# I-HACK FINAL ROUNDS WRITEUP
# BUDAK PULUHAN (UNISEL)

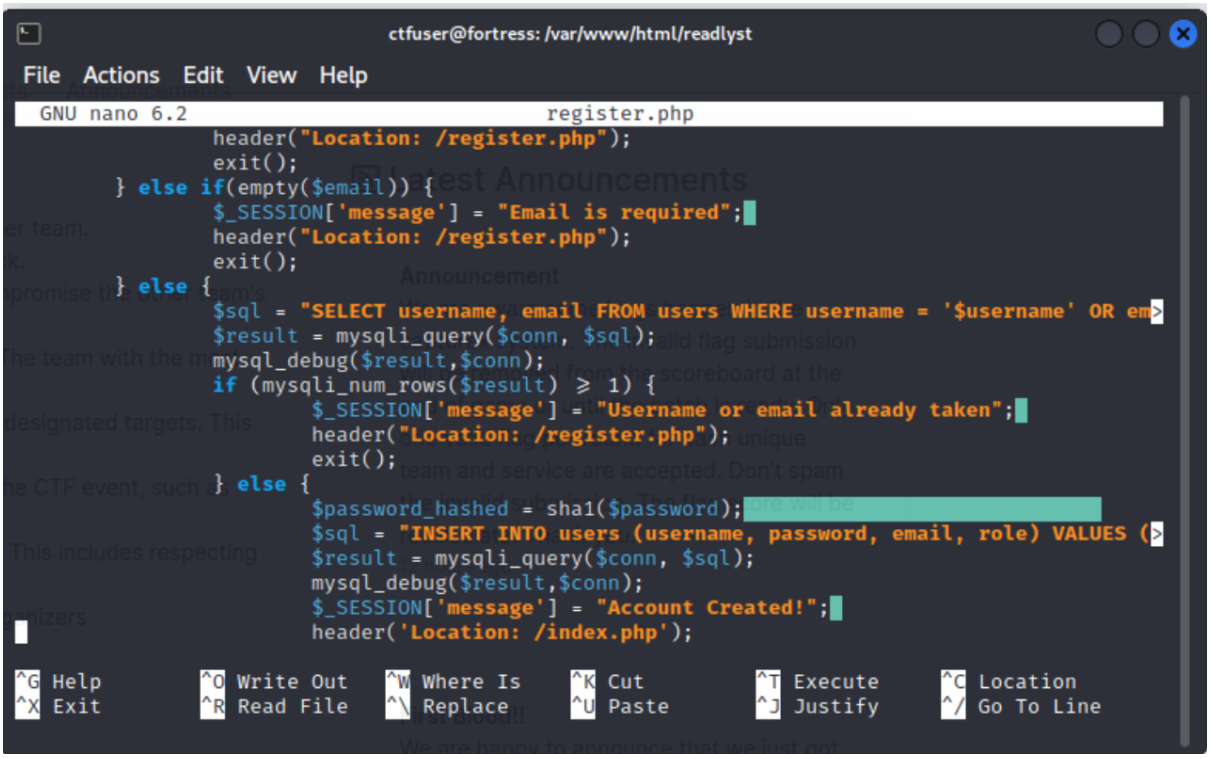
**Team Members :**
Muhammad Firdaus bin Amran
Muhammad Shariff bin Umar
Muhammad Hazmi bin Abdul Basik

# I-Hack Final Rounds Writeup

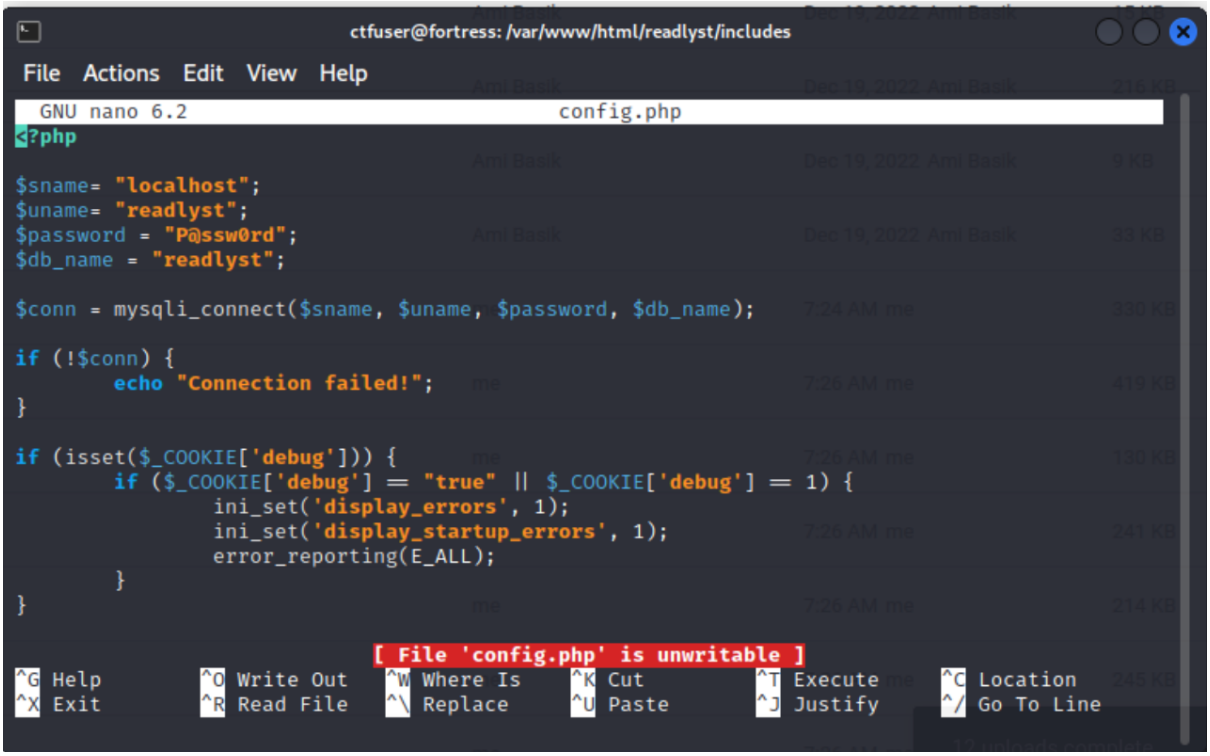
## 1. Readlyst Port 80

Browse to the **/var/www/html** and you will find a **readlyst** folder. Inside the folder you will find various kinds of file such as registration, login, update etc.

We could see that the registered password in stored in **SHA1** which is unsafe.

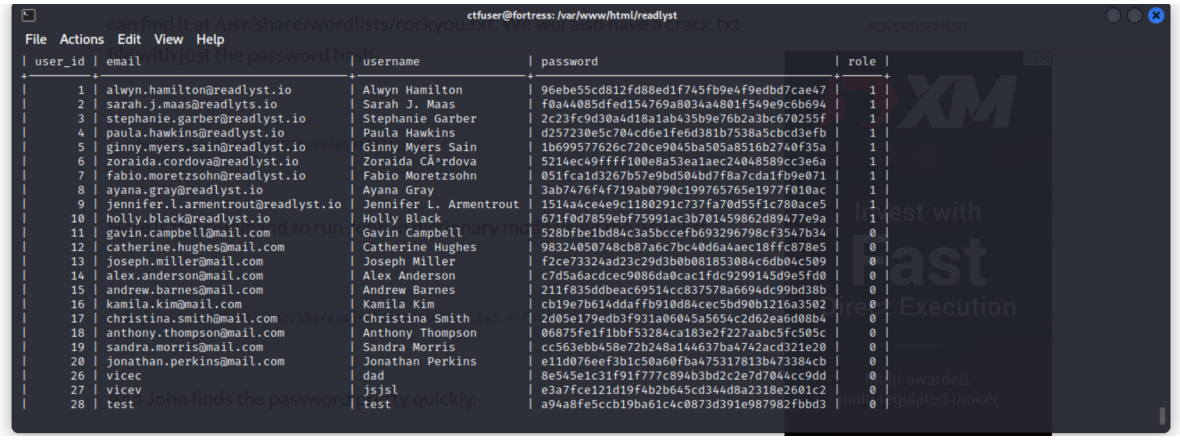



Then, we browse into **includes** folder and we found **config.php** which have the details to database connection.



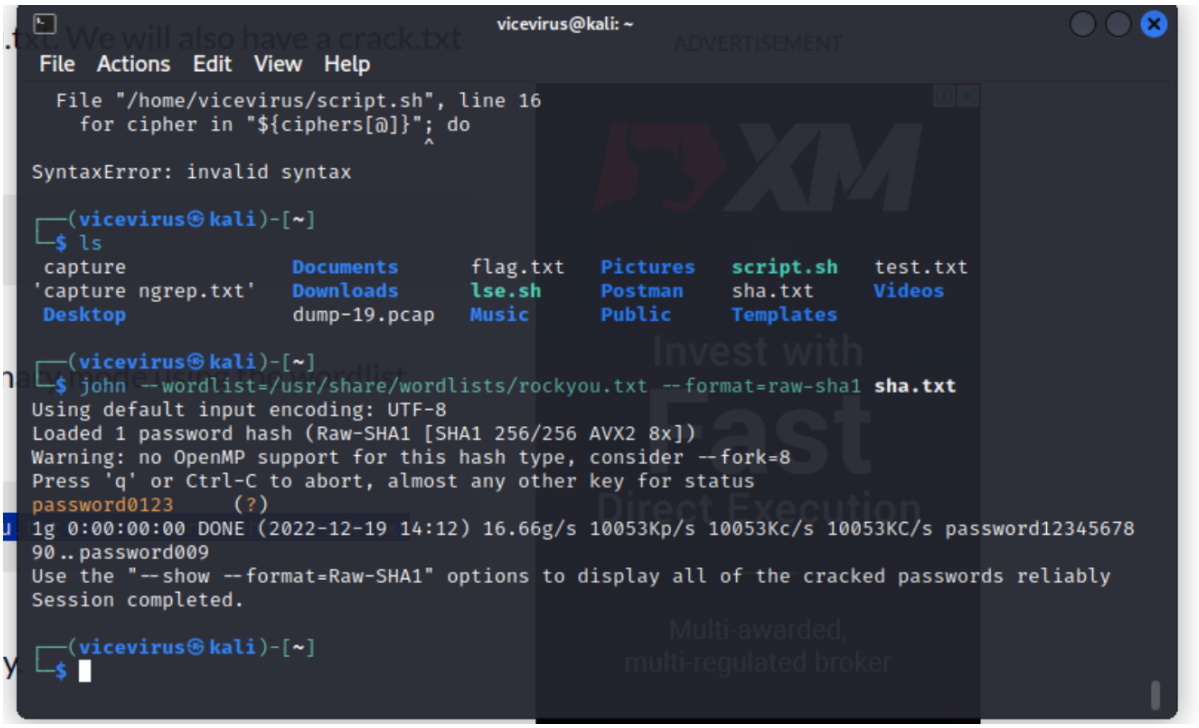
Next, we tried to register an account on the website. It was successful and I was able to put anything other than emails in the email field. (**No validation)**

Then, we tried logging in to **mysql** using the stated credentials inside **config.php**. We found a table named **users** and tried to take a look at it. We use '*select * from users'* to fetch the users data inside the table.



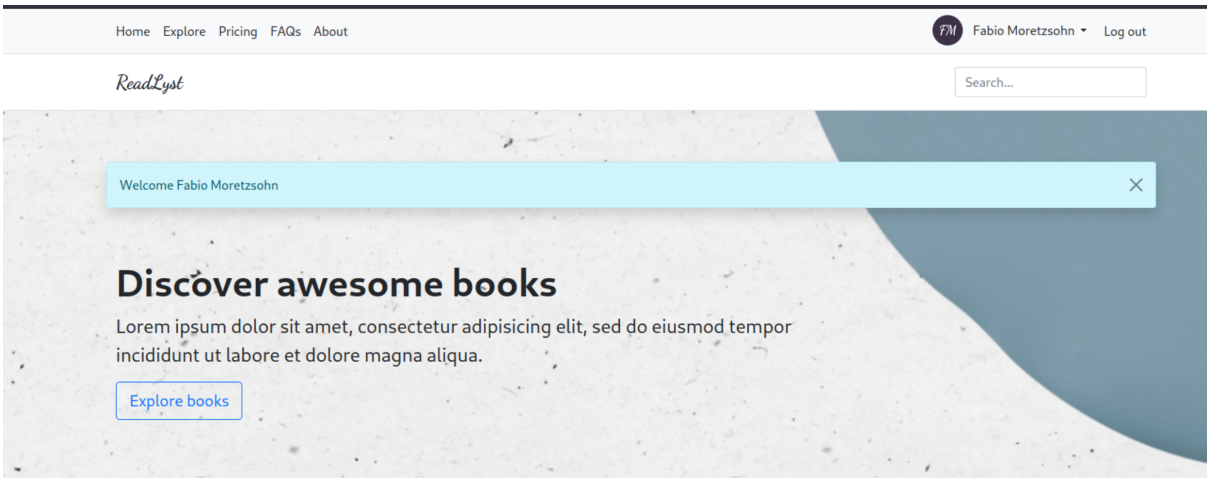

What we found was the user that we have created existed in the database. And the column **role** looks interesting. We took the chance to get one of the **password** stored in the database and tried to run bruteforce using **john the ripper** with **rockyou wordlist**.

We took fabio.moretzsohn@readlyst.io **password** for this one.

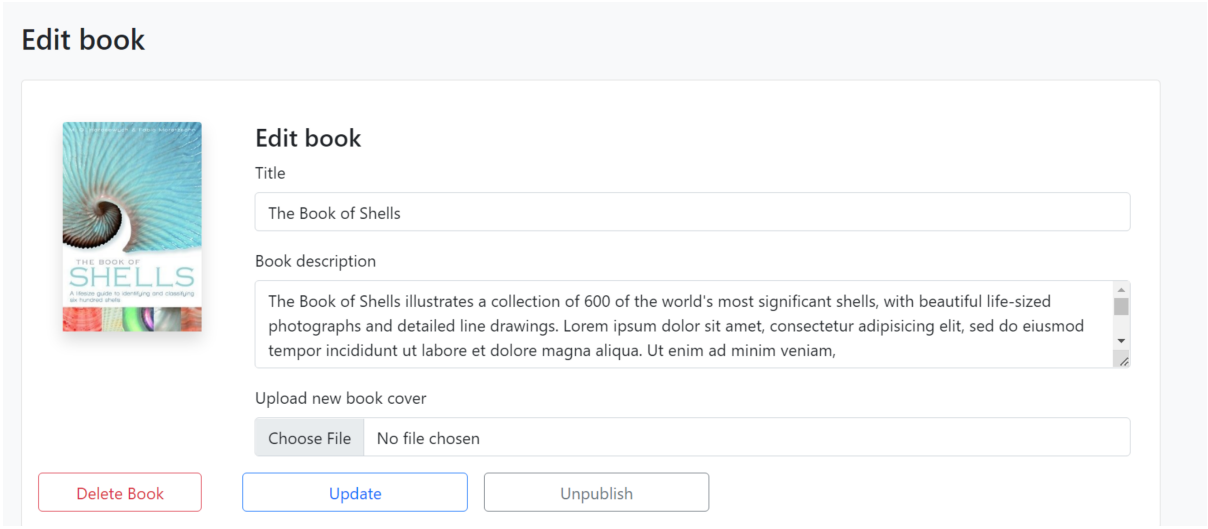



And we found a match for the **SHA1** stored password.

Then, we tried logging in to the website and it was successful.

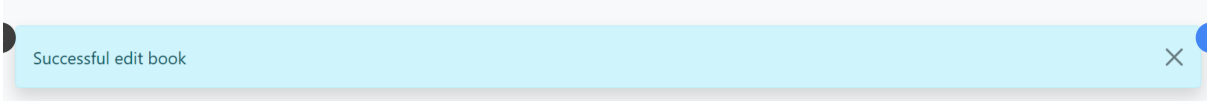



We explored almost every part of website and we found an interesting place for us to upload an **RCE payload.**

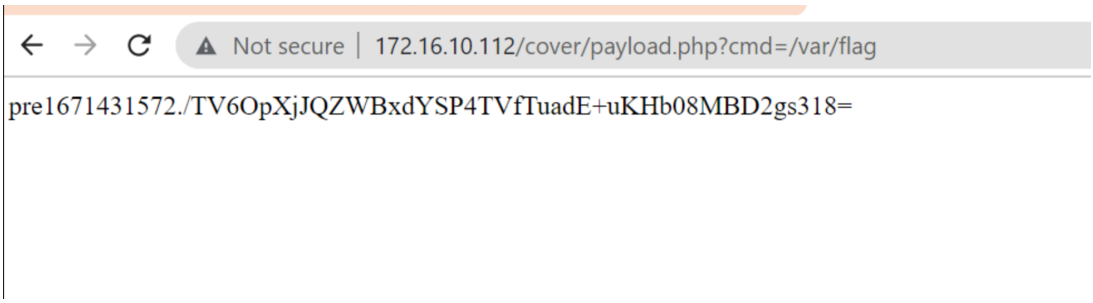



We found that **author/books/update.php** does not check for file extensions or MIME. It will allow any file to be uploaded to the server. In this case, we crafted a payload and uploaded a **.php** file through the **upload** field to the server.


And done! We have successfully uploaded the **payload** to the server.

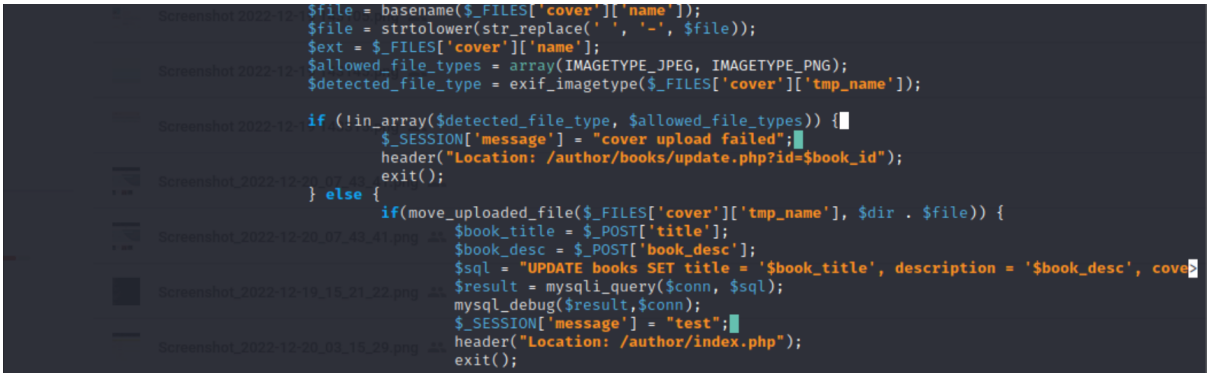


We could access our payload through **/cover/** folder and send **get** request for **/var/flag** execution. And we found the **flag!**




**Remedy / Patch**

Updated the code with **file type** exclusion.



```
$file = basename($_FILES['cover']['name']);
$file = strtolower(str_replace(' ', '-', $file));
$ext = $_FILES['cover']['name'];
$allowed_file_types = array(IMAGETYPE_JPEG, IMAGETYPE_PNG);
$detected_file_type = exif_imagetype($_FILES['cover']['tmp_name']);

if (!in_array($detected_file_type, $allowed_file_types)) {
        $_SESSION['message'] = "cover upload failed";
        header("Location: /author/books/update.php?id=$book_id");
        exit();
} else {
        if(move_uploaded_file($_FILES['cover']['tmp_name'], $dir . $file)) {
                $book_title = $_POST['title'];
                $book_desc = $_POST['book_desc'];
                $sql = "UPDATE books SET title = '$book_title', description = '$book_desc', cove
                $result = mysqli_query($conn, $sql);
                mysql_debug($result,$conn);
                $_SESSION['message'] = "test";
                header("Location: /author/index.php");
                exit();
```

**$allowed_file_types** stated above will only allow JPEG and PNG to be uploaded.
**$detected_file_type** will check what kind of file to be uploaded.

If the file is not in the **allowed file types,** they will not be uploaded.

We use the **payload** uploaded to use the **'cp'** command to copy **patch.php** file from **/cover/** folder to replace **../author/books/update.php**.

**'cp patch.php ../author/books/update.php'**

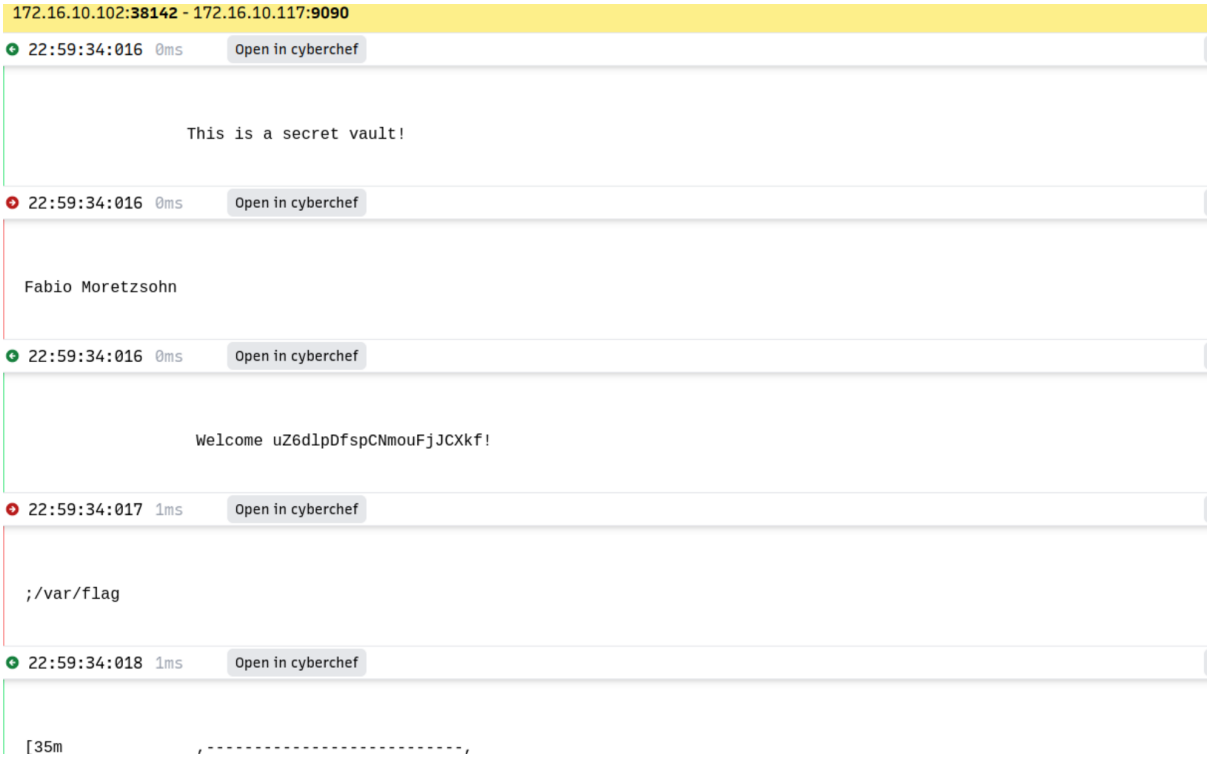
## 2. PassBook  Port 9090 (Unintended solution)

The way we found this flag is by sniffing the network of players trying to attack our server.
We set up a **tcpdump** pipe to write **.pcap** file directly onto our own laptop.

```
vicevirus@viceviruss-MacBook-Air ~ % tcpdump -i ens160 -w - not port 22 | \ ssh
root@192.168.10.136 tcpdump -r - -w -w /home/vicevirus/Downloads/tulip/services/
test_pcap/capture_%H_%M.pcap  -G 3600 -l -vvv -s 0 -Z root
```

This line of command is quite unsafe because we are using our **root** directly to write files locally from outside.

We analyzed the packet captured using **tulip.** And we found attacks toward our port 9090. From there, we learned that to get the flag we just have to input the correct words.



Then, we tried to run our own attacks replicating what the attacks before did.


And we found the **flag!**