ISSN:0975-9646

Detection Model for CSRF and Broken Authentication and Session Management Attack

Virginia Mary Nadar[#1], Madhumita Chatterjee[#2,] Leena Jacob[#3]

[#1,2]*Department of Information Technology*
[#3]*Department of Computer Engineering*
Mumbai University
PIIT, New Panvel, Navi Mumbai, India

***Abstract*—Online application security is information security that consist of security of websites, web applications and or web services. Developed online application security relies on the foundation of application security but focuses on world wide web and their libraries. Due to the advancement in Web 2.0, more knowledge sharing through social networking and increasing business adoption over the web for doing business and delivering services, web applications are directly attacked. Attackers rather try to compromise the company network or the users accessing the website by forcing them to click on the forged malicious input, because of which industry is focusing more attention to online application security along with security of the underlying computer network and operating systems. Online application designing should be improved by including security analysis and checks at early stages of development as well as throughout the software development life-cycle. As most of the existing systems detect only one attack at a time with limited rules, we propose an enhanced model that can detect two attacks within the same simulation environment with updated rule libraries.**

Keywords— Cross-Site Request Forgery, CSRF, Broken Authentication and Session Management, XSS

I. INTRODUCTION

Security test is a method of evaluating the weakness or flaws of a computer system or a network by methodically validating to monitor the quality of application security controls. Online application security test aims on checking the security of different online applications. A security test includes filtered analysis of the application for any design flaws, coding mistakes or component misconfiguration. Whether if any security drawbacks are identified, the client is alerted.

Vulnerability is identifying failure and/or flaws in a system's design, implementation, working or management that exploits the system's security objectives.

Threat is defined in terms of a malicious external attacker, an internal user, a system instability, etc which can endanger the assets owned by an application like resources of value, such as the data in a database or in the file system by exploiting vulnerability.

Malicious user can probably use many different ways to do harm to the business or organization online application. Every path of the application represents a risk that may or may not require to warrant attention.

A test explains an action to depict that whether an online application meets the security requirements of its stakeholders or not.

The layout of the paper is as follows: Section I gives Introduction of CSRF and Broken Authentication and Session Management attack. Section II about Literature Survey, Section III explains the Proposed System . Section IV gives the Conclusion of the study.

II. LITERATURE SURVEY

We highlight the relevant literature survey that uses various techniques to detect CSRF and Broken Authentication and Session Management attacks. The goal of this literature survey is to classify the various techniques and methods that will help identify the related attack to that of the proposed architecture. It provides an appropriate solution that will help to detect the attack and to notify client of the same.

Hossain Shahriar and Mohammad Zulkernine, 2010 [1] explains the identification of CSRF attacks with the motive of visibility and content checking of suspected requests. The working of the system is to intercept a suspected request containing parameters and inputs in the value field. In an open window it relates them with one of the visible forms.

Jinxin You and Fan Guo, 2014 [2] proposed javascript delegation mechanism to combine forms with onfocus and onsubmit events and then dynamically created requests that was effectively handled. The evaluation results show that improved CSRFGuard can be effective to defend CSRF attacks.

Yin-Chang Sung, Michael Cheng, Yi Cho, Chi-Wei Wang, Chia-Wei Hsu, Shiuhpyng Winston Shieh, 2013 [3] presented a technique of light-weight CSRF detection method in which a filtered system is used to check suspicious scripts on the server-side. This technique avoids using script filtering and rewriting approach and uses the method that is based on a new labelling mechanism called Content Box.

Yusuke Takamatsu, Yuji Kosuga and Kenji Kono, 2012 [4] explained a new system of automatically detecting session management vulnerabilities in online applications by performing real time attacks in the simulated environment. Experiments demonstrated that the techniques was able to detect vulnerabilities in some real-world web applications.

Raymond Lukanta, Yudistira Asnar, A. Imam Kistijantoro, 2014 [5] proposed a session management vulnerability scanning tool which was designed and

developed using Nikto and Google Chrome extension. Flaws in vulnerability that can be detected is session management vulnerabilities that includes session fixation, CSRF, and insufficient cookies attributes.

Birhanu Eshete, Adolfo Villafiorita, Komminist Weldemariam, 2011 [6] performed assessment of security misconfiguration vulnerabilities in web server environments. Developed a usable tool to perform automated web security configuration vulnerability auditing, fixing and also safety rating for Apache, MySQL and PHP. Also conducted a detailed evaluation of the tool on eleven real-life online application development and deployment environments.

III. PROPOSED SYSTEM

The main motivation towards the proposed system is to improve online application security, to manage real security risks, to develop a secure software development life-cycle (SDLC), to avoid data breach or data manipulation and also to protect the company brand and to enable new businesses online. Therefore, a new architecture is proposed so that it will help to enhance the security of online web applications.

The proposed architecture(refer Figure: 1) helps to detect CSRF attack and Broken Authentication and Session Management attack.A complete new work flow is designed for Broken Authentication and Session Management attack which overcomes the disadvantages of the existing solutions. The proposed architecture will be implemented using ASP.NET installed on windows operating system. MySQL Server 2008 is needed for the database to maintain a log file of the input request that is generated and processed.

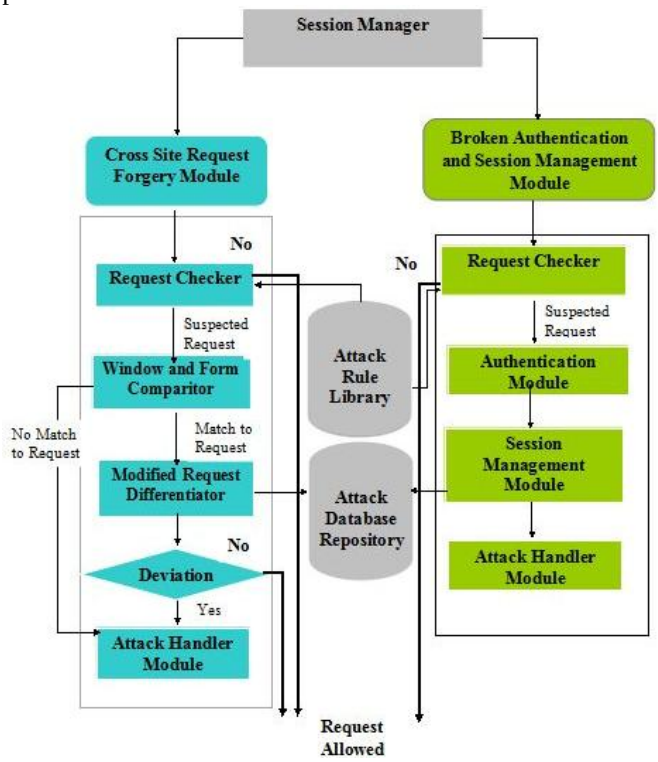

Figure 1: Block Diagram of the Proposed Architecture

A. Cross-Site Request Forgery Attack:

In Cross-site request forgery, the risk of a third-party request to a web application on behalf of an authenticated user who unknowingly executes the link on a malicious site or is the victim of cross- site scripting injecting HTML content to the view to perform the attack is explained. CSRF is an attack against web application users who are vulnerable to attack. The attacker helps a victim's browser to conduct an unwanted action on a legitimate website via a malicious link. Allows an attacker to perform unauthorized activities without the knowledge of the authenticated user. CSRF is also known as Cross-Site Reference Forgery.

1) Role of Session Manager in the Proposed System:

Session Manager helps to manage the web application simulation environment. It guides to detect vulnerability of the incoming request and identify the type of attack if any. It helps to process the incoming request and check for either CSRF attack or Broken Authentication and Session Management attack and alert the authenticated client of the same.

2) Request Checker:

In request checker module, the incoming request from a particular user is checked. The GET and the POST field of the request is checked to identify whether it consist of any parameters or values in the input field. Furthermore, the request checker also helps in the tokenization of the incoming request.

3) Window and Form Comparitor:

Following component evaluates all the open windows and forms that already exist online that are authenticated and are not vulnerable. It checks whether the incoming request matches to any of the open window or forms that is validated. If no match is found then the request is identified to be suspicious and is forwarded to the next module for verification.

4) Modified Request Differentiator:

In modified request differentiator, the requested page is checked. Unvalidated scripts and tags which are not present in the whitelist is identified to be malicious. Attackers mostly inject malicious script in the header fields of the GET and the POST request. Therefore, the script tags of the GET and the POST request should be validated with whitelist which is predefined.

5) Attack Handler Module:

The module identifies a suspected request and generates an alert message to the browser. It alerts the user about the difference between the original request to that of the modified request in the form. Further, the module reports expected and the actual content type mismatch. User makes a decision whether to execute the request or no.

6) Attack Rule Library:

It defines all the possible rules and policies of CSRF attack that is needed to validate the incoming request to check if any malicious script is injected into the request. If any malicious script is found in the incoming request then it is passed on to different validating blocks of CSRF module. If the request does not contain any malicious script then the request is allowed.

7) Attack Database Repository:

This module maintains a log file of all the incoming request and generates a report for further system enhancement.

B. Broken Authentication and Session Management Attack:

A user session is a working context that holds instance specific application data for a user. If a user session is authenticated, then authorization can be enforced so application functionality executes in the boundaries of a user's permissions and privileges.

1) Password Length:

Password should be longer than 10 characters. This becomes difficult for the attacker to crack the password and perform any malicious activity without the knowledge of the authenticated user.

2) Password Complexity:

Every online application should enforce users to create a complex password.

From the below mentioned rules, atleast 3 rules should be satisfied to generate a complicated password:

- Minimum one UPPERCASE character (A-Z)
- Atleast one lowercase character (a-z)
- At the most one digit (0-9)
- One special character. (punctuation)- consider space as special characters too.
- Atleast 10 characters.
- Atmost 128 characters.
- Same two identical characters in a row should not be present. (e.g., 111 not allowed)

3) Password Topologies:

- Commonly used password topology should be avoided.
- Pattern generation of the password should be different every time.
- Passwords should be changed between regular time intervals by the user.

If the password does not match the required criteria of password length and password topology then it is considered to be a weak password that can be attacked by the attacker and so a alert message is send to the client as per requirements. Passwords should be strong enough so that it is not easy for the attacker to manipulate or guess the password via any method of brute-force attack. With regard to implement a pattern of good password, the above mentioned rules should be followed and checked for any vulnerability that can be exploited by the attacker if there is any weakness that is been detected of the user.

4) Session ID Name Fingerprinting:

The pattern in which the session id is generated should not be descriptive. It should be meaningless so that it becomes difficult for the attacker to guess session id.

Web application development framework help to reveal out the session id details. For eg, PHPSESSION (PHP), JSESSIONID (J2EE), CFID and CFTOKEN (ColdFusion), ASP.NET_SessionId(ASP.NET) will help attacker know on which language the web application is developed.

To avoid this, it is suggested to modify the by default session id name that is generated by the web development framework to a generic name.

5) Session ID Length:

Range of the session id should be long enough in such a way that it becomes difficult for the attacker to guess the value of the session id by means of any brute force attack. Values of the session id should be unique for every session so that it is difficult to hack sessions. The value field of the session id should be atleast of 128 bits (16 bytes).

Long session id will help prevent against session fixation attack as it will be difficult for the attacker to guess or perform any brute force attack on session ID is longer than the usual pattern.

6) Session ID Entropy:

The session id generated should be unpredictable so that the attacker finds it difficult to guess the id by means of any statistical analysis technique. ID generated should be unique for every session. This can be done with the help of a Pseudo Random Number Generation where the seed value changes for every session.

7) Content or Value of Session ID :

Session contents or the values of the session id should be meaningless so that it is not easy for the attacker to guess id. If the session id is descriptive and easy to crack then it is easy for the attacker to manipulate the session id and inject malicious script.

One technique to create meaningless session id is to create cryptographically strong session id through the method of cryptographic hash functions such as SHA1 (160 bits). This encryption will help protect the ID content and prevent attackers from inserting malicious script. The algorithm will help validate the Session ID content and will help prevent against session management attacks. Strong session ids will help protect victims navigation onto the web browsers without attackers interference.

IV. CONCLUSION

As CSRF and Broken Authentication and Session Management is considered as one of the top most web application attack, various web applications ascertain the importance of web application security. The proposed architecture will be cost effective as it helps to detect two attacks within the same simulation environment with enhanced rules and policies. With the help of the proposed system, by detecting session management attack it also helps to detect and prevent cross-site scripting (xss) attacks. Therefore, the proposed architecture can further be enhancement to detect xss attacks.

REFERENCES

[1] " Client-Side Detection of Cross-Site Forgery Attacks", Hossain Shahriar and Mohammad Zulkernine, 2010 IEEE 21st International Symposium on Software Reliability Engineering.

[2] "Improved CSRFGuard for CSRF Attacks Defense on Java EE Platform", Jinxin You and Fan Guo, The 9th International Conference on Computer Science & Education (ICCSE 2014)

[3] " Automated Detection of Session Management Vulnerabilities in Web Application", Yusuke Takamatsu, Yuji Kosuga and Kenji

Kono, 2012 Tenth Annual International Conference on privacy, Security and Trust.

[4] "A Vulnerability Scanning Tool for Session Management Vulnerabilities", Raymond Lukanta, Yudistira Asnar, A. Imam Kistijantoro, 2014 IEEE.

[5] "Early Detection of Security Misconfiguration Vulnerabilities in Web Applications", Birhanu Eshete, Adolfo Villafiorita, Komminist Weldemariam, 2011 Sixth International Conference on Availability, Reliability and Security.

[6] " Threat Modelling for CSRF Attacks", Xiaoli Lin, Pavol Zavarsky, Ron Ruhl and Dale Lindskog, 2009 International Conference on Computational Science and Engineering.

[7] " A Privacy-Preserving Defense Mechanism Against Request Forgery Attacks", Ben S.Y. Fung and Patrick P.C.Lee, 2011 International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11.

[8] " Preventing Cross-Site Request Forgery Attacks", Nenand Jovanovic, Engin Kirda and Christopher Kruegel, Technical University of Vienna, IEEE 2006.

[9] " SWART: Secure Web Application Response Tool", Kanika Sharma and Naresh Kumar, 2013 International Conference on Control, Computing, Communication and Materials (ICCCCM)