International Journal of Computer and Internet Security.
ISSN 0974-2247 Volume 8, Number 1 (2016), pp. 25-33

http://www.irphouse.com

Different Threats and Attacks on Online Application: A Survey

Virginia Mary Nadar
Department of Information Technology, PIIT, New Panvel, Mumbai University,

Leena Jacob
Department of Computer Engineering, PIIT, New Panvel, Mumbai University,

Madhumita Chatterjee
Professor, Department of Information Technology and Computer Engineering, PIIT, New Panvel, Mumbai University,

Abstract

Online application security is basically Information Security that focuses on security of websites, web applications and different web services. Because of the advancement in Web 2.0, there is increased information sharing through social networking and increased business adoption over the web as a means of doing business and delivering services, websites are often hacked directly if no proper security measure is implemented. The primary goal of this survey paper is to review few of the online application threats and attacks like Broken Authentication and Session Management, Security Misconfiguration attack, Cross-Site Request Forgery, attack based on Unvalidated Redirects and Forwards and Missing Function Level Access Control attack.

Keywords— Security Test, Broken Authentication and Session Management, Security Misconfiguration, Cross-Site Request Forgery, Unvalidated Redirects and Forwards, Missing Function Level Access Control.

1. INTRODUCTION

A security test is a way of evaluating the security of a computer system or network by

validating and verifying the effectiveness of different web application security controls. A web application security focuses on evaluating the security of different online applications. Testing involves an active analysis of the web applications for any system weakness, design flaws or vulnerabilities. If any security problem is detected, it will be presented to the system owner along with an assessment of the impact and a technical proposal or a solution to mitigate the impact. Security test is also an action to check if an application meets the security requirements as needed of its stakeholders. Vulnerability is defined as a flaw or any weakness in the system's coding, design, implementation or management which can be exploited by the attacker to compromise the system's security objectives.

In Broken Authentication and Session Management the functions related to authentication and session management are often implemented incorrectly, allowing attackers to gain access to passwords, keys or session tokens or to exploit other implementation flaws to assume victim identities. In Security Misconfiguration, a good security requires secure configuration that is defined implemented and deployed for the web application, frameworks, application server, web server, database server and platform. Secure application function settings should be defined, implemented and maintained as defaults are often insecure. Also, additional software should be kept up to date and maintained. Cross-Site Request Forgery (CSRF) attack manipulates a online victim to send a manipulated HTTP request including the victim's session cookie and any other automatically included authentication and personal information of the victim to a vulnerable web application. This helps the attacker to manipulate the victim's browser to generate requests which the vulnerable application thinks are valid requests from the victim. In Unvalidated Redirects and Forwards, web applications frequently redirect and forward user to other vulnerable web pages or websites and use untrusted data to determine the destination pages. Without proper authentication and validation, attackers can redirect victims to phishing/vulnerable/malware sites or use forwards that is created by the attacker to access unauthorized pages. In Missing Function Level Access Control attack, most web applications verify function level access rights and controls before making that functionality visible in the UI. However, web applications need to implement the same access control checks on the server side when each function is accessed. If requests are not verified and checked properly, attackers will be able to forge requests in order to access functionality without proper authentication and validation.

2. BROKEN AUTHENTICATION AND SESSION MANAGEMENT

A session holds information about the user's working context over a particular web application. If the user session is authenticated then authorization can be enforced so that the application functionality executes in the boundaries of the user's permissions and privileges.

Passwords-It should be ensured that passwords are securely saved in an encrypted format in an identity store. Policies should be applied to the quality of passwords that enforces strong password. The life-cycle of passwords should be limited. To change passwords, a single mechanism should be implemented. Applications should not be

storing passwords either in memory or in the database in an unencrypted format.
SSL-The login form of the user must submit the user credentials with transport layer security (SSL) applied to ensure passwords are not sent in clear text format.
Authentication-Several ways are available to authenticate the user such as Java EE basic authentication or using certificates or by single sign-on.

2.1 Vulnerabilities of Broken Authentication and Session Management

A. Session Fixation Vulnerability:

In Session Fixation [2] a victim is forced to use a session identifier (SID) that an attacker has manipulated. A disadvantage of session fixation is that an SID that a web application has issued to a visitor can also be used by other visitor's also. As web applications identify visitor's by their SID's, an attacker can present himself as a visitor after the visitor has logged into the web application with the attacker's SID. This enables the attacker to take control and hack over the victim's account.

B. Cross-Site Request Forgery (CSRF) Vulnerability:

CSRF attacker [2] forces a victim to execute any actions on behalf of the attacker at the target web application. If an attacker can have a victim's browser sending a request on behalf of the attacker while the victim is logging in a web application, the attacker can perform the request with the same privileges as that of the victim.

2.2 Detecting the Vulnerabilities of Broken Authentication and Session Management Attack

A. Detector Work-flow for Session Fixation Vulnerability:

The main problem of session fixation [3] is that the SID remains unchanged even after the authentication process of the victim. According to best approaches and practices, SID should be changed after the change on access level.

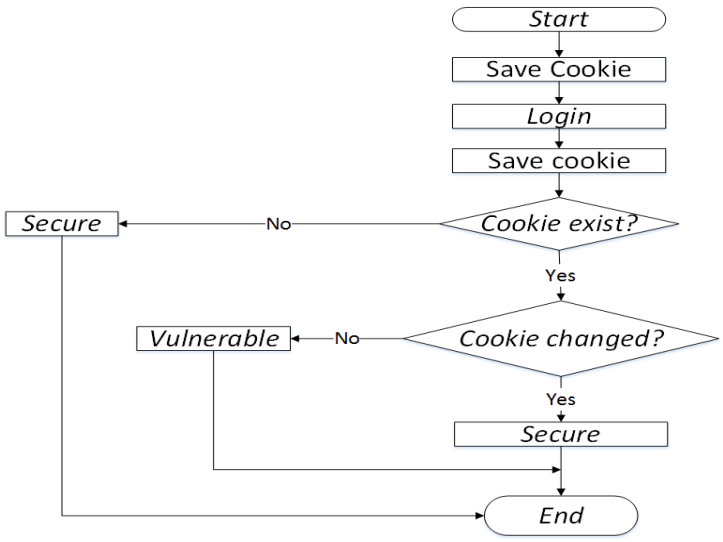

Figure 1: Work-flow for Detecting Session Fixation in Web Browser [3]

B. Detector Work-flow for CSRF Vulnerability:
Based on best practices and approaches [3], CSRF can be prevented by embedding a random token in the HTML form. Every data in the HTML form will be embedded on post data or request body of the HTML form. Detection is done by the analysis component by monitoring and evaluating whether the sent request body contains a random token or not. If there is no random token generated in the sent request of the HTML form, then the request is suspected to have CSRF vulnerability.

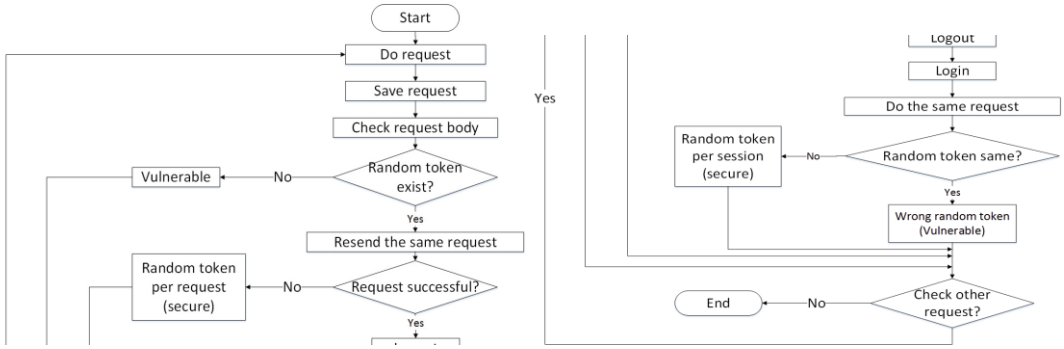

Figure 2: Work-flow for Detecting CSRF in Web Browser [3]

3. SECURITY MISCONFIGURATION

In Security Misconfiguration, [4] attack can happen at any functional level of an application stack including the web application platform, web server, database server, framework and custom code. To avoid security misconfiguration attack, developers and system administrators need to work together to ensure that the entire stack is configured properly with appropriate security measures.

3.1 Vulnerabilities of Security Misconfiguration

Security Misconfiguration attack [4] is mostly associated with Apache, MySQL and PHP (AMP) since it is the most widely used web application server environment as these components are open source. According to a recent survey in January 2011, Apache remains the dominant hosting 59.13% of the websites of NetCraft web server on 273,301,455 different websites. Also, the widespread use of PHP along with MySQL and Apache share almost 75% of live websites on the web application written in PHP.
As the popularity of AMP environment is increasing now-a-days for web application development and management it has made the environment a day-to-day easy target for attackers who can easily take advantage of the potential weakness of which security misconfiguration is one of the most widely attacked vulnerability.

3.2 Detector Architecture and System Details of Security Misconfiguration Attack

The system architecture framework [4] consists of one auxillary component and three main components for Security Configuration Assistant for Apache, MySQL and PHP (SCAAMP) tool. The major components are the Security Configuration Auditor, the

Fixer and the Configuration Safety Rating Module. The auxillary component consist of all the utilities module. The basic workflow of SCAAMP is that a user, that is, a web developer or the administrator if wants to audit or fix a security configuration of a particular AMP server environment then with the the help of the Web UI and Initalizer module, the user provides credentials used to initialize the system (e.g., detection of platform and configuration file paths for each AMP component, permission tokens). As the initialization is over, the user launches the Auditor/Fixer of Apache/MySQL/PHP. Based on the AMP component that is selected, the system will then conduct a security configuration auditing or fixing by invoking the respective module of the SCAAMP architecture.

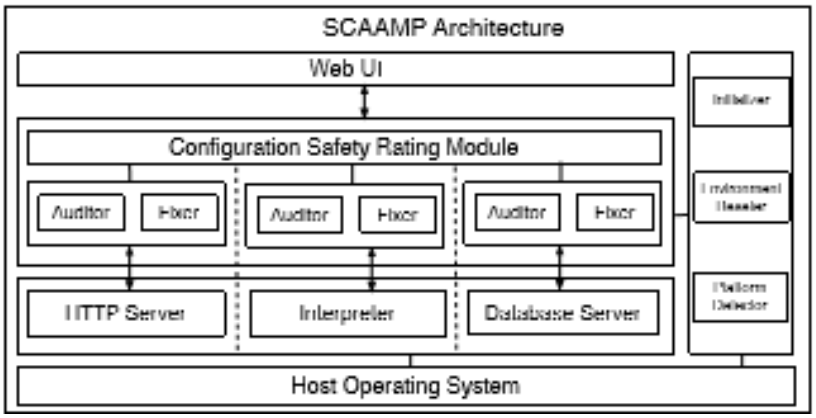

Figure 3: System Architecture of SCAAMP Framework

4. CROSS-SITE REQUEST FORGERY

Cross-Site Request Forgery attack [5] explains the risk of a third-party request to a vulnerable web application on behalf of an authenticated user who unknowingly executes the link on a phishing site or is the victim of cross-site scripting, injecting malicious HTML content to perform the attack. The attacker forces the victim's browser to perform an unauthorized action on a trusted website with the help of a malicious link or other content. CSRF is also known as Cross-Site Reference Forgery.

4.1 Detection Framework for CSRF Attack

In CSRF Attack Detection Framework [6] it helps to detect the attack over a trusted website which is viewed by a user in a window after performing an authentication process and the session information is saved in the browser. The components of the detection framework are as follows:

1) Request Checker Module:

Request Checker module [6] receives a request that is launched by one of the pages present in a browser. It then checks the request type by examining the header of the request.

2) Window and Form Checker Module:
This module [6] will examine all the open windows and will identify whether any of the open window is displaying a page that is downloaded from the destination domain.

3) Request Differentiator Module:
The following module [6] is responsible for modifying an original request of the user by removing either all or hidden parameters and values.

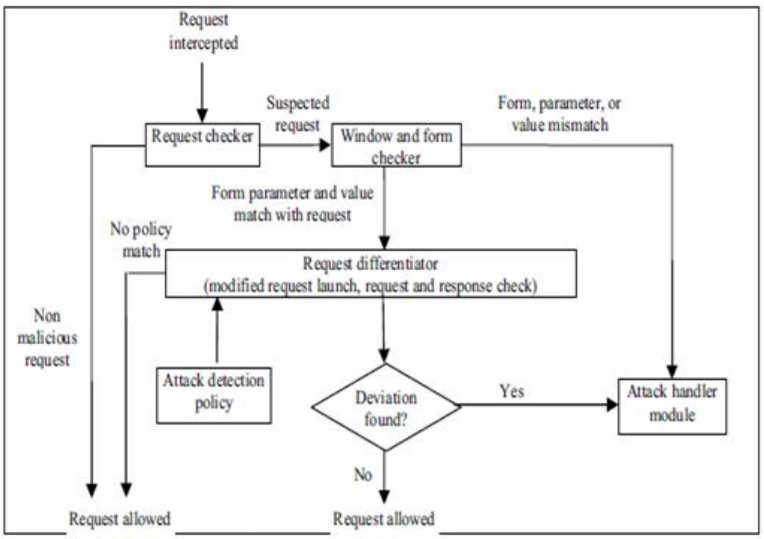

Figure 4: Detection Framework for CSRF Attack [6]

5. OTHER ONLINE APPLICATION ATTACKS

5.1 Unvalidated Redirects and Forwards

Users of different web applications are frequently redirected and forwarded to different web pages over different websites and if the web application is vulnerable to a certain attack, the attacker can gain access to untrustworthy data to determine the destination pages. Without proper authentication and validation, attackers can redirect victims to phishing, vulnerable/malware sites or use forwards to access unauthorized pages.

A. Preventive Techniques:

In-order to redirect a user directly to another web page without an action of the user such as clicking on a hyperlink in between web application traversal, following line of code can be implemented in the web application design framework

- Java: response.sendRedirect(http://www.virginia.com)
- In PHP: <?php/* Redirect browser */ header (“Location:??www.virginia.com/”);?>
- In ASP.NET: Response.Redirect (“~/folder/login.aspx”)
- In Rails: redirect_to login_path

5.2 Missing Function Level Access Control

In Missing Function Level Access Control, vulnerabilities could exist because of insufficient protection of sensitive request handlers within web application programming. If requests are not authenticated and verified properly, it will become easy for attackers to manipulate the requests in order to access unauthorized functionality.

A. Preventive Techniques:

- Designing of process for managing entitlements and also to ensure that update and audit can be done easily.
- The enforcement mechanisms should deny all access by default and should consist of only explicit grants to specific roles and functionalities for access to every function.
- If the function or a method is involved in a particular work-flow, necessary check should be done to ensure that all the conditions are in the proper state of flow to allow access.

6. COMPARATIVE ANALYSIS

Different online application attacks such as Broken Authentication and Session Management, Security Misconfiguration, Cross-Site Request Forgery, Unvalidated Redirects and Forwards and Missing Function Level Access Control were analyzed based on different web applications and approaches and an comparative analysis is designed as follows.

Table 1: Comparative Analysis of different Web Applications against Web Application Threats

SN	Web Applications	Web Application Threats				
		CSRF	Security Misconfiguration	Broken Authentication and Session Management	Missing Function Level Access Control	Unvalidated Redirects and Forwards
1	Company Websites	✓	✓	-	✓	-
2	Entertainment Websites	-	-	-	-	✓
3	Online Banking	✓	-	✓	✓	-
4	Government Websites	✓	✓	✓	-	-
5	Mails & Messages	✓	-	-	-	✓
6	E-Business Websites (B2B, B2C, C2C)	✓	✓	✓	✓	✓
7	Educational Websites	✓	-	✓	-	✓

Table 2: Comparative Analysis of Broken Authentication and Session Management and CSRF attack based on different approaches

SN	Different Approaches	Attack Vectors (Ĕxploitability)	Security Weakness		Impacts	
			Prevalence	Detectability	Technical	Business
1	Automated Detection of Session Management Vulnerabilities in Web Application.	Easy	Common	Easy	Moderate	Online Banking, Government Websites, E-Business Websites, Educational Websites
2	A Vulnerability Scanning Tool for Session Management Vulnerabilities.	Easy	Common	Easy	Moderate	
3	Threat Modeling For CSRF Attacks.	Average	Widespread	Difficult	Severe	Official Websites, Government Websites, Online Banking, E-Business Websites, Mails & Messages, Educational Websites
4	Client-Side Detection of Cross-Site Forgery Attacks.	Easy	Common	Average	Moderate	
5	A Privacy-Preserving Defense Mechanism against Request Forgery Attacks.	Average	Very Widespread	Average	Moderate	
6	Preventing Cross-Site Request Forgery Attacks.	Difficult	Widespread	Average	Moderate	

7. CONCLUSION

From the survey study, we can analyze and conclude the different possible attacks and threats on online applications and related countermeasures. It covers different technological as well as organizational aspects that focuses on real-life risks and not just legal issues. The paper provides survey review on how to implement effective privacy techniques by using different approaches in web application security with the aim of helping and assisting developers and online application providers to better understand the web application design and improve privacy by referring to various techniques and methods.

References

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

[2] "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.

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

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

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

[6] "Client-Side Detection of Cross-Site Forgery Attacks", Hossain Shahriar and Mohammad Zulkernine, 2010 IEEE 21st International Symposium on Software Reliability 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] "Reliable protection against session fixation attacks", M. JOHNS, B. BRAUN, M. SCHRANK, AND J. POSEGGA, ACM Symposium on Applied Computing (2011), pp. 1531–1537.

[10] "Client side protection against session riding", M. JOHNS, AND W. JUSTUS, In Proc. of OWASP Europe Conf., refereed papers track, Report CW448 (2006), pp. 5–17.

[11] "New Threats and Attacks on the World Wide Web", T. Holz, S. Marechal, and F. Raynal, IEEE Security and Privacy, vol. 4, pp. 72–75, March 2006.

[12] "Research Directions in Web Site Evolution II: Web Application Security", P. Tramontana, T. Dean, and S. Tilley, 2007 9th IEEE International Workshop on Web Site Evolution. IEEE Computer Society, 2007, pp. 105–106.

[13] "An Introduction to A Common Web Application Weakness", J. Burns, Cross Site Request Forgery, White paper, Information Security Partners LLC.

[14] "Cross-Site Request Forgeries: Exploitation and Prevention", W. Zeller and E. Felten, Technical Report, Princeton University, October 2008.