ControlCase
Compliance as a Service

ABC PVT. LTD.

Web Application Penetration Test Report

Version 1.0

December 20, 2017

Statement of Confidentiality

This Confidential Information is being provided to **ABC PVT. LTD.** as a deliverable of this consulting engagement. The sole purpose of this document is to provide you with the results of, and recommendations derived from this consulting engagement. Each recipient agrees that, prior to reading this document, it shall not distribute or use the information contained herein and any other information regarding **ControlCase** for any purpose other than those stated.

Table of Contents

1 Executive Summary

1.1 Introduction

ABC PVT. LTD. engaged **ControlCase** to conduct a Web Application Penetration Test of their **WEB_APP** web application. The purpose of the engagement was to identify and prioritize the potential areas of security vulnerability.

The engagement began on **November 02, 2017** and included multiple phases of testing, analysis and documentation. All testing was conducted from ControlCase Lab.

This document summarizes the analysis, Observations and recommendations for the assessment carried out by **ControlCase**.

1.2 Goals & Objectives

The purpose of this assessment was to identify technical as well as logical vulnerabilities in the application and provide recommendations for risk mitigation that may arise on exploiting these vulnerabilities. The idea behind this testing was to discover whether an attacker can leverage flaws in the application to compromise the confidentiality, integrity and availability of the information. **ControlCase** worked with **ABC PVT. LTD.** to achieve the following key objectives:

To determine whether adequate information security controls have been built into the application.

Perform supplemental research and development activities to support analysis.

Prioritize vulnerabilities based upon the ease of exploit, level of effort to remedy, and severity of impact if exploited.

Assess current practice against industry best practices.

Deliver report which includes **ControlCase**'s Observations, analysis, and recommendations.

Transfer knowledge.

1.3 Approach & Methodology

ControlCase Application Security consultants follow the **OWASP** (**O**pen **W**eb **A**pplication **S**ecurity **P**roject) an established guidelines in application security methodology. In the course of the assessment, ControlCase consultants use a variety of commercial, open-source tools as well as homegrown scripts & tools.

ControlCase has defined following approaches while doing application security assessment.

Black box testing – This is a technique to attempt to penetrate application where the source code of the application is not available to the tester. ControlCase team will attempt to elicit exception conditions and anomalous behavior from the Web Application by manipulating the identified inputs - using special characters, SQL keywords, maliciously crafted requests, and so forth. Any unexpected reaction from the Web Application is noted and investigated. This may take the form of scripting error messages, server errors or half-loaded pages. The goal of this method is to simulate an attack by an external hacker.

Grey box testing – This approach is similar to black box testing; however, the attack team is given the same privileges as an 'admin/normal' user of the application and the goal is to simulate an attack by a malicious insider. The attack team tries to escalate the privileges of a normal user to administrator user.

Types of tests performed	**Checked**
1. Application Security Assessment Test	
• Automated scanning of possible web application vulnerability	✓
• Manual exploit on discovered vulnerability	✓
• Compliance Specific checks (e.g. PCI DSS)	✓
2. OWASP Top 10 2017	
• Injection Flaws	✓
• Broken Authentication	✓
• Sensitive Data Exposure	✓
• XML External Entities (XXE)	✓
• Broken Access Control	✓
• Security Misconfiguration	✓
• Cross Site Scripting (XSS)	✓
• Insecure Deserialization	✓
• Using Components with Know Vulnerabilities	✓
• Insufficient Logging & Monitoring	✓

1.4 Project Team – Contact Information, Credentials/Qualification

The engagement involved contributions from the following team members:

ControlCase Team	ABC Team

1.5 Penetration Timeline

The following table outlines key milestones during the penetration test:

Penetration Timeline

Date	Milestone
November 02, 2017	Start of Project
December 20, 2017	Final Deliverable

1.6 Target Description

The penetration testing for **WEB_APP** application was carried out on one application URL. The approach conducted was a black box testing followed by grey box testing. The application was hosted externally in the **ABC's** environment.

Technical Details of the Target:

Total one application was subjected to assessment. The target application's URL is given below:

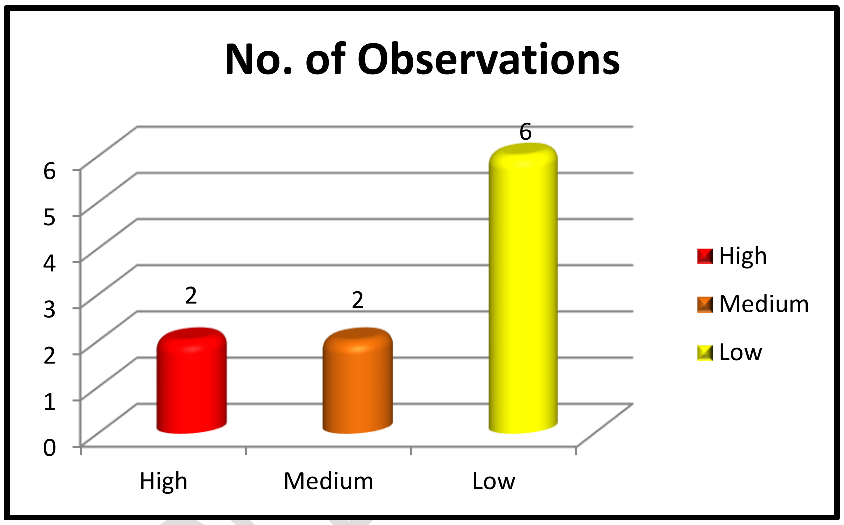
Sr. No.	Application URL(s)	IP Address(es)
1.	https://www.abc.com/WEB_APP	12.XX.XX.23

1.7 Summary of Observations

The tested web application is having two (02) high-risk, two (02) medium-risk and six (06) low-risk vulnerabilities that an attacker can target or exploit. It is important to periodically check, review and modify the application logic if any kind of change is being applied to the application.

The graph below gives the status of severity of the vulnerabilities found during the Application Security Assessment.

Risk Severity Level	No. of Observations
High	2
Medium	2
Low	6
Total	11



<u>Given below is the summary of the Observations:</u>

Sr. No.	Observations	Risk Level
1.	Encryption not enforced	High
2.	SQL Injection	High
3.	Frame Injection	Medium
4.	Wordpress username enumeration	Medium
5.	Flash parameter AllowScriptAccess was set to always	Low
6.	Insecure Implementation of Session Cookie	Low
7.	Wordpress File Disclosure	Low
8.	Possible Server Path Disclosure Pattern Found	Low
9.	Auto-Complete Attribute Not Set to Off	Low
10.	HTTP TRACE Method Enabled	Low

1.8 Statement on Compliance

ControlCase has determined that **ABC PVT. LTD’s WEB_APP** application is **Non-Compliant** with **ControlCase** validation requirement as mentioned in section 1.3.

2 Detailed Observations

2.1 Overview

The following format shows a typical vulnerability representation and provides in detail information of vulnerabilities discovered during Application Vulnerability Test.

2.2 Vulnerability Table

1. Vulnerability Title	
Risk Level	🟥
OWASP Category	
Abstract	
Ease of Exploitation	
Impact	
Recommendations	
Substantiated Assessment	
Affected URL	
Note	
Reference	
CWE	

- **Vulnerability Title – A short title that describes the vulnerability.**

  The title bar for each vulnerability table is color coded for a quick identification of the risk level. Title bar color codes are as follows:

Risk Level	Description
🟥	**High risk** vulnerability can be exploited by an attacker to gain full administrative access to the application or its underlying operating system.
🟧	**Medium risk** vulnerability reveals information about the application and its underlying infrastructure that can be used by an attacker in conjunction with another vulnerability to gain administrative control of the application or its underlying operating system.
🟨	**Low risk** vulnerability can result in enumeration of vital information held by or about the Application or its underlying operating system.

- **OWASP Category –** Refers to OWASP top 10-2014 vulnerability category.
- **Abstract –** Describes the flaw or bugs that cause the vulnerability.

- **Ease of Exploitation –** Provides a metric for the skill level required to exploit the vulnerability. The categories are:

Metric	Skill-level
Easy	Casual user
Medium	Computer-savvy individual
Hard	Determined hacker

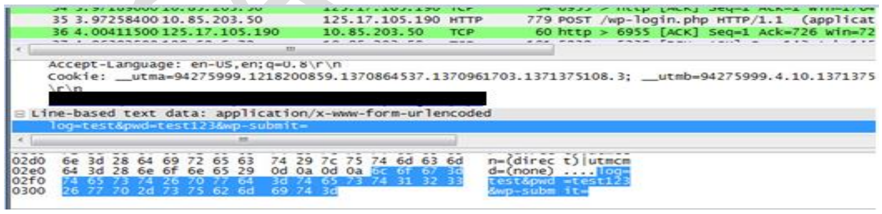
- **Impact –** Describes the possible business impact if this vulnerability is successfully exploited.
- **Recommendation –** Provides solutions or workarounds to mitigate the risk arising from this vulnerability.
- **Substantial Assessment –** The evidence of the vulnerability being present, wherever possible, is provided in the form of screenshots.
- **Affected URL –** Provides URLs and respective parameters which are affected with that specific vulnerability
- **Note –** A brief description of how the vulnerability can be exploited by internal/external attacker or limitations for exploitation which may result in minimizing the risk of the reported vulnerability.
- **Reference –** It provides reference to outside resource such as OWASP, SANS etc.
- **CWE –** Provides Common Weakness Enumeration ID

2.3 Vulnerability Discovery Phase

This phase has been completed successfully. Assessor observed two (02) high-risk, two (02) medium-risk and six (06) low-risk vulnerabilities during the application penetration testing.

2.3.1 Encryption Not Enforced

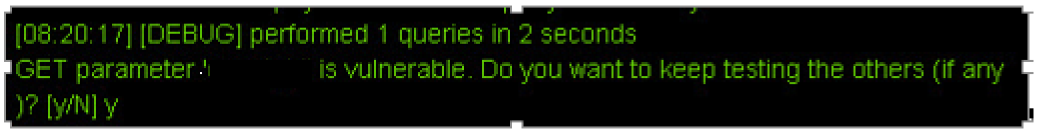
Risk Level	🟥 **High**
OWASP Category	A3 – Sensitive Data Exposure
Abstract	Assessor observed that the server also uses HTTP protocol for sensitive data transfers. It was also noted that HTTPS is enabled on the server however is not enforced to be used.
Ease of Exploitation	Hard
Impact	HTTP service does not encrypt its data and control connections. Sensitive data is transmitted in clear text and may be intercepted by a network sniffer, or via a man-in-the-middle attack. **Technical Threats**: Sniffing, Man in the Middle attacks **Logical Threats:** Data loss, sensitive information (including credentials) loss
Recommendations	It is recommended that – • HTTP protocol should be completely disabled if possible and only HTTPS should be used • Any traffic coming for HTTP service should be redirected towards HTTPS service

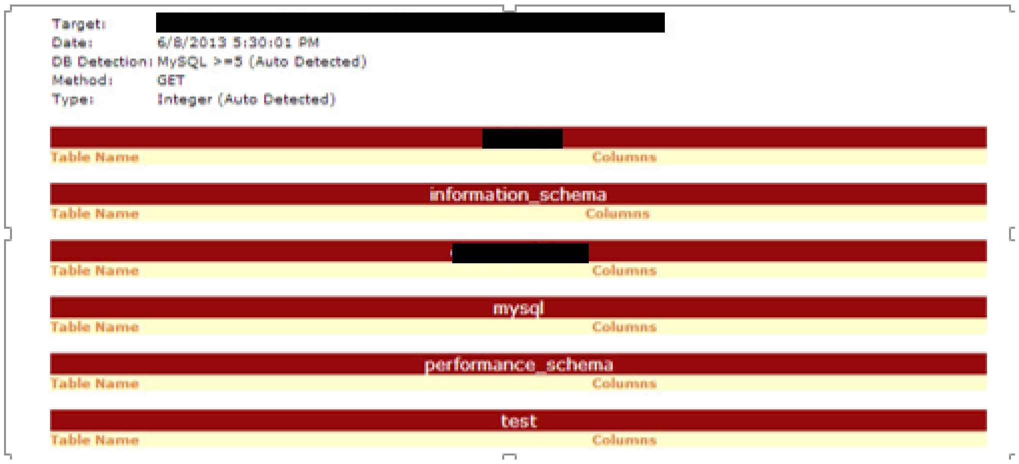
Substantiated Assessment	
	`35 3.97258400 10.85.203.50 125.17.105.190 HTTP 779 POST /wp-login.php HTTP/1.1 (applicat`
	`36 4.00411500 125.17.105.190 10.85.203.50 TCP 60 http > 6955 [ACK] Seq=1 Ack=726 Win=72`
	`Accept-Language: en-US,en;q=0.8\r\n`
	`Cookie: __utma=94275999.1218200859.1370864537.1370961703.1371375108.3; __utmb=94275999.4.10.1371375`
	`\r\n`
	[redacted]
	`Line-based text data: application/x-www-form-urlencoded`
	`log=test&pwd=test123&wp-submit=`
	`02d0 6e 3d 28 64 69 72 65 63 74 29 7c 75 74 6d 63 6d n=(direc t)|utmcm`
	`02e0 64 3d 28 6e 6f 6e 65 29 0d 0a 0d 0a 6c 6f 67 3d d=(none) ....log=`
	`02f0 74 65 73 74 26 70 77 64 3d 74 65 73 74 31 32 33 test&pwd =test123`
	`0300 26 77 70 2d 73 75 62 6d 69 74 3d &wp-subm it=`

Affected URL	https://www.abc.com/WEB_APP/signin/
Note	Attacker could capture network traffic to the application to get sensitive information like user credentials exchanged between user and application server.
Reference	https://www.owasp.org/index.php/Top_10-2017_A3-Sensitive_Data_Exposure
CWE	311

2.3.2 SQL Injection

Risk Level	🟥 **High**
OWASP Category	A1 – Injection
Abstract	It was observed that the application is not validating/sanitizing the parameters sent in the request and thus is vulnerable to SQL injection. SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.
Ease of Exploitation	Hard
Impact	SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, destroy the data or make it otherwise unavailable and escalate his privileges to become administrator of the database server.
Recommendations	**It is recommended that measures provided below shall be applied/considered to mitigate this vulnerability -** **Input validation:** Use a standard input validation mechanism to validate all input data for length, type, syntax, and business rules. Reject invalid input rather than attempting to sanitize potentially hostile data. Use strongly typed parameterized query APIs with placeholder substitution markers, even when calling stored procedures Enforce least privilege when connecting to databases and other backend systems Avoid detailed error messages that are may disclose sensitive information Show care when using stored procedures since they are generally safe from SQL Injection. However, be careful as they can be injectable (such as via the use of exec() or concatenating arguments within the stored procedure) Do not use dynamic query interfaces (such as *mysql_query()* or similar) Do not use simple escaping functions, such as PHP's *addslashes()* or character replacement functions like *str_replace("'", "")* when using simple escape mechanisms. Watch out for canonicalization errors. Inputs must be decoded and canonicalized to the application's current internal representation before being validated.

Substantiated Assessment	Testing for vulnerable parameter:
	
	`[08:20:17] [DEBUG] performed 1 queries in 2 seconds`
	`GET parameter '[redacted] is vulnerable. Do you want to keep testing the others (if any)? [y/N] y`

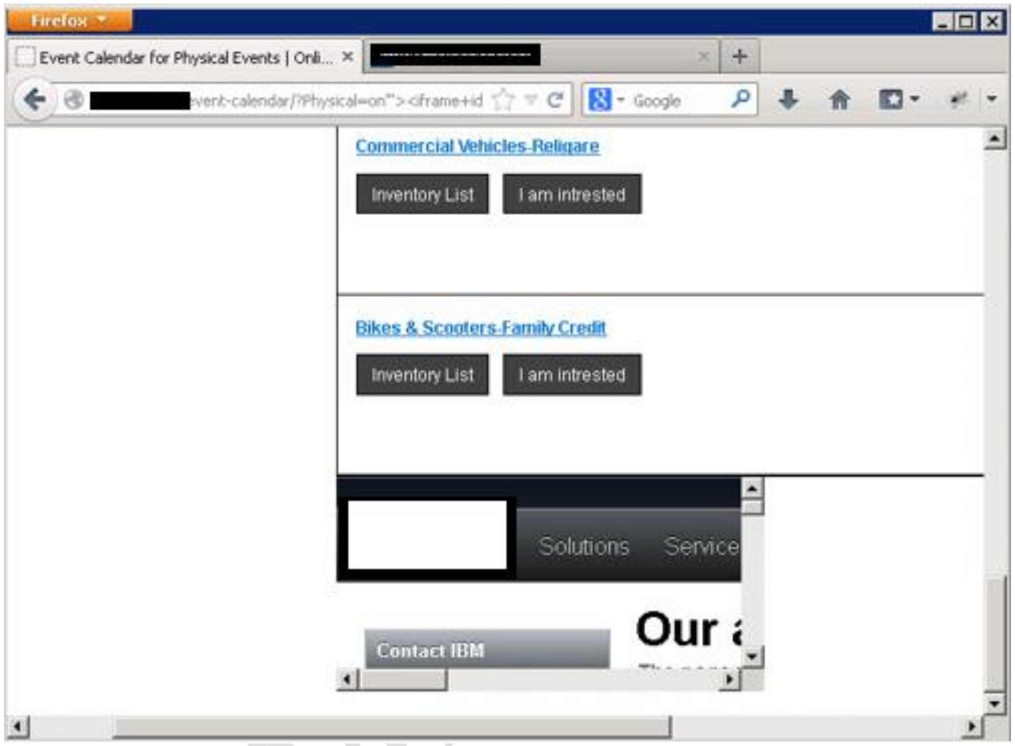
	Databases identified:  Target: [redacted] Date: 6/8/2013 5:30:01 PM DB Detection: MySQL >=5 (Auto Detected) Method: GET Type: Integer (Auto Detected) [redacted] Table Name Columns information_schema Table Name Columns [redacted] Table Name Columns mysql Table Name Columns performance_schema Table Name Columns test Table Name Columns
Affected URLs	https://www.abc.com/WEB_APP/event-details Parameter: event_id https://www.abc.com/WEB_APP/event-results-inventory/ Parameter: orderby https://www.abc.com/WEB_APP/media/photos/photos-gallery/ Parameter: gid https://www.abc.com/WEB_APP/photos-gallery/ Parameter: gid
Note	Attacker can send malicious queries as parameter values for vulnerable parameters to fetch the databases and their content.
Reference	https://www.owasp.org/index.php/Top_10-2017_A1-Injection
CWE	

2.3.3 Frame Injection

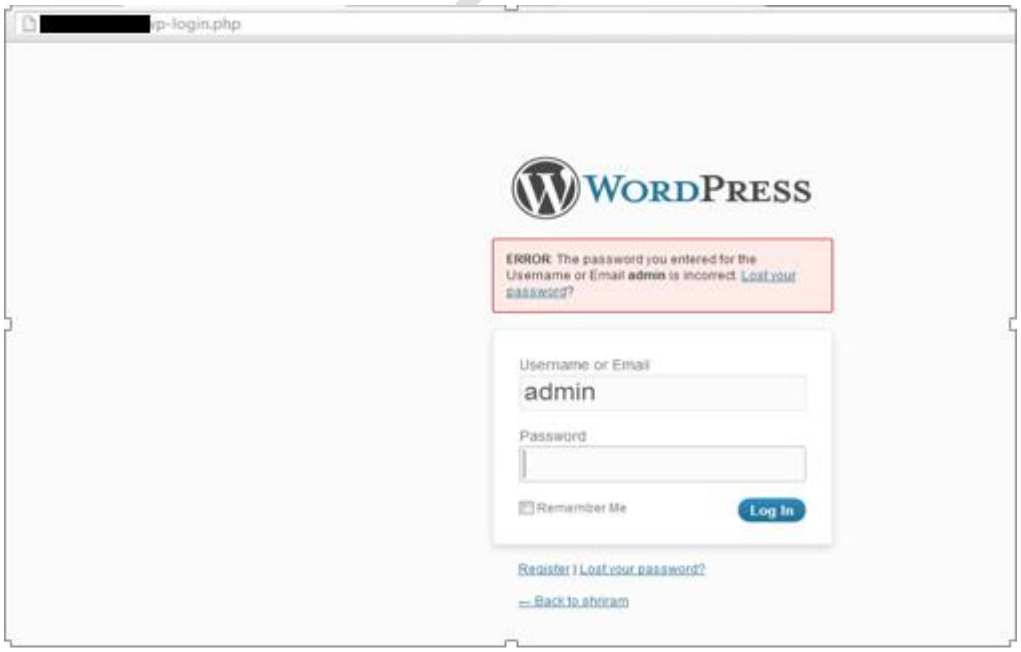
Risk Level	**Medium**
OWASP Category	A7 – Cross-Site Scripting (XSS)
Abstract	It is possible for an attacker to inject a frame or an IFrame tag with malicious content which resembles the attacked site.
Ease of Exploitation	Hard
Impact	An incautious user may browse the injected frame and not realize that he is leaving the original site and surfing to a malicious site. The attacker may then lure the user to login again, thus acquiring his login credentials.
Recommendations	**To mitigate Frame Injection follow the below given instructions:** The application must perform validation of all headers, cookies, query strings, form fields, and hidden fields (i.e., all parameters) against a rigorous specification of what should be allowed. Also filter the content of image files being uploaded to the server. Any meta-characters should be filtered for, in all input accepting fields, both at client side as well as sever side. Server side validation is mandatory. The validation should not attempt to identify active content and remove, filter, or sanitize it. Encoding user supplied output can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. The application must be configured to filter meta-characters and unexpected characters such as:

Character	Encoding
<	&lt; or &#60;
>	&gt; or &#62;
&	&amp; or &#38;
"	&quot; or &#34;
'	&apos; or &#39;
(	&#40;

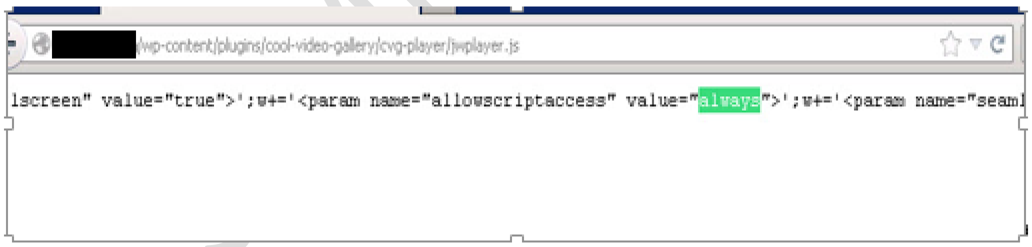
Character	Encoding
)	&#41;
#	&#35;
%	&#37;
;	&#59;
+	&#43;
-	&#45;

	Note: The recommendations must be applied on all the parameters irrespective of they are vulnerable or not. Moreover, the input validation must be handled through centralized way and not separately for each parameter.
Substantiated Assessment	 Firefox Event Calendar for Physical Events | Onli... [redacted] [redacted]event-calendar/?Physical=on"><iframe+id Google Commercial Vehicles-Religare Inventory List I am intrested Bikes & Scooters-Family Credit Inventory List I am intrested Solutions Service Contact IBM Our
Affected URL	https://www.abc.com/WEB_APP/event-results-inventory/ Parameters: calSearch https://www.abc.com/WEB_APP/event-calendar/ Parameters: Physical, Online, Negotiated, limit, eventdate
Note	Attacker can send malicious queries as parameter values for vulnerable parameters to fetch the databases and their content.
Reference	https://www.owasp.org/index.php/Top_10-2017_A7-Cross-Site_Scripting_(XSS)
CWE	

2.3.4 Wordpress username enumeration

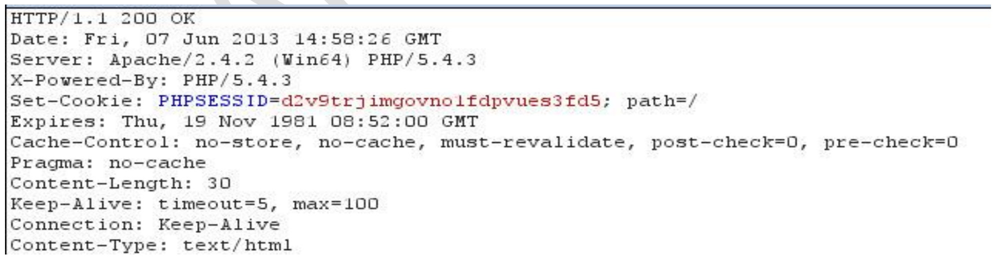
Risk Level	**Low**
OWASP Category	A3 – Sensitive Data Exposure
Abstract	It was observed that the user name enumeration was possible because of the generic error messages displayed by the application.
Ease of Exploitation	Medium
Impact	An attacker can identify users of the systems and may perform a brute-force attack against identified users to compromise their passwords.
Recommendations	Application should not specify which of the provided credentials are wrong. It is recommend that the error message should be like “The username and/or password entered is incorrect.”
Substantiated Assessment	 [redacted]wp-login.php **ERROR**: The password you entered for the Username or Email **admin** is incorrect. Lost your password? Username or Email: admin Password: Remember Me Log In Register | Lost your password? ← Back to ahriram
Affected URL	https://www.abc.com/WEB_APP/wp-login.php
Note	An attacker can find out users of the system and may perform a brute-force attack them in order to compromise the user session.
Reference	https://www.owasp.org/index.php/Top_10-2017_A3-Sensitive_Data_Exposure
CWE	

2.3.5 Flash parameter AllowScriptAccess was set to always

Risk Level	**Low**
OWASP Category	A9 - Using Components with Known Vulnerabilities
Abstract	It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user.
Ease of Exploitation	Hard
Impact	The AllowScriptAccess parameter determines whether the loaded SWF (or any SWF it subsequently loads) will be permitted to access the web page in which the SWF is embedded. If the parameter is set to 'always' then the SWF loaded from any domain could inject a script into the hosting web page.
Recommendations	Set the AllowScriptAccess parameter to 'sameDomain' which tells the Flash Player that only SWF files loaded from the same domain as the parent SWF will have script access to the hosting web pages
Substantiated Assessment	 [redacted]/wp-content/plugins/cool-video-gallery/cvg-player/jwplayer.js `lscreen" value="true">';w+='<param name="allowscriptaccess" value="always">';w+='<param name="seaml`
Affected URLs	https://www.abc.com/WEB_APP/wp-content/plugins/cool-video-gallery/cvg-player/jwplayer.js https://www.abc.com/WEB_APP/wp-content/themes/twentyeleven-child/js/jquery.prettyPhoto.js https://www.abc.com/WEB_APP/wp-content/themes/twentyeleven-child/js/js_common4.js https://www.abc.com/WEB_APP/wp-content/themes/twentyeleven-child/js/min_js/min1.js
Note	Attacker can exploit this vulnerability to load SWF from other domain and may steal user sessions, cookies etc.
Reference	https://www.owasp.org/index.php/Top_10-2017_A9-Using_Components_with_Known_Vulnerabilities
CWE	

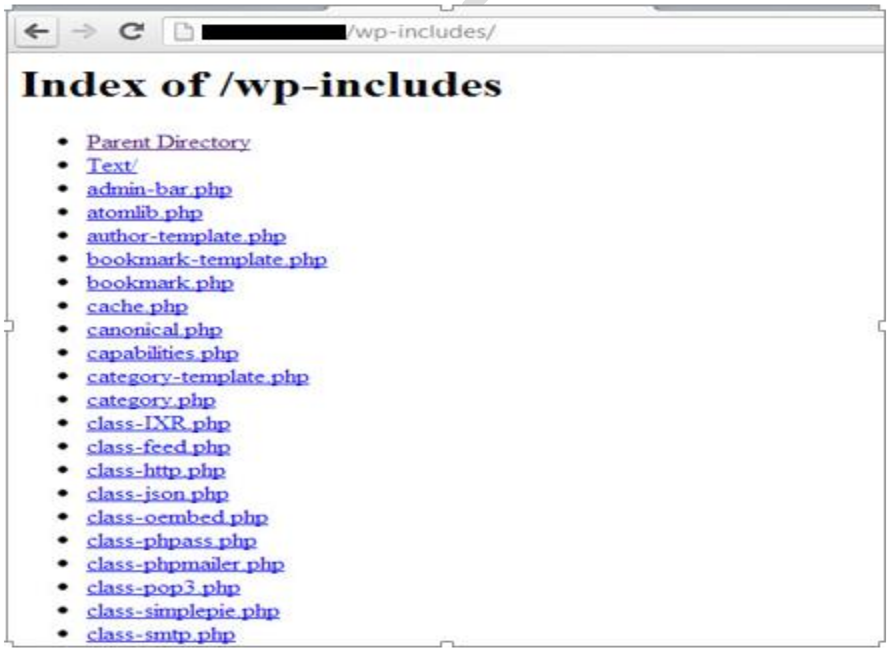
2.3.6 Insecure Implementation of Session Cookie

Risk Level	**Low**
OWASP Category	A2 - Broken Authentication
Abstract	It was observed that session attributes like as “HTTPOnly”, “Secure” & “Domain” name are not set with Session IDs.
Ease of Exploitation	Hard
Impact	Without the “HTTPonly” attribute the session cookie can be used by other protocols/services which are running on the target system. Without “Secure” attribute the application can transfer the session cookie over unencrypted channel Without “Domain” attribute the cookie can be used by other domains and facilitate cross-site request forgery. Also By combining this vulnerability with others an attacker can perform attacks on session ID such as session hijacking.
Recommendations	It is recommended that the application should be configured to set the session attributes such as HTTPOnly, Secure and Domain with session ID.

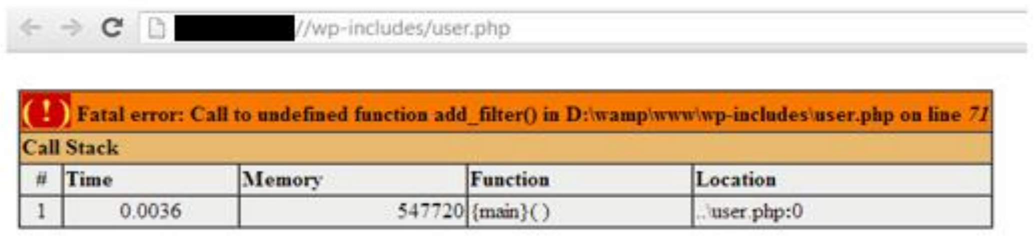
Substantiated Assessment	 `HTTP/1.1 200 OK`
	`Date: Fri, 07 Jun 2013 14:58:26 GMT`
	`Server: Apache/2.4.2 (Win64) PHP/5.4.3`
	`X-Powered-By: PHP/5.4.3`
	`Set-Cookie: PHPSESSID=d2v9trjimgovno1fdpvues3fd5; path=/`
	`Expires: Thu, 19 Nov 1981 08:52:00 GMT`
	`Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0`
	`Pragma: no-cache`
	`Content-Length: 30`
	`Keep-Alive: timeout=5, max=100`
	`Connection: Keep-Alive`
	`Content-Type: text/html`

Affected URL	https://www.abc.com/WEB_APP/wp-content/plugins/wp-tell-a-friend-popup-form/tell-a-friend-save.php
Note	Without “HTTPOnly”, “Secure” and “Domain” attributes, attacker can capture the cookies and may perform malicious actions on behalf of legitimate user.
Reference	https://www.owasp.org/index.php/Top_10-2017_A2-Broken_Authentication
CWE	

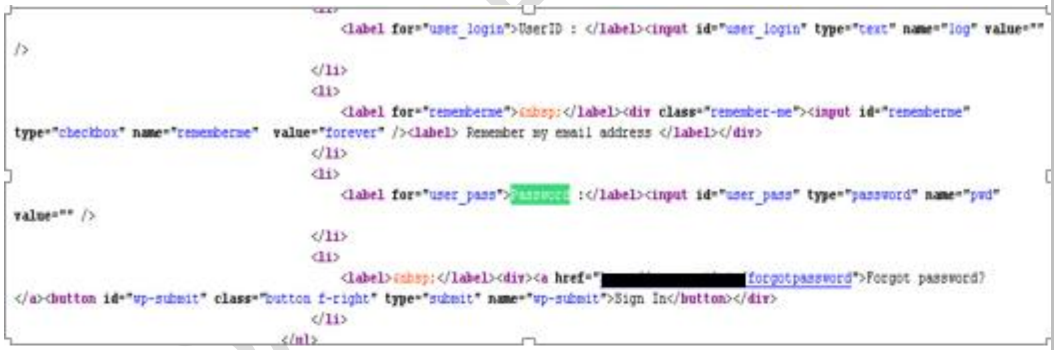
2.3.7 Wordpress Directory Enumeration

Risk Level	**Low**
OWASP Category	A5 – Broken Access Control
Abstract	It was observed that restricted/wordpress system files present on the server which can be accessed by any attacker.
Ease of Exploitation	Hard
Impact	Attacker can get the knowledge of structure of web-Application and my plan further attacks accordingly.
Recommendations	It is recommended that the application should block the request for any files which are not allowed to be viewed to the user.
Substantiated Assessment	 [redacted]/wp-includes/ **Index of /wp-includes** • Parent Directory • Text/ • admin-bar.php • atomlib.php • author-template.php • bookmark-template.php • bookmark.php • cache.php • canonical.php • capabilities.php • category-template.php • category.php • class-IXR.php • class-feed.php • class-http.php • class-json.php • class-oembed.php • class-phpass.php • class-phpmailer.php • class-pop3.php • class-simplepie.php • class-smtp.php
Affected URL	https://www.abc.com/WEB_APP/wp-includes/
Note	Attacker can get this information and could plan further attacks to obtain sensitive information stored on the server.
Reference	https://www.owasp.org/index.php/Top_10-2017_A5-Broken_Access_Control
CWE	

2.3.8 Possible Server Path Disclosure Pattern Found

Risk Level	**Low**
OWASP Category	A6 – Security Misconfiguration
Abstract	It is possible to retrieve the absolute path of the web server installation, which might help an attacker to develop further attacks and to gain information about the file system structure of the web application.
Ease of Exploitation	Hard
Impact	An attacker may be able to exploit this information to access sensitive information on the directory structure of the server machine which could be used for further attacks against the site.
Recommendations	It is recommended that – 1. Download the relevant security patch depending on the issue existing on your web server or web application. 2. Restrict access to the relevent pages of the application by providing custom error messages.
Substantiated Assessment	 [redacted]//wp-includes/user.php (!) Fatal error: Call to undefined function add_filter() in D:\wamp\www\wp-includes\user.php on line *71* Call Stack # Time Memory Function Location 1 0.0036 547720 {main}( ) ..\user.php:0 t
Affected URL	https://www.abc.com/WEB_APP/wp-includes/
Note	Attacker can get this information and could plan further attacks to obtain sensitive information stored on the server.
Reference	https://www.owasp.org/index.php/Top_10-2017_A6-Security_Misconfiguration
CWE	

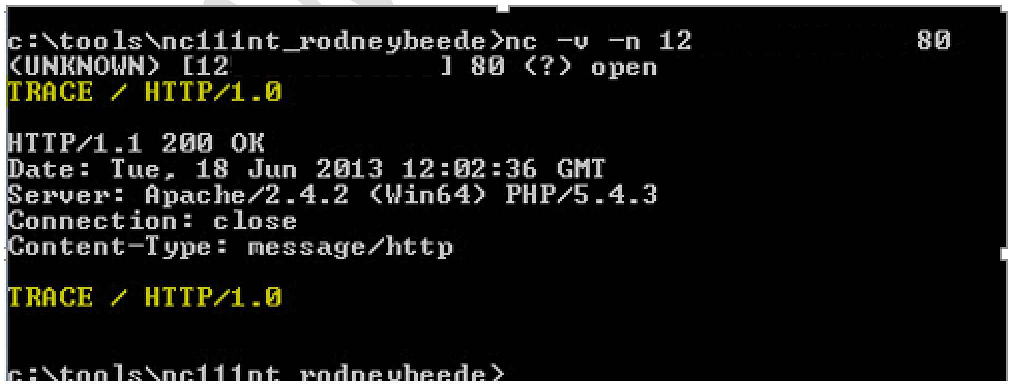
2.3.9 Auto-Complete Attribute Not Set to Off

Risk Level	**Low**
OWASP Category	A6 – Security Misconfiguration
Abstract	Assessor observed that the value of “AutoComplete” attribute is not set to “off”. This allows browsers to store information entered by the user in the cache memory.
Ease of Exploitation	Easy
Impact	Attacker may use this flaw to recover the entered “username” and “password” from browser cache.
Recommendations	It is recommended that the attribute “autocomplete” should be added to the source code for all the input fields accepting sensitive information and the value of this attribute should be set to "off".
Substantiated Assessment	
Affected URLs	https://www.abc.com/WEB_APP/signin/ https://www.abc.com/WEB_APP/wp-login.php
Note	Attacker if gets access to victim’s machine could extract the password from browser cache.
Reference	https://www.owasp.org/index.php/Top_10-2017_A6-Security_Misconfiguration
CWE	

```
                                 <label for="user_login">UserID : </label><input id="user_login" type="text" name="log" value=""
/>
                              </li>
                              <li>
                                 <label for="rememberme">&nbsp;</label><div class="remember-me"><input id="rememberme"
type="checkbox" name="rememberme"  value="forever" /><label> Remember my email address </label></div>
                              </li>
                              <li>
                                 <label for="user_pass">Password :</label><input id="user_pass" type="password" name="pwd"
value="" />
                              </li>
                              <li>
                                 <label>&nbsp;</label><div><a href="[redacted]forgotpassword">Forgot password?
</a><button id="wp-submit" class="button f-right" type="submit" name="wp-submit">Sign In</button></div>
                              </li>
                           </ul>
```

2.3.10 HTTP TRACE Method Enabled

Risk Level	**Low**
OWASP Category	A6 – Security Misconfiguration
Abstract	Assessor observed that HTTP TRACE method is enabled on remote server. The HTTP TRACE method is normally used to return the full HTTP request back to the requesting client for proxy-debugging purposes.
Ease of Exploitation	Easy
Impact	An attacker can create a webpage using XMLHTTP, ActiveX, or XMLDOM to cause a client to issue a TRACE request and capture the client's cookies. This effectively results in a Cross-Site Scripting attack.
Recommendations	It is recommended to disable the TRACE method in the web server configuration. Unfortunately, it is not possible to do this using the Limit directive since the processing for the TRACE request skips this authorization checking. Instead the following lines can be added which make use of the mod_rewrite module. RewriteEngine on RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK) RewriteRule .* - [F]

Substantiated Assessment	
	`c:\tools\nc111nt_rodneybeede>nc -v -n 12[redacted] 80`
	`(UNKNOWN) [12[redacted]] 80 (?) open`
	`TRACE / HTTP/1.0`
	
	`HTTP/1.1 200 OK`
	`Date: Tue, 18 Jun 2013 12:02:36 GMT`
	`Server: Apache/2.4.2 (Win64) PHP/5.4.3`
	`Connection: close`
	`Content-Type: message/http`
	
	`TRACE / HTTP/1.0`
	
	`c:\tools\nc111nt_rodneybeede>`

Affected URL	https://www.abc.com/WEB_APP
Note	Attacker could modify the requests made by victim via an application level proxy and could get sensitive information like passwords etc.
Reference	https://www.owasp.org/index.php/Top_10-2017_A6-Security_Misconfiguration
CWE	