Scientific Journal Impact Factor (SJIF): 1.711

e-ISSN: 2349-9745
p-ISSN: 2393-8161

International Journal of Modern Trends in Engineering and Research
www.ijmter.com

A Review paper on Securing PHP based websites From Web Application Vulnerabilities

Deven Gol[1], Nisha Shah[2]
[1] Information Technology, The Gujarat Technology University SVIT,
Vasad, Gujarat 388306, India
[2] Hod, Department of IT, Gujarat Technology University SVIT,
Vasad, Gujarat 388306, India

Abstract: In today's Era, Web applications are one of the most part ubiquitous platforms for information sharing and services over Internet which play significant role in individual life as well as in any country's growth. Web applications have gone through a very rapid Growth As they are increasingly used for the financial organization, government, hospitality and many critical services. Web applications become a popular and precious target for security attacks. at the present time, billions of transactions are done online through net banking, online shopping, online billing and many more. Even though these applications are used by lots of people modern web applications often implements the complex structure requires for user to carry out actions in given order, in many cases the security level is too low, which makes them vulnerable to get compromised. Even though a large number of techniques have been developed to build up web applications and mitigate the attacks toward web applications, there is little effort constant to drawing relations among these techniques and building a big picture of web application security(WAS) research. In this paper, we present a survey on various types of web application vulnerabilities(WAV).

Keywords: *Web application vulnerability, Firewall, SQL Injection, XSS.*

I. Introduction

Although traditional firewalls and IDS have efficiently prohibited network-level attacks, most upcoming attacks will be at the application level, where existing security mechanisms are insufficient to provide proper security. The application level security inherent in the web application's code, despite of the technology used for web application development or the security of the web server or the database on which it is built. But the vulnerabilities are still present in the application as the firewall or the intrusion detection systems keep open the port 80 and 443 for online transaction purpose.
The web application provides web pages which contain the images, HTML, script, etc. as grow to be more user-friendly but also exploits the web security vulnerabilities. Now days the web applications like financial purpose, healthcare purpose, government sites, etc. are interact with the back-end database for fulfill client's request response.
If the security of web application is compromised that will result in financial transaction, informational, ethical, legal cost issues. The protection of the web application is most significant, according to the details by Web Application Security association, about 49% web application

contains the high severity level vulnerabilities and 13% of them are automatically get compromised for the security vulnerabilities. The unsecure web application directs to the well-known security vulnerabilities such as SQL Injection, cookie pilfering, security miss configuration, (CSRF)Cross site request forgery, (XSS)Cross-site scripting, session hijacking, worm's attacks, Trojans etc.

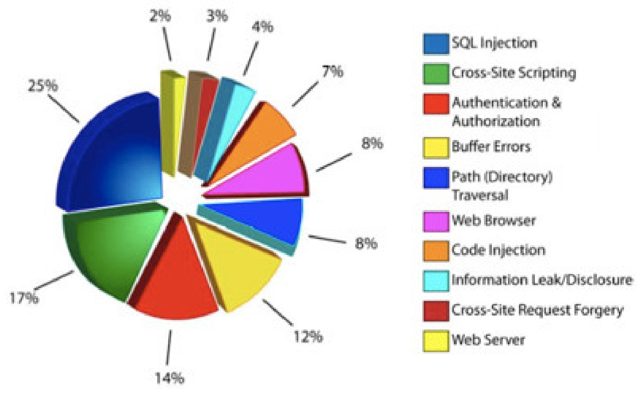

Fig 1: Amount of occurring Different types of Vulnerability chart[18]

The significant amount of research had been devoted into hardening and mitigating the different vulnerabilities of web application. Many of these solutions produce based on some assumption on the web technologies used in application development and mitigate the concern security flaws. The probability of these solutions can be applied to other similar type of concern may be very significant amount due to narrow mind approach or practitioner may focus on provided that an correct solution to particular technology related vulnerability only [2].

II. Understand WAS vulnerabilities

A secure web application has to satisfy desired security properties under the given threat model. In the area of web application security, the following threat model is usually considered: 1) the web application it-self is compassionate (i.e., not hosted or owned for malicious purposes) and that is hosted on a trusted and hardened infrastructure. 2) The attacker is able to influence either the contents or the sequence of web requests sent to the web application, but can't directly concession the infrastructure or the relevance code. The vulnerabilities within web application implementations may disobey the deliberated security properties and permit for consequent successful exploits.
Many of these techniques make assumptions on the web technologies used in the application development and only address one particular type of security flaws; their prototypes are often implemented and evaluated on limited platforms. A practitioner may wonder whether these techniques are suitable for their scenarios. And if they can't be directly applied, whether these techniques can be extended and/or combined. Thus, it is desirable and urgent to provide a systematic framework for exploring the root causes of web application vulnerabilities, uncovering the connection between the existing techniques, and sketching a big picture of current research frontier in this area. Such a framework would help both new and experienced researcher to better understand web application security challenges and assess existing defenses, and inspire them with

new ideas and trends. In this paper, we survey the state of the art in web application security, with the aspire of arranging the existing methods into a big picture that promotes future research. In current web development practices, distillation routines are usually placed by developers manually in an ad-hoc way, which can be either imperfect or incorrect, and thus introduce vulnerabilities into the web application [16].
As shown in fig 2 is created by Web Hacking Incident Database for 2013 (WHID), which clearly shows different attack methods by hijackers that are the most popular.
In this paper we have described some of the most common areas of vulnerability found in PHP web applications as well as suggestions on how they can be managed and prevented.
By showing you about allegation of flaws, and how each particular error can be exploited, we will understand not just how to avoid these particular mistakes, but also why they result in security vulnerabilities that harm the web application.

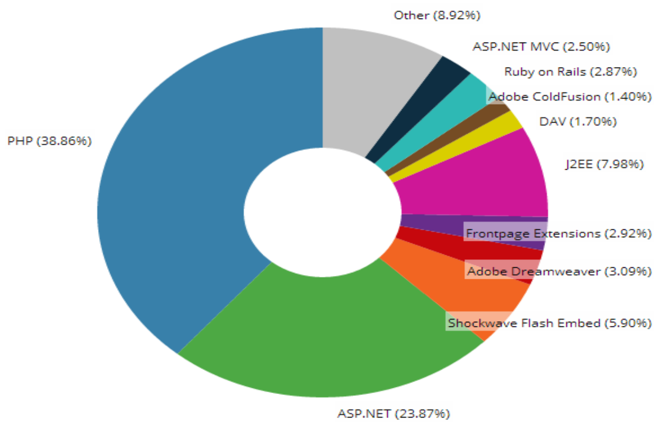

Fig 2: Comparison by Programming language[19]

Missing distillation allows trusted web contents without validating the system that affected by malicious user input; improper developed web application allows bypass the validation procedure given by malicious user. A web application with the explained vulnerabilities not succeed to achieve the input verification property, thus is vulnerable to a class of attacks, which are referred to as script attacks, SQL injections, Session Hijacking attacks or input validation attacks. This type of attacks embeds malicious contents within web requests, which are exploited by the web application and executed later on. Examples of input validation attacks include cross-site scripting (XSS), SQL injection, session and cookie hacking, path directory traversal, enclosure of filename, etc. In the following, we demonstrate the most five popular input validation attacks[9].

2.1 *SQL Injection*

SQL injection attack is a code injection technique successfully launched when malicious contents within user input flow into SQL queries without correct validation in server's database [7]. The database trusts the web application and without correct validation executes all the SQL queries issued by the application. Using this attack, the attacker is able to embed SQL keywords or operators within user input to manipulate the SQL query structure and result in not deliberated

execution. Chain Reaction of SQL injections includes authentication bypass, disclosure of information, and denial of service attack and even the damage of the entire database.
For example:

An vulnerable SQL Query statement would be:
"SELECT * FROM login WHERE U_name = 'deven' ";

An SQL Injection query will be outcome in the following given Query:
"SELECT * FROM login WHERE U_name = " or ' 1 = 1'";
The end result that provided here will be mostly correct, and thus the content of entire login table would be displayed even with sensitive disclosure of information. Proposition for this type of PHP security issue, Hackers can gain access in to the database which provide all the information in the database including Usernames, Id, passwords and many more, even sensitive information as well. Additionally, It can be altered or dropped which could obviously have terrible effects on the front end of website and the entire website can be defaced if attacker gains administrative privileges for all the information in the hacked databases[11].

2.2 Cross Site Scripting (XSS)

A cross-site scripting (XSS) attack is successfully launched when the user supplied data from web pages without proper validation or evading that content. XSS enables attacker to inject client-side script into web pages that exploit the interpreter in the browser. Using this attack, the attacker is able to execute inject malicious scripts into web, which get access within the victim's web browser.

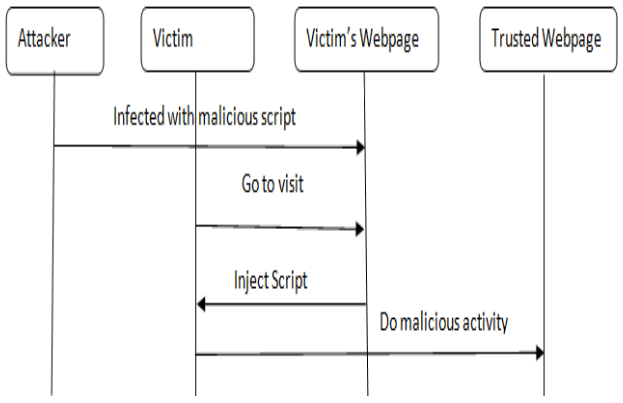

Fig 3 : Sequence view of XSS attack

The most common consequence of XSS is the disclosure of sensitive information, e.g., session hijacking, cookie theft. XSS usually provided as the first step that enables promote sophisticated attacks. There are several options of XSS attacks, according to attacker's injected scripts, including stored or persistent XSS (most probably malicious scripts are injected into continual storage), reflected XSS, DOM-based XSS, content-sniffing XSS, etc. The pattern of XSS attack is shown in the figure no 3. Once the attacker got succeed in injecting malicious code inside the

stored database of the website then it will displayed to the victim, even code can also be run inside the trusted site which steals cookies, and sends important information such as session ID. The resulting page can be now distorted to a malicious third-party website. It can also be used for things such as redirecting users to a spam website or in a more sophisticated attack like key logging
(sending the user's keystrokes to an external database) through JavaScript into the HTML source. XSS is also used for user-account hacking though authentication bypass and stealing the secret information of users. A Cross-Site Request forgery method is also known as concurrence of XSS. This is the method which provides malicious code tricks to the users' browser to send requests under the assumption of legal user; for example, it can use a users' online bank account to perform transactions without user's knowledge [8].

2.3 Session and Cookie Hacking

Session hijacking is the exploitation of a valid computer session, it is also called a session key (session ID) to gain unauthorized access to confidential information or services in a system [7].
The session and cookie hacking can't violate the database directly or the web application, but it can compromise the user accounts for malicious purpose. A session is an entity triggered when users initiate contact with a web server which consists of some duration of period interaction between users and web application. This is mostly possible through or authenticated using security procedures like confidential information such as a username and password. During this particular session, the web application will stores a cookie inside the browser or file on the users' browser which will contain critical information about the session like the users' preferences, authentication related information, unique codes or online transaction related information.

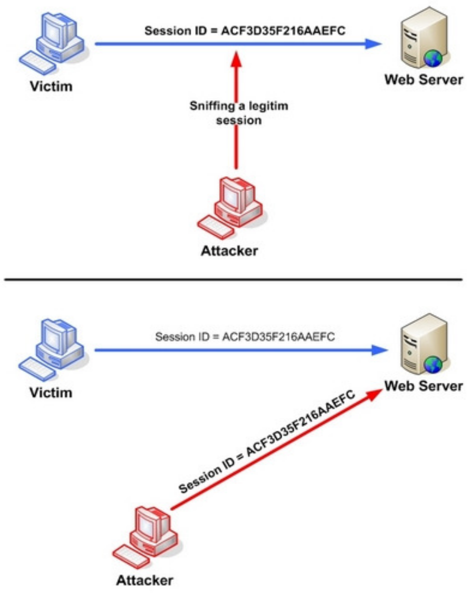
Session ID = ACF3D35F216AAEFC
Victim
Web Server
Sniffing a legitim session
Attacker
Session ID = ACF3D35F216AAEFC
Victim
Web Server
Session ID = ACF3D35F216AAEFC
Attacker

Fig 4: The session hijacking attack[9].

In fig 4, as we can see that the first attacker captures a valid token session called "Session ID", then he uses the valid token session to gain unauthorized access to the Web Server. This can provide full access to hacker.

There are four main methods which is used for a session hijacking listed below:
1. Cross-site scripting.
2. Session side jacking.
3. Attempt to steal the session key
4. Session fixation.

Proposition for this type of PHP security issue When user login inside the website, a session ID is created, which can be obtain the legitimate user's session ID and user's secret information for malicious purpose by hacker. When the hacker tries to reuse that session ID, it is known as session fixation, and which will provide hackers to login as an authentic user and cause alteration inside the users account or even damage the user's account, which is very dangerous when the user contain some sensitive information like nation security or someone whose account contains important information or sensitive data.

2.4 Remote File Inclusion and Remote Code Execution

The Remote file inclusion vulnerability exist when hackers to do malicious things through inserting a malicious script inside your web server to execute the wicked code. This can lead to further types of vulnerabilities of the web application. Not just does this tolerate assessment of remote hostile scripts, it can also be used to access local file servers. It is important to know that most of the time file inclusion attacks can be possible by turning register global off. Because of improper input validation most of the vulnerability occurs which provides

Types of inclusion are listed below: a simple way for hackers to install a malicious script or file onto the web server.

Local file inclusion:

I. Local File Inclusion is a type
of vulnerability most probably found on websites
This would not configure properly. It allows an attacker to include local files on the server located at his own system. The vulnerability is also due to the improper use of input validation.

II. Remote file inclusion: Remote File Inclusion is similar to Local File Inclusion vulnerability except instead of including local files situated in current system; it includes remote files which are situated on Remote machine. I.e. files on the Remote server can be included through a malicious script on the web server.

Proposition for this type of PHP security issue this vulnerability allows an attacker to run uninformed, system level code on the susceptible server and rescue any preferred information contained by the server. Improper validation in coding errors direct to this vulnerability. Sometimes, it is complicated to determine this vulnerability for the period of penetration testing but such troubles are often exposed while doing a source code evaluation. Web application is essential to retain information that abuse of this vulnerability can lead to total system compromise with the same rights as the Web server itself. Remote file inclusions lead to remote code execution on the server, data stealing, and script executed on the client-side that lead to troubles such as XSS and denial of access. The remote code execution on the server side means that if the file that the hacker has included is a shell prompt, and then the hacker can execute system-level code on the server and use this to alter or retrieve data on that web server or hack the end-user's terminal. Client side remote script execution leads to cross site scripting (XSS) which is so dangerous for further vulnerabilities. Denial of access attacks prevents the legitimate user of a website from being able to access the resources of the website by preventing the servers from serving web pages, or it can overload the server with requests that results slow-down, making the website inaccessible.

III. Proposed Preventive Measures for WAV

3.1 SQL Injection

To provide the prevention of this vulnerability, the data must be validated, verified and cleaned up with proper coding before it can enter inside the application. All the susceptible information such as passwords should be encrypted using MD5 or SHA. Technical information can occasionally contain technical details that might disclose web application vulnerabilities to an attacker. Therefore, it must be discarded from error messages. An attacker purposely looks to error messages to get information such as database related information, usernames, passwords of admin user and table name, for this reason, you should disable error messages or you can create your own convention error messages for validation purpose.

You can also bound the permissions approved on the database and if possible provide very few permissions results which will be result as a lower chances of attack in application. You may use stored procedures and beforehand defined cursors to abstract data access from the users who do not get directly access into the tables or views of database. You can also avoid keywords such as 'INSERT', 'UPDATE', 'DROP', and 'UNION' from being added to the database which can alter the whole database and affect the integrity of information.

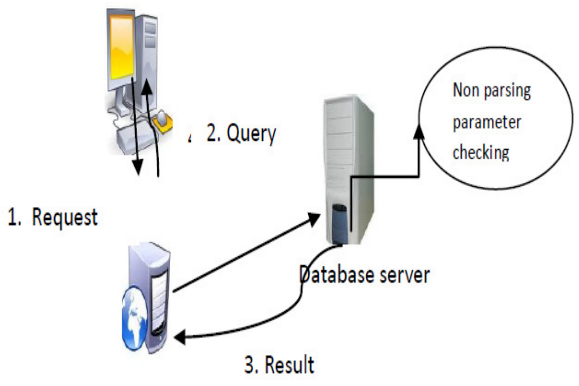

Fig. 5 Protection against DOS attack

3.2 XSS(Cross Site Scripting)

The solution for the prevention of XSS is to use escape functions properly, particularly to use escape characters that consist of HTML and JavaScript language rules like '>' or '<' or to translate these into HTML entities. In websites, where users want to post HTML like links, an alternative syntax like bbcodes can be used in order to overcome the escaping of HTML characters. Based on the actual testing and systematic analysis of the major browsers, the attack vectors are transformed in the following three techniques:*1) Special characters*. Such as newline, Tabs and so on;*2) Changes in the quote form*. For example, change the double quotes to single quote or backtick; *3) Decimal encoding*. Attack vectors are encoded by decimal encoding to bypass code checks. The htmlspecialchars () function indentifies any output you don't want to be

HTML output and converts it into plain HTML entities, for example: '&' becomes '&' and ''' (double quote) becomes '"'. You should carefully test the website before launching it.

3.3 Session and Cookie Hacking

To prevent attackers from setting session ID's prior to login, ID's should be changed often, therefore, the session_regenerate_id() function should be used every time the user logs in, assigning them a fresh ID;The risk of this hacking can be mitigated by revalidating a user who is about to perform important or sensitive tasks like resetting their password (i.e. by making them re-enter their old password);If user's password is to be stored in a session variable, it must be encrypted (using the sha1() function)If your web application is handling sensitive information like debit and credit card numbers, then using an SSL or any other secured connection can also prevent session and cookie hacking.

3.4 Remote File Inclusion and Remote Code Execution

You should make OFF The register_globals directive, in later versions of PHP, it is OFF by default, but one should always check that. If the directive has to be set to ON for some reason, all variables must be properly initialized. There are other PHP directives can be used to prevent this security breach, which include:allow_url_fopen (set to on by default) that controls whether remote files should be includable and should be turned to OFFallow_url_include that controls whether include_once(), include(), require() and require_once() commands are able to include remote files into the code Enabling safe_mode which forces PHP to test of user ID permissions before opening any file Always validate user input and be very careful with any data retrieved from remote servers or locations. In order to stop it, you can ensure that all include files are locally hosted and never accept files from anywhere else unless absolutely necessary Restrict user privileges to an absolute minimum can also help in prevention from this security threat[9].

4. Conclusion

This survey paper provided a recent research results in the area of web application security. PHP has developed into the majority of well-liked programming language which is widely used for rapid development of dynamic websites for this digital world. Because of the web servers get easily access, they present important role in security of web application vulnerabilities. Security is a process, not a product so by adopting a well research approach to web application security will allocate you to produce higher and more robust code for particular application to become secure. In this survey paper, I have mentioned the top most five PHP issues of web application, if we are not be careful to avoid this issues web sites can be hacked easily and malicious activity can be done by attacker resulted in bad activities. We illustrate characteristics of secure web application development, identified significant security assets that secure web applications.

References

[1] Verizon 2010 Data Breach Investigations Report "http://www.verizonbusiness.com/resources/reports/rp 2010 databreach-report en xg.pdf.
[2] WhiteHat Security, "WhiteHat website security statist ic report 2010."
[3] WhiteHat Security, "WhiteHat website security statistic report 2010."
[4] J. Bau and J. C. Mitchell, "Security modeling and analysis," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 18–25, 2011.

[5] H. J. Wang, C. Grier, A. Moshchuk, S. T. King, P. Choudhury,and H. Venter, "The multi-principal os construction of the gazelle web browser," in *USENIX'09: Proceedings of the 18th conference on USENIX security symposium*, 2009, pp. 417–432.
[6] https://www.owasp.org/index.php/Top_10_2013
[7] https://www.owasp.org/index.php/Crossite_Scripting_(XSS)
[8] 1.Antunes, N. and M. Vieira, "Defending against Web Application Vulnerabilities." Computer, 2012. 45(2): p. 66- 72.
[9] http://blog.templatemonster.com/2014/05/08/php-security-issues/
[10] Monika Sachdeva et al, "Performance Analysis of Web Service under DDoS Attacks", 2009 IEEE International Advance Computing Conference (IACC 2009) Patiala, India, 6-7 March 2009
[11] J. Dhanamma, and T. Rohini, "The Unified Approach for Organizational Network Vulnerability Assessment", IJSEA,Vol 4, No.5, September 2013.
[12] V. Nithya, R.Regan, J.vijayaraghavan,"A Survey on SQL Injection attacks, their Detection and Prevention Techniques""International Journal of Engineering and Computer Science April, 2013.
[13] H. Tian, X. Jing, L. Kunmei, and Z. Ying, " Strong association rule based web application vulnerability detection" , ICCSIT 2009
[14] S. Splaine, "Testing Web Security: Assessing the Security Of Web Sites and Applications", Wiley Publication.
[15] Bisht, P., P. Madhusudan, and V.N. Venkatakrishnan, "CANDID:Dynamic candidate evaluations for automatic prevention of SQL injection attacks." ACM Trans. Inf. Syst. Secur., 2010. 13(2): p. 1-39.
[16] N. Swamy, B. J. Corcoran, and M. Hicks, "Fable: A language for enforcing user-defined security policies," ,Oakland '08: Proceedings of the 29th IEEE Symposium on Security and Privacy.
[17] Xiaowei Li and Yuan Xue, "A Survey on Server-side Approaches to Securing Web Applications", ACM Computing Surveys (to appear in vol 46, issue 4)
[18] http://www.dailytech.com/Security+Study+Lists+Firefox+Most+Vulnerable+Browser+IE 8+Among+the+Safest/article16796.htm
[19] http://www.troyhunt.com/2013/08/its-time-to-hack-yourself-first-with.html