ISSN: 2277-9655
Impact Factor: 4.116

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

DEVELOPING A SECURE WORDPRESS WEBSITE AND HOSTING ON NIC PLATFORM

Rohit Kumar*
Assistant Engineer, Uttarakhand Council for Biotechnology, Haldi-263146,
Pantnagar, U. S. Nagar, Uttarakhand, India

DOI: 10.5281/zenodo.60877

ABSTRACT

Now-a-days most people explores the web at the end of the when a majority of business are closed and the reason being that websites are always there. A website allows us to do business for 24 hours a day, 7 days a week, 365 days a year. If your competitor has a website then they will have a definite advantage to influence your business or even if you have an informative website then also it will help you to make your stakeholders aware about the progress of your organization. Moreover with powerful search engines it is easier to locate your organization online. A professional website improves your public image and your customers/stakeholders will have more confidence in your organization. Most small businesses are only able to market to their town and surrounding communities. With a website, you can take your products and services globally.

With the edges of having a website it is required to develop a website but to develop a website it is an obvious thought that the knowledge of HTML, CSS, PHP, MYSQL, etc. is a must. This is not a necessity to have absolute knowledge of above languages although it is advantageous if you possess that, the reason being the availability of several open source content management system in the public domain some of them are wordpress, joomla and drupal. The basic knowledge of above CMS will help you developing websites in minute or hours.[1][2]

Now the other side of website development emphasizes the security constraint or the recovery after hack which is the common issues now-a-days with daily advancements in web technologies. Although there is a lot of online support for these abrupt issues but still a smart knowledge is required to deal with these issues to have a healthy website.

In the proposed work, the limitation associated with the website are studied and removed. The complete method was presented that how the website got secured after vulnerability detection.

KEYWORDS: wordpress, hack, security threat, website.

INTRODUCTION

In the past years, several days were required for the development of a professional website and a professional web developer team or expert was required for the same. But now the technological advancement brought us with several open source content management software platform such as wordpress, joomla and drupal eliminating the issue of the requirement of professional knowledge of website development but only some basic knowledge will drive us to create a professional website. Wordpress is considered in this paper as the work of the author http://www.ucb.uk.gov.in is in wordpress only. Firstly the security assessment report have been elaborated in this paper and thereafter the details of bugs fixation and verification have been described.[2]

WORDPRESS

WordPress is a free and open-source content management system (CMS) written in PHP and based on MySQL licensed under the GPL.To create a website using wordpress the wordpress is installed on a web server. The basic features of wordpress includes a plugin architecture and a template system. Recently wordpress have been used by more than 26.4% in top 10 million websites. Being the easiest, wordpress is very popular blogging system on the Web. WordPress allows users to create and edit websites from a central control panel, which includes a text editor for modifying content and menus to change various design elements. WordPress provides various plugins in WordPress Plugin Directory, and it is quite easy to upload and promote plugins in WordPress. Users can write posts in this platform and other people can write comments about the post.[3]

WordPress has a web template system using a template processor having files size of about 20 MB. WordPress users can use different themes which allows to change the look and functionality of the website. The plugin architecture allows users to extend the features and functionality of a website or blog. WordPress also features integrated link management; a search engine–friendly, clean permalink structure; the ability to assign multiple categories to articles; and support for tagging of posts and articles. WordPress Multisites allows multiple blogs to exist within one installation but is able to be administered by a centralized maintainer.[3]

PROBLEM FORMULATION AND OBJECTIVES

In the past, millions of user implemented their website using wordpress. We also implemented our website http://www.ucb.uk.gov.in using wordpress open source content management system. In this work, we have implemented the website including security measures after vulnerability detection. In this first method, we will focus upon the level 1 vulnerability report including OWASP, SANS/CWE, WASC vulnerability standards[4]. In the second method we will include the ways to remove existing vulnerability. In this third method, we will check the website performance after the vulnerability patch-up .

Objectives for this work are:

1. To implement a website using wordpress custom design.
2. The main focus of this research is the detection of vulnerabilities to reduce the problem of hacking.
3. The objects of interest is testing on security platform.
4. To analyze the use of several plugin in performance enhancement.
5. To detect the recovery and security performance after hack.

RESEARCH METHODOLOGY

The basic idea of secure designing of wordpress 4.2 is to create a website bugs free, then to update the website as soon as new bugs are reported. Till this work was completed now the wordpress have been updated to the version 4.5.2 which had fixed most of the reported issues.

In this research, we have followed the given steps to implement the research problem.

Step1: First we developed the website using wordpress 4.2.
Step2: Secondly we had undergone level 1 security audit to determine the vulnerabilities.
Step3: Then we implemented the required changes as per the fixation of the bugs.
Step4: Thereafter we have level 2 security assessment to determine if any vulnerability persists.
Step5: Finally, the website got hosted on http://www.ucb.uk.gov.in .
Step6: Recent verification from online malware detecting softwares.

3.1 Security Audit Level 1 report: The website with wordpress 4.2 has following vulnerabilities according to the NIC website security audit vendor's report:

ISSN: 2277-9655
Impact Factor: 4.116

Table 1: Details of initial security audit report

Vulnerability Name	Severity	Status
Improper Session Termination	High	X
Stored Cross-Site Scripting	High	X
Session Fixation	High	X
File Upload Vulnerability	High	X
HTML Injection	High	X
Iframe Injection	High	X
Insufficient Transport Layer Protection(SSL/TLS)	Medium	X
Improper Cache Control	Medium	X
Login Brute Force	Medium	X
Cookie Secure Attribute Not Present	Low	X
Auto-Complete Feature Enabled	Low	X
HTTPOnly Attribute Not Present	Low	X
Weak Password Policy	Low	X
Http Method Enabled	Low	X

X = Found

Fig 1. Vulnerabilities by OWASP Top 10 2013

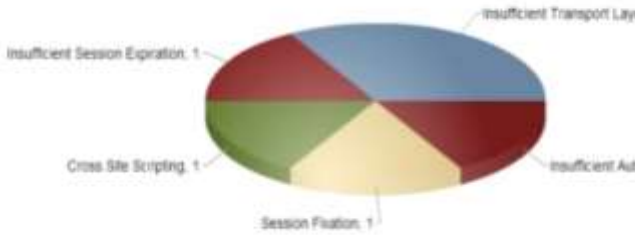

Fig 3. Vulnerabilities by SANS/CWE Top 25

Fig 4. Vulnerabilities by Category

Vulnerability Details with Evidences

1. Improper Session Termination: Improper session termination is a vulnerability which relates to the session management module of the application. The applications vulnerable to the improper session termination, do not properly terminate the application session due to which the authenticated pages in the application can later be browsed by pressing the browser's back button.[5]

Fig 5. Improper session termination vulnerability

2. Stored Cross-Site Scripting: A stored XSS attack occurs when a web application gathers a malicious input from user and stores the malicious input inside the vulnerable webpage permanently. Thus, this malicious code will be executed every time a user visits the vulnerable webpage. The input that is stored is not correctly filtered. As a consequence, the malicious data will appear to be part of the web site and run within the users browser under the privileges of the web application. For example: 1. Application offers a form to users to post their questions about the application and give feedback. 2. In social networking application, where a user can see other user((s profiles. 3. Comment form in application where user can post their comments.[6]

ISSN: 2277-9655
Impact Factor: 4.116

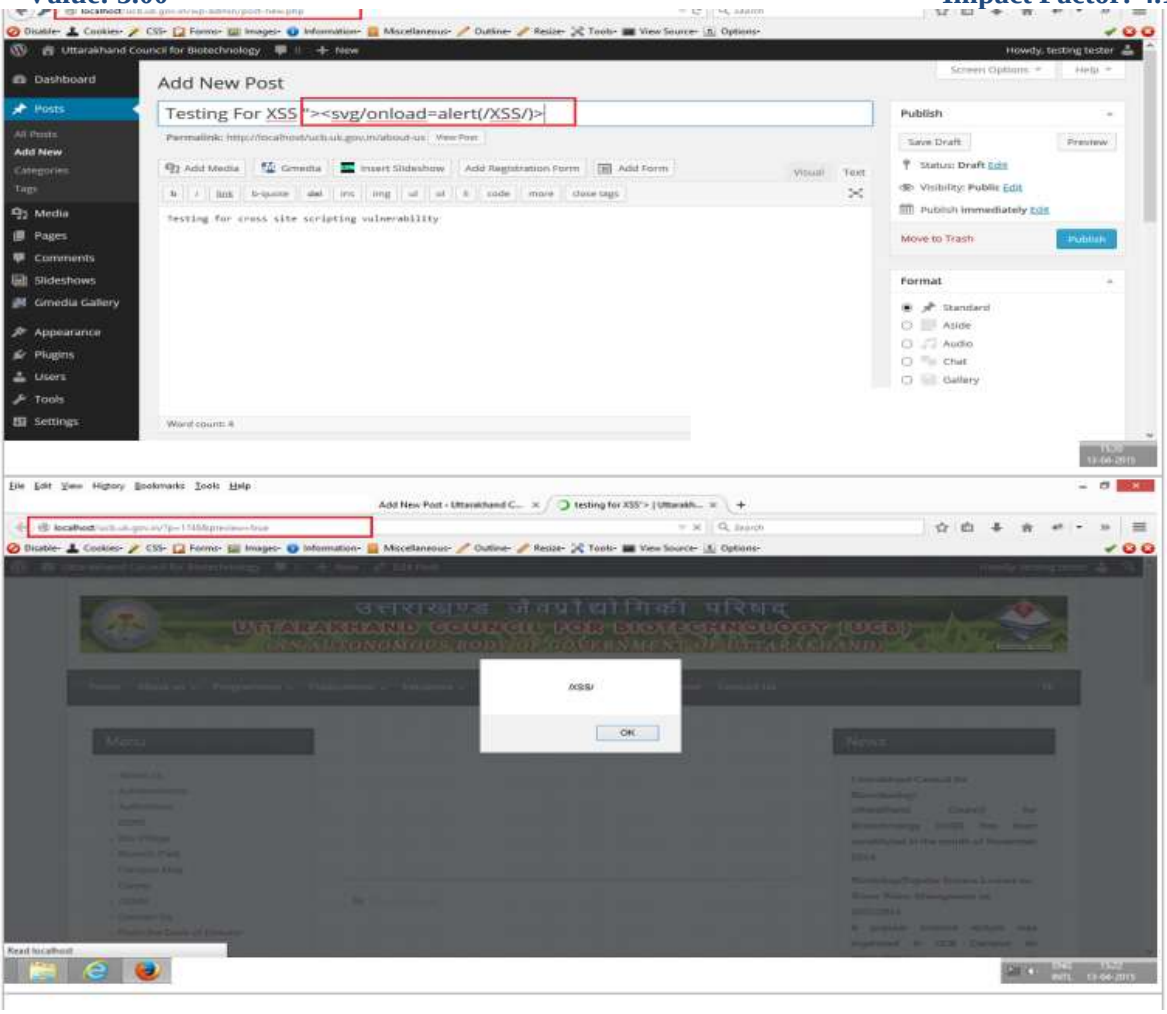

Fig 6. Cross site scripting vulnerability

3. Session Fixation: Session fixation vulnerability occurs when the web application utilizes the preset session identifier instead of generating a new session identifier for every logon event. Application is also vulnerable to session fixation if the session ID issued at the login page is renewed on the successful login. An attacker can force a victim to use preset session identifier and gain access to victim's account once the session token is associated with the account.[7][8]

The example below explains the session fixation attack process:

1. The attacker establishes a legitimate connection with the web server to acquire a session ID or creates a new session with the proposed session ID.
2. Attacker then sends the crafted link with the established session ID to the victim.
3. Victim clicks the link sent by the attacker.
4. The web application renews the session ID instead of generating the new session ID.
5. On a successful login to the web application, the session ID gets activated and attacker then hijacks the victim's session.[7][8]

ISSN: 2277-9655
Impact Factor: 4.116

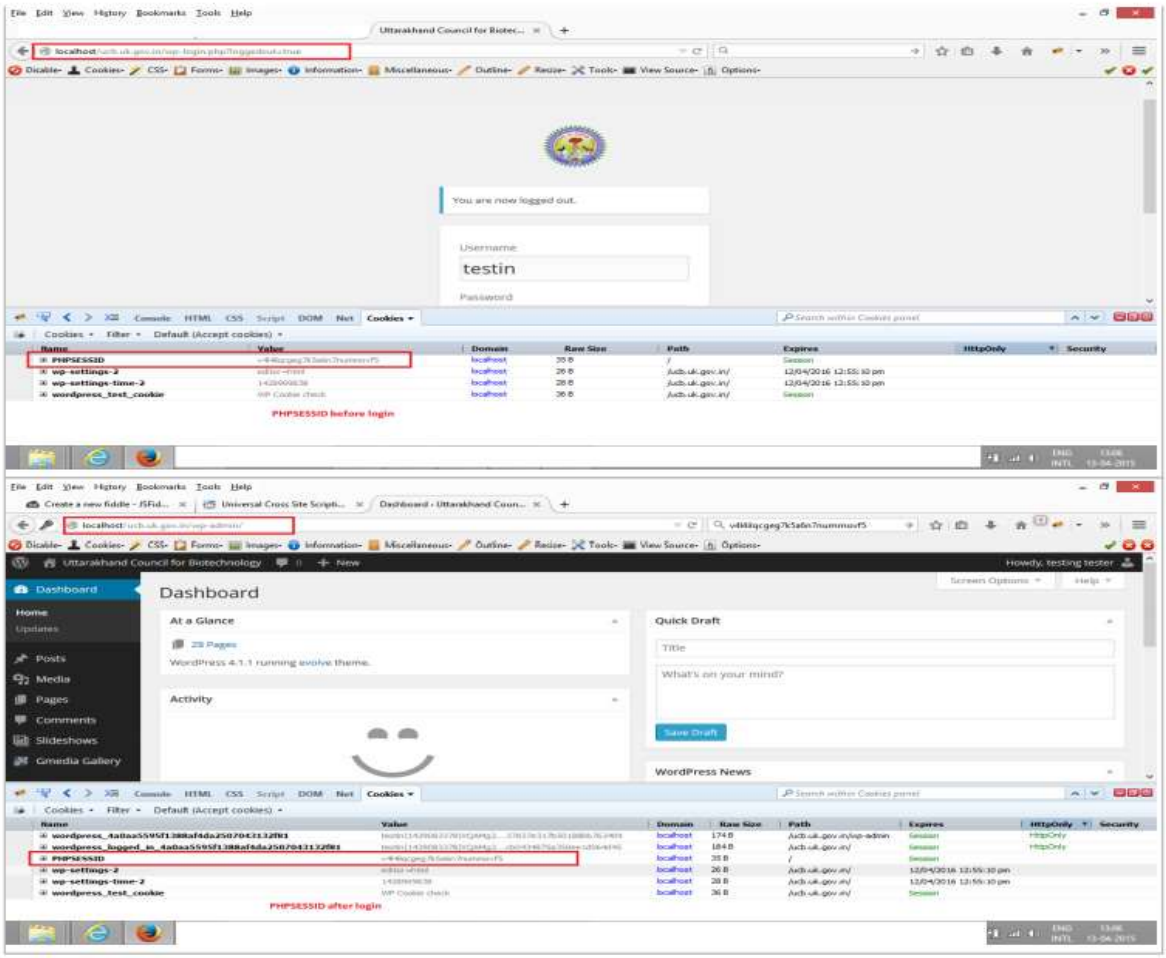

Fig 7. Session fixation vulnerability

4. File Upload Vulnerability: It is observed that the application does not validate the file type being uploaded to the server. An attacker can upload malicious files which may lead to Remote Code Execution and the total defacement of the Web Application.

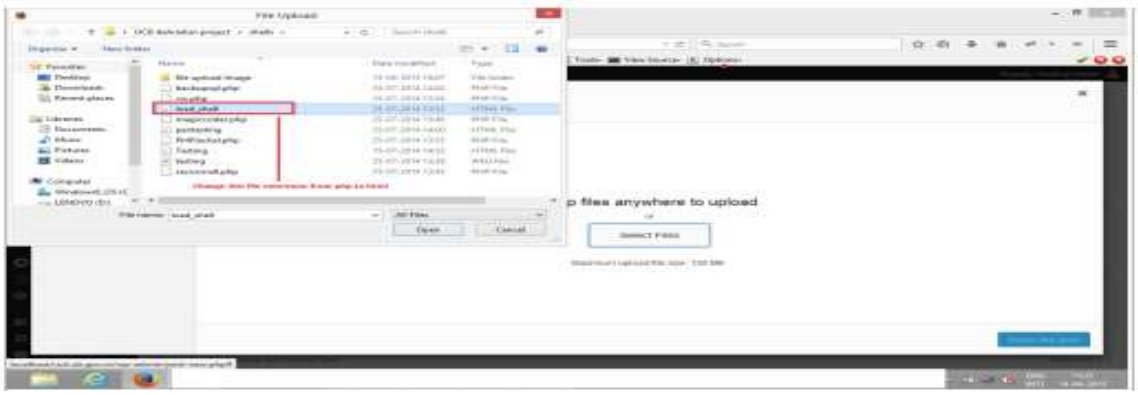

Fig 8. File upload vulnerability part 1

ISSN: 2277-9655
Impact Factor: 4.116

Fig 9. File upload vulnerability part 2

5. HTML Injection: HTML Injection attack, closely relates to XSS, occurs due to inadequate validation of the input data into the web application and at the input fields where the input entered by the user is displayed back in user browser after the submitted data was processed at the server side. For example: 1. User submits a web page with field Full Name with the input text as John Smith. 2. Server receives the submitted data and responds to the user browser Welcome, John Smith. If the input field is not properly sanitized or validated, malicious user could insert a HTML Tags like which should execute on the browser to display an image. HTML Injection however, can exist due to input from GET, POST sometimes even HTTP headers when used in the server response by the application.[9][10][11]

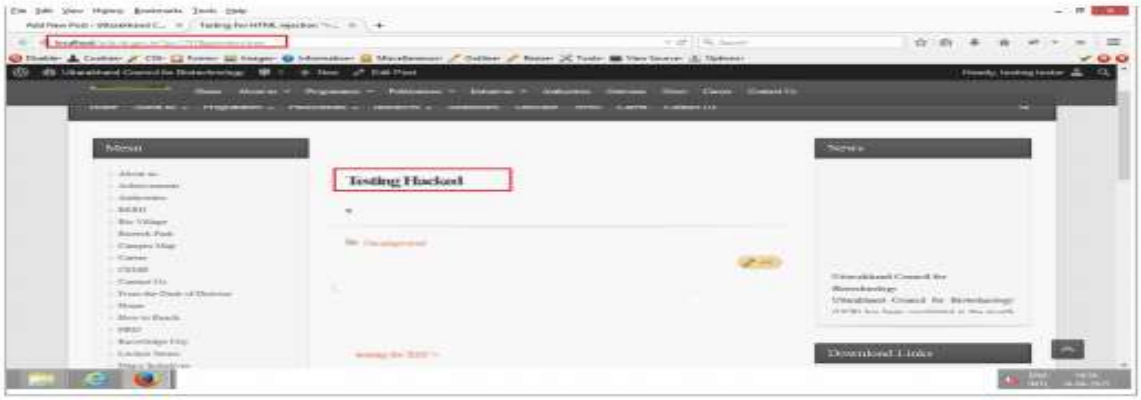

Fig 10. HTML injection vulnerability

6. Iframe Injection : Tag stands for Inline Frame and is used to insert contents from another website or server. It is observed that application is vulnerable to Iframe injection. This injection is performed with html's Iframe tag which is inserted into the vulnerable area of the application. Vulnerability occurs due to inadequate data validation controls within the application.[12][13]

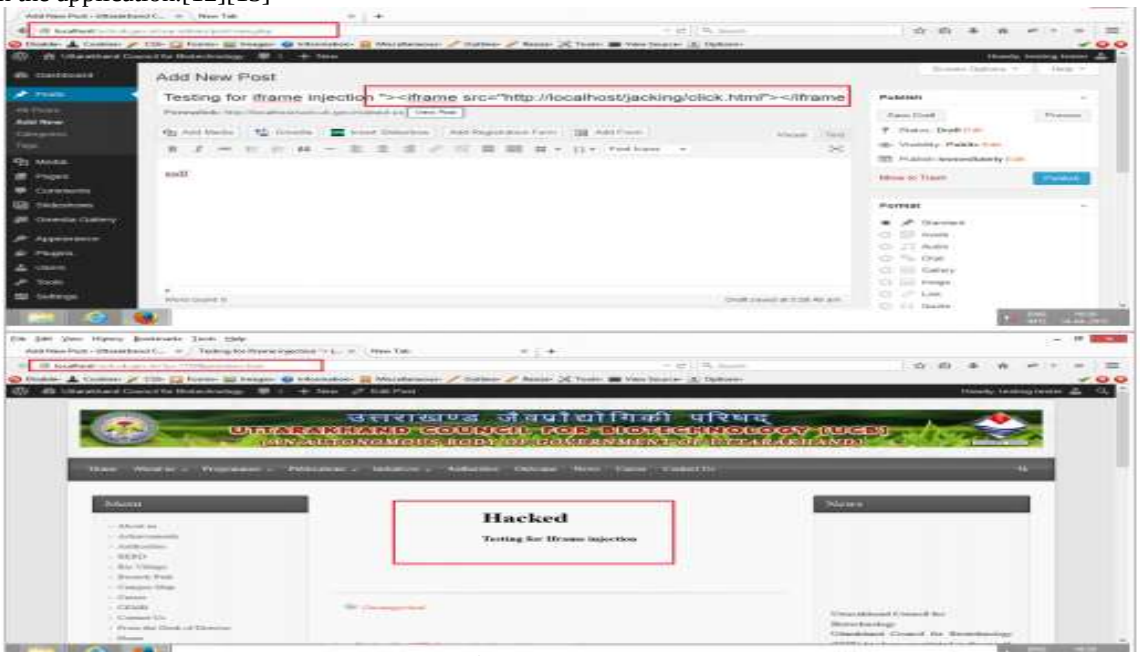

Fig 11. Iframe injection vulnerability

7. Insufficient Transport Layer Protection(SSL/TLS): Insufficient Transport Layer Protection is a security weakness caused by applications not taking any measures to protect network traffic. During the authentication, applications may use SSL/TLS, but they often fail to make use of it elsewhere in the application, thereby leaving data and session ID's exposed. Not using SSL/TLS at all throughout the application, leaves the application open to sniffing of usernames and passwords by an attacker. Sometimes applications are deployed using weak certificate algorithms, expired or invalid certificates, or do not use them correctly. [14][15][16]

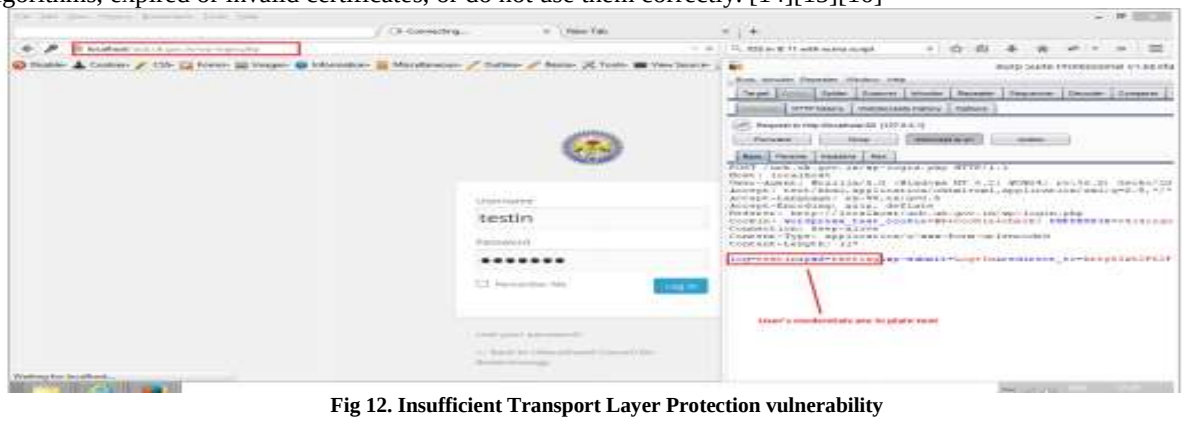

Fig 12. Insufficient Transport Layer Protection vulnerability

8. Improper Cache Control: Improper cache control vulnerability occurs due to improper implementation of cache control directives. It happens when sensitive data get stored in browser cache and it is possible to access authenticated pages of the application after logout, specifically from the history of the browser in offline mode. For example: 1. A user visits to his profile after authentication and views some internal pages that have some sensitive information like credit card number or social security number and then logs out of the application. 2. Later, an

ISSN: 2277-9655
Impact Factor: 4.116

attacker with access to victim's computer can view restricted pages previously accessed by the victim from history or the cache of the system.[17][18]

Fig 12. Improper Cache Control vulnerability

9. Login Brute Force: It is observed that login page does not restrict multiple failed login attempts. A brute force attack is a method to determine an unknown value by using an automated process to try a large number of possible values. For example: While an 8 character alphanumeric password can have 2.8 trillion possible values, many people will select their passwords from a much smaller subset consisting of common words and terms.[20]

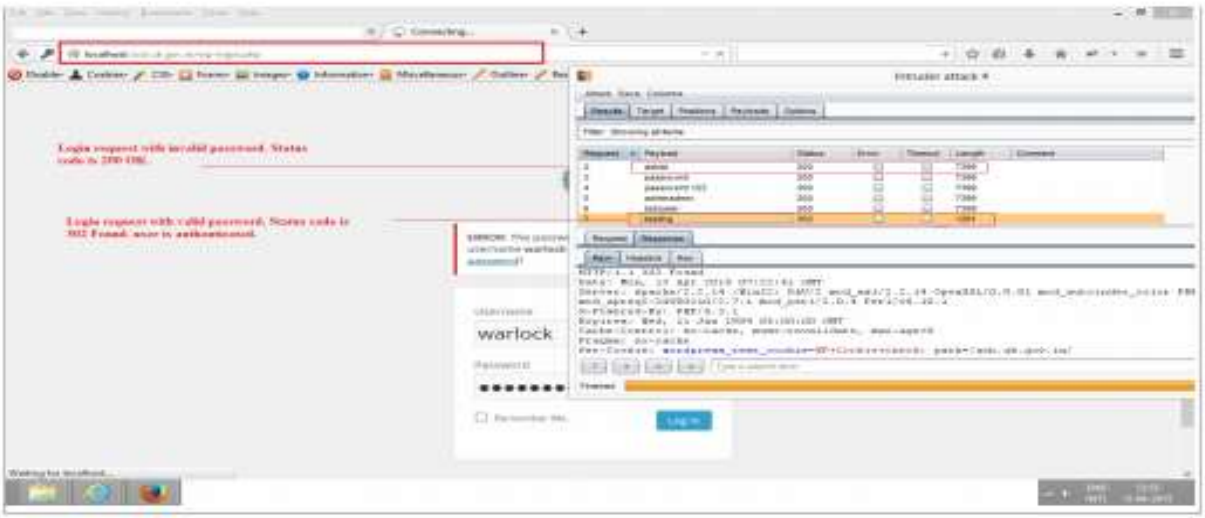

Fig 13. Login Brute Force vulnerability

3.3 Removal of Vulnerabilities:

1. Improper Session Termination, Improper Cache Control & Session Fixation: Vulnerability due to improper cache management solved. File location \wp-include.php Line no 1003-1040. The code below represent the affected area.

$headers = array('Expires' => 'Wed, 11 Jan 1984 05:00:00 GMT', 'Cache-Control' => 'no-cache,no-store,private,pre-check=0,post-check=0, mustrevalidate, max-age=0', 'Pragma' => 'no-cache',);

2. Stored Cross-Site Scripting, HTML Injection & Iframe Injection: Removal of HTML, XSS and iframe injection in post –title field in admin dashboard using input sanitization on title field. File location \wp-includes\post-templates.php. (line no 78-162). The code below represent the affected area.

```
$title = $r['before'] . $title . $r['after']; $title = esc_attr( strip_tags( $title ) ); $title = sanitize_text_field( $title );
```

3. File Upload Vulnerability: Removal of FILE upload vulnerability by removing HTML & .exe files from allowed file types. Location of file /wp-include/functions.php and the code can be found b/w line no's 2164-2277.

```
//'htm|html' => 'text/html',
//'exe' => 'application/x-msdownload',
```

4. Insufficient Transport Layer Protection(SSL/TLS), Login Brute Force and Weak Password Policy: These issues were removed using all in one wordpress security plugin.

5. Cookie Secure and HTTPOnly Attributes Not Present: The cookie secure and Http functions were enabled. File location \WP-confg.php.

```
@ini_set('session.cookie_httponly','On');
@ini_set('session.cookie_secure','On');
@ini_set('session.use_only_cookies','On');
```

6. Auto-Complete Feature Enabled: To disable auto-complete feature we implement two changes: a) First change in functions.php:

```
function kill_autocomplete() {
  wp_register_script('kill-ac',
     get_template_directory_uri() . '/js/autocompleteoff.js',
     array('jquery'),
     '1.0' );
  wp_enqueue_script('kill-ac');
}
add_action('wp_enqueue_scripts', 'kill_autocomplete');
```

b) Second change is in /js/autocompleteoff.js: `jQuery('#user_pass').attr('autocomplete','off');`

7. Http Method Enabled: The http method is disabled from .htaccess file as follows:

```
# Rules to block unneeded HTTP methods
        RewriteCond %{REQUEST_METHOD} ^(TRACE|DELETE|TRACK) [NC]
        RewriteRule ^(.*)$ - [F]
```

3.4. Scanning http://www.ucb.uk.gov.in **using online vulnerability scanning tools:**

1. Google safety diagnostics tool: The scan results shows that the website is safe and not dangerous.

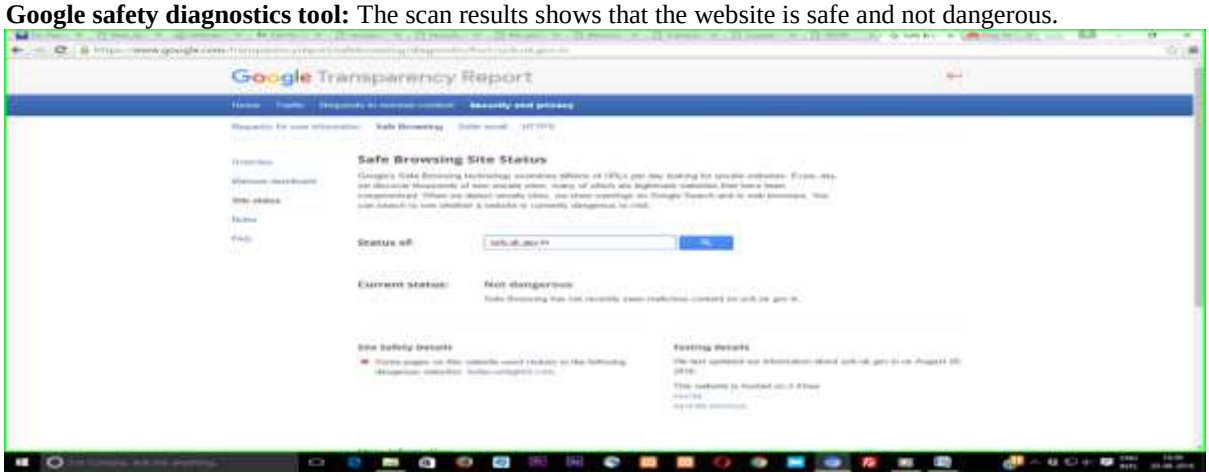

Fig 14. Google transparency report

2. Norton safe web vulnerability scanner tool: The scan results shows that the website is safe.

3. Beyond security online vulnerability scanner tool: The scan results shows that the website is safe with vulnerability score 81.

RESULTS AND DISCUSSION

This work is an attempt to develop a secured wordpress website in which after the basic installation the it was tested according to OWASP, SANS, WASC standards[4]. The flaws in the security were cleared to maintain the website security. The corrected code were applied to the related wordpress file and the updated wordpress was then re-audited to check the vulnerability status. The results of level 2/final website security audit report is as follows:

Table 2: Details of final security audit report

Vulnerability Name	Severity	Status
Improper Session Termination	High	✓
Stored Cross-Site Scripting	High	✓
Session Fixation	High	✓
File Upload Vulnerability	High	✓
HTML Injection	High	✓
Iframe Injection	High	✓
Insufficient Transport Layer Protection(SSL/TLS)	Medium	▲
Improper Cache Control	Medium	✓
Login Brute Force	Medium	✓
Cookie Secure Attribute Not Present	Low	▲
Auto-Complete Feature Enabled	Low	▲
HTTPOnly Attribute Not Present	Low	▲
Weak Password Policy	Low	✓
Http Method Enabled	Low	▲

✓ = Fixed ▲ = Business Exception

The website www.ucb.uk.gov.in is safe and secure according to National Informatics Centre security audit vendor report and is safe for hosting at NIC platform. For the sake of counter verification the website was also audited from online vulnerability diagnostic tools and some of the reports also confirm the safety of the website. After this work the wordpress recently had been updated to the wordpress version 4.5.2 along with hardened .htaccess , php.ini and other security measures to compete with the latest vulnerability issues. And the current status is also safe and secure.

REFERENCES

1. **Rushabhkumar H. Baldaniya and Prof. H.J.Baldaniya,** 2014 *"Web Development Using Content Management System",* International Journal of Emerging Research in Management &Technology, April 2014, ISSN: 2278-9359, Volume-3, Issue-4.
2. **Vimal Ghorecha and Chirag Bhatt,** 2013 "A guide for Selecting Content Management System for Web Application Development", International Journal of Advance Research in Computer Science and Management Studies Research Paper, August 2013, ISSN: 2321-7782, Vol. 1, Issue 3.
3. https://en.wikipedia.org/wiki/Wordpress , 20 August 2016.
4. https://en.wikipedia.org/wiki/Web_application_security , 20 August 2016.
5. https://www.owasp.org/index.php/Testing_for_Logout_and_Browser_Cache_Management_(OWASP-AT-007) , 01 may 2015.
6. https://www.owasp.org/index.php/Testing_for_Stored_Cross_site_scripting_(OTGINPVAL-002) , 01 may 2015.
7. https://www.owasp.org/index.php/Session_fixation, 01 may 2015.
8. http://cwe.mitre.org/data/definitions/384.html, 01 may 2015.
9. https://www.owasp.org/index.php/HTML_Injection, 01 may 2015.
10. https://www.owasp.org/index.php/HTML_Entity_Encoding, 01 may 2015.
11. http://php.net/manual/en/function.html-entity-decode.php, 01 may 2015.
12. https://www.owasp.org/index.php/Cross_Frame_Scripting, 01 may 2015.
13. http://resources.infosecinstitute.com/iframe-security-risk/, 01 may 2015.

14. https://www.owasp.org/index.php/Top_10_2010-A9-Insufficient_Transport_Layer_Protection, 01 may 2015.
15. http://projects.webappsec.org/w/page/13246945/Insufficient%20Transport%20Layer%20Protection, 01 may 2015.
16. http://www.troyhunt.com/2011/11/owasp-top-10-for-net-developers-part-9.html, 01 may 2015.
17. https://www.owasp.org/index.php/Testing_for_Browser_cache_weakness_%28OTGAUTHN-006%29, 01 may 2015.
18. http://condor.depaul.edu/dmumaugh/readings/handouts/SE435/HTTP/node24.html, 01 may 2015.
https://www.owasp.org/index.php/Blocking_Brute_Force_Attacks