of user input values are not performed or HTML code is input and executed, it can be said to be vulnerable. As a workaround, you can add verification logic for the argument values input from the user in the website bulletin board, archive, URL, etc., or prevent execution even if the argument values are entered. In case of using HTML on the bulletin board inevitably, it should be set to allow input of only necessary codes among HTML codes.

2.3 Cross-site request tampering

Cross-site request tampering is an attack type that allows a user to request an attacker's intended action (modification, deletion, registration) from a specific website regardless of his/her will.

It was created for the purpose of preventing tampering with requests in trust (authentication) information by validating proper filtering and authentication of user input values. There is a security threat that can perform a malicious attack with the authority of the user by tampering with the user's request within the user's trust (authentication) information. It can be said that it is good when verification and filtering of user input values are performed. However, if the user input value is not filtered or HTML code (or script) is input and executed, it can be said to be vulnerable. As a countermeasure, verification logic and filtering should be additionally applied to the user input value.

2.4 Insufficient session expiration

It was created to prevent an attacker from using an unexpired session by implementing the session timeout function. As a security threat, if the expiration period of the session is not set or if the expiration period is set too long, a malicious user may use an unexpired session to gain illegal access. It can be said that it is good if the session end time is real with the source code and the web server as the inspection object. However, if it is not set at the session end time and session reuse is possible, it can be said to be vulnerable. As a countermeasure, session end time setting or automatic logout function should be implemented. Since the session end time may vary depending on the characteristics of the site, an appropriate time should be set according to the characteristics of the site.

3. Web Vulnerability Diagnosis

The web vulnerability diagnosis tool used in this study was used to check and modify parameter values transmitted to the server with Burpsuite, and the web scanner OWASP-ZAP was used to check web application vulnerabilities.

3.1 Cross-site scripting diagnosis

This is a vulnerability that can be exploited through session hijacking and malicious program installation as a vulnerability in which an arbitrary script is executed on the client of another end user using a web application. Validation and filterlet of user input values are not done, and HTML code is often executed upon input. When an article was written on the bulletin board so that the cookie value could be exposed as a warning window, it was confirmed that the cookie value was exposed as a warning window as shown in the following Figure 1. An attacker can exploit it by storing the cookie value in a different path. Figure 1 shows the cookie value exposed in the warning window.

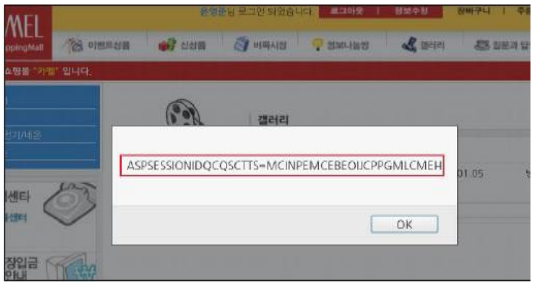
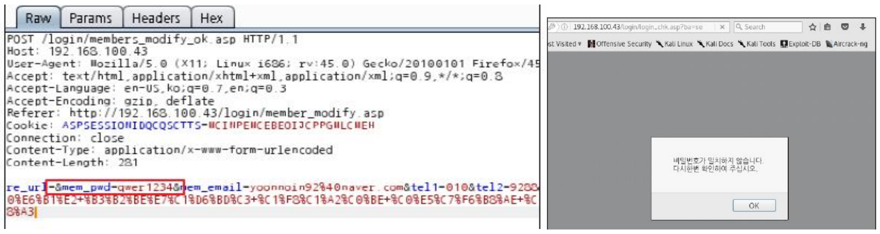

ASPSESSIONIDQCQSCTTS=MCINPEMCEBEOIJCPPGMLCMEH

Figure 1. Result of cross-site scripting diagnosis

3.2 Diagnosis of cross-site request tampering

Cross-site request tampering is a vulnerability that occurs in web applications based on automatically entered trust (authentication) information such as session cookies, SSL certificates, and Windows domain authentication. it is possible to perform There is no filterlet for user input values, and it is vulnerable when executed by entering HTML code. When modifying personal information, it was confirmed that the parameter value related to password was exposed in plain text. If a malicious script phrase that causes passwords to be changed by clicking a link is inserted on the bulletin board, it can make it impossible to log in with the message that the passwords do not match in the login with the existing password. Figure 2 shows that the password entered in HTML is exposed because no filterlet is performed on the user input value.



```
POST /login/members_modify_ok.asp HTTP/1.1
Host: 192.168.100.43
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:45.0) Gecko/20100101 Firefox/45
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,ko;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.100.43/login/member_modify.asp
Cookie: ASPSESSIONIDQCQSCTTS=MCINPEMCEBEOIJCPPGMLCMEH
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 281

re_url=&mem_pwd=qwer1234&mem_email=yoonnoin92%40naver.com&tel1=010&tel2=9288
0%E6%B1%E2+%B3%B2%BE%E7%C1%D6%BD%C3+%C1%F8%C1%A2%C0%BE+%C0%E5%C7%F6%B8%AE+%C
8%A3
```

비밀번호가 일치하지 않습니다.
다시한번 확인하여 주십시오.

Figure 2. Result of cross-site request tampering

3.3 Diagnosis of insufficient session expiration

This is a vulnerability that allows an attacker to take advantage of an unexpired session by not setting the expiration period of the session or setting the expiration period too long. Since the session end time is not set, session reuse is always possible. As shown in the Figure 3, the administrator session is maintained even after a certain period of time (more than 10 minutes) has elapsed. Figure 3 shows that the administrator session is maintained without logout even after a certain period of time has elapsed.

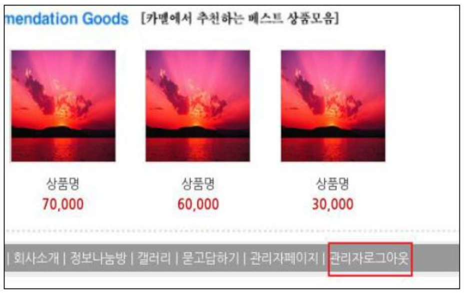

mendation Goods [카멜에서 추천하는 베스트 상품모음]
상품명
70,000
상품명
60,000
상품명
30,000
| 회사소개 | 정보나눔방 | 갤러리 | 묻고답하기 | 관리자페이지 | 관리자로그아웃

Figure 3. Result of insufficient session expiration

4. Security Measures for Vulnerability

4.1 Security measures of cross-site scripting

Page types that are vulnerable to XSS are likely to occur in bulletin boards that support HTML, Search Pages, Join Form Pages, Pages using Referrer, and all other pages that are input from users and output to the screen. Figure 4 is an example of a script that can cause XSS.

```
<script> … </script>
<img src="javascript:…..">
<div style="background-image:url( javascript…)"></div>
<embed>…</embed>
iframe></iframe>
```

Figure 4. Example of cross-site scripting security measures

The security setting method in cross-site scripting requires restricting the use of tags corresponding to HTML or JavaScript in posts in advance, and filtering the values entered by the user. Filterlet is performed not only on the body of the post, but also on the title, comment, search word input window, and all other forms and parameter values that trust the value passed by the user. When implementing filterlet logic for input values, it must be implemented on the server side by using trim and replace functions that remove whitespace characters.

4.2 Security measures of cross-site request tampering

The cross-site request tampering check method checks whether there is a page with data modification functions such as registration and change, and then retransmits the request transmitted from the data modification page to check whether the normal data modification function is re-executed. have. The security setting method should be designed to add an arbitrary token in Form/URL and verify this token so that a normal request can be distinguished from an abnormal request by implementing the verification logic for the value input by the user. The use of tags corresponding to HTML or JavaScript should be restricted in advance, and a filterlet for user input values should be implemented at the server level. In addition, when the above actions cannot be taken due to the use of HTML Editor, it should be designed to take action in the server side/servlet/DAO (Data Access Object) area.

4.3 Security measures of insufficient session expiration

The method of checking the vulnerability of insufficient session expiration is to obtain the request of the page from which the session is normally issued after authentication and check whether normal processing is performed when retransmitted after a certain period of time (10 minutes or more) has elapsed.

As for the security setting method, if the session timeout function is not implemented, there is no protection device for users who have been absent for a long time.

■ Apcache

Timeout function can be implemented using *session object that can store user information by creating a session for each visitor. Session object: It is often used to permit or prohibit page access or to store information for each user. The session object can be used only when the visitor's browser supports the cookie function. If the following settings are applied, the session is deleted immediately when the user logs out, and if there is no request to the web server for 10 minutes without logging out, the session is lost. Figure 5 shows the setting so that the session is terminated when there is no request from the web server for more than 10 minutes by setting the session timeout time to 10 minutes.

```
// Setting of session maintenance time
Session.timeout = 10
```

Figure 5. Session time setting of Apcache

■ JSP

To implement the session timeout function, use session.getLastAccessedTime() to automatically end the session if the session is not accessed again within a certain period of time from the last access time of the session. The session timeout can be set in two ways.

Figure 6 shows how to specify a timeout using the <session-config> tag in the web.xml file.

```
Web.xml : "minute" unit
        <session-config>
       <session-timeout>10</session-timeout>
        </session-config>
```

Figure 6. Session timeout setting of web.xml in JSP

Next, by using the setMaxInactiveInterval() method provided by the session default object, you can automatically end the session by entering the unit of seconds. Figure 7 shows how to set the session duration using the setMaxInactiveInterval() method provided by the session default object.

```
String strTime = param.getPropertyFromXML("SessionPersistenceTime");
If (strTime == null) {
        Session.setMaxInactiveInterval(600);
} else {
        Session.setMaxInactiveInterval((new Integer(strTime)).intValue());
}
```

Figure 7. Setting of setMaxInactiveInterval()

5. Conclusion

For web applications, from the development stage to the operation stage, it is possible to stably operate the web only when there is a policy that is commonly applied to each task to diagnose vulnerabilities, remove vulnerabilities, and quickly recover from damage. For vulnerabilities in web applications, 28 items are presented based on the evaluation items for technical vulnerability analysis of major information and communication infrastructure provided by KISA. In this paper, the vulnerabilities in the CAMEL automobile shopping mall website were diagnosed. As a result of diagnosing 28 items, the diagnosis was based on 3 items: cross-site scripting, cross-site request tampering, and insufficient session expiration. As a result of the diagnosis, in the case of cross-site scripting, if an article was written on the bulletin board so that the cookie value could be exposed as a warning window, the cookie value was exposed as a warning window. Therefore, an attacker can exploit this by storing the cookie value in a different path. In the case of cross-site request falsification, when personal information is modified, password-related parameter values are exposed as plain text. If a malicious script phrase to change the password was inserted through a link click on the bulletin board, it was possible to make it impossible to log in with the message that the password did not match in the login with the existing password. In case of insufficient session expiration. It was confirmed that session re-use is always possible because the session end time is not set. By suggesting security measures according to these vulnerabilities, it was possible to remove the discovered security threats, prevent breaches, and secure the stability of web services.

REFERENCES

[1] D. S. Jeong, *Many security vulnerabilities found in most web applications*, Apr, 2017 http://digitalyeogie.com/entry/49451?locPos=25 Q&ts=1487390696&page=4862, Apr. 2017

[2] KISA, *Web vulnerability analysis and technical support*, Korea Internet Security Agency, 2016

[3] M. S. Kim, "*Bombu hacking is attacking web vulnerability DB-the reason for the 2 million personal information was revealed,*" http://news.kukinews.com/news/article.html?no=317262, May 2017.

[4] J.B. Kim, "*A Study on the Successful Implementation about Vulnerability Supplementation and Effective Recovery from Damage related with Web Application,*" *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology (AJMAHS),* Vol.6, No.2, pp.53-60, 2016 doi:10.35873/ajmahs.2016.6.2.007

[5] H.H. Jin, and H.K. Kim, "*A Study on Web Vulnerability Risk Assessment Model Based on Attack Results : Focused on Cyber Kill Chain*," *Journal of The Korea Institute of Information Security & Cryptology,* Vol.31, No.4, pp.779-791, 2021
doi:/10.13089/JKIISC.2021.31.4.779

[6] J.H Lee, "*Risk Level Assessment Method on Major Information and Communication Infrastructure's Web Vulnerability Check Item using by AHP Technique*," *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology (AJMAHS)*, Vol.9, No.6, pp.719-728, 2019
doi: 10.35873/ajmahs.2019.9.6.070

[7] J.H. Lee and S.J. Lee, "*Improvement of Dynamic Web Vulnerability Inspection Method and Procedure by Website Structuring and Calculating Each Page's Action Size*," *Journal of Knowledge Information Technology and Systems (JKITS),* Vol.12, No.5, pp.747-763, 2017
doi:10.34163/jkits.2017.12.5.015

[8] H.W. Kim, "*A Study on the Mobile Application Security Threats and Vulnerability Analysis Cases," International Journal of Internet, Broadcasting and Communication (IJIBC),* Vol.12 No.4 180-187, 2020
http://dx.doi.org/10.7236/IJIBC.2020.12.4.180

[9] Korea Internet & Security Agency*, Detailed guide on how to analyze and evaluate technical vulnerabilities of major information and communication infrastructure*, Feb, 2017

[10] Daeil Yang*, Information Security Overview*, Hanbit Academy Publisher, pp. 235-238, 2013