Ammar Meslmani - CBS-01

a.meslmani@innopolis.university

the repo link to check the output of this assignment: full report

Lab 2:

- let's unzip the file, check its type, and try to run it:

```
ammar@ubuntu:~/Desktop/lab2$ tar -xvzf hack_app.tar.gz
hack_app
ammar@ubuntu:~/Desktop/lab2$ file hack_app
hack_app: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=0813fa481818746
28c171f5ed6f0f48b6af0d844, for GNU/Linux 3.2.0, not stripped
ammar@ubuntu:~/Desktop/lab2$ ./hack_app
./hack_app: error while loading shared libraries: libcrypto.so.1.1: cannot open shared object file: No such file or directory
```

- there is an issue because `libssl1.1` is missing
- let's solve the issue

```
ammar@ubuntu:~/Desktop/lab2$ wget http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/libssl1.1_1.1.0g-2ubuntu4_amd64.deb
--2025-04-19 22:03:52--  http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/libssl1.1_1.1.0g-2ubuntu4_amd64.deb
Resolving archive.ubuntu.com (archive.ubuntu.com)... 185.125.190.83, 91.189.91.81, 185.125.190.81, ...
Connecting to archive.ubuntu.com (archive.ubuntu.com)|185.125.190.83|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1128092 (1.1M) [application/vnd.debian.binary-package]
Saving to: ‘libssl1.1_1.1.0g-2ubuntu4_amd64.deb’

libssl1.1_1.1.0g-2ubuntu4_amd64.deb     100%[=========================================================================>]   1.08M   816KB/s    in 1.3s

2025-04-19 22:03:54 (816 KB/s) - ‘libssl1.1_1.1.0g-2ubuntu4_amd64.deb’ saved [1128092/1128092]

ammar@ubuntu:~/Desktop/lab2$ sudo dpkg -i libssl1.1_1.1.0g-2ubuntu4_amd64.deb
Selecting previously unselected package libssl1.1:amd64.
(Reading database ... 261476 files and directories currently installed.)
Preparing to unpack libssl1.1_1.1.0g-2ubuntu4_amd64.deb ...
Unpacking libssl1.1:amd64 (1.1.0g-2ubuntu4) ...
Setting up libssl1.1:amd64 (1.1.0g-2ubuntu4) ...
Processing triggers for libc-bin (2.40-1ubuntu3.1) ...
```

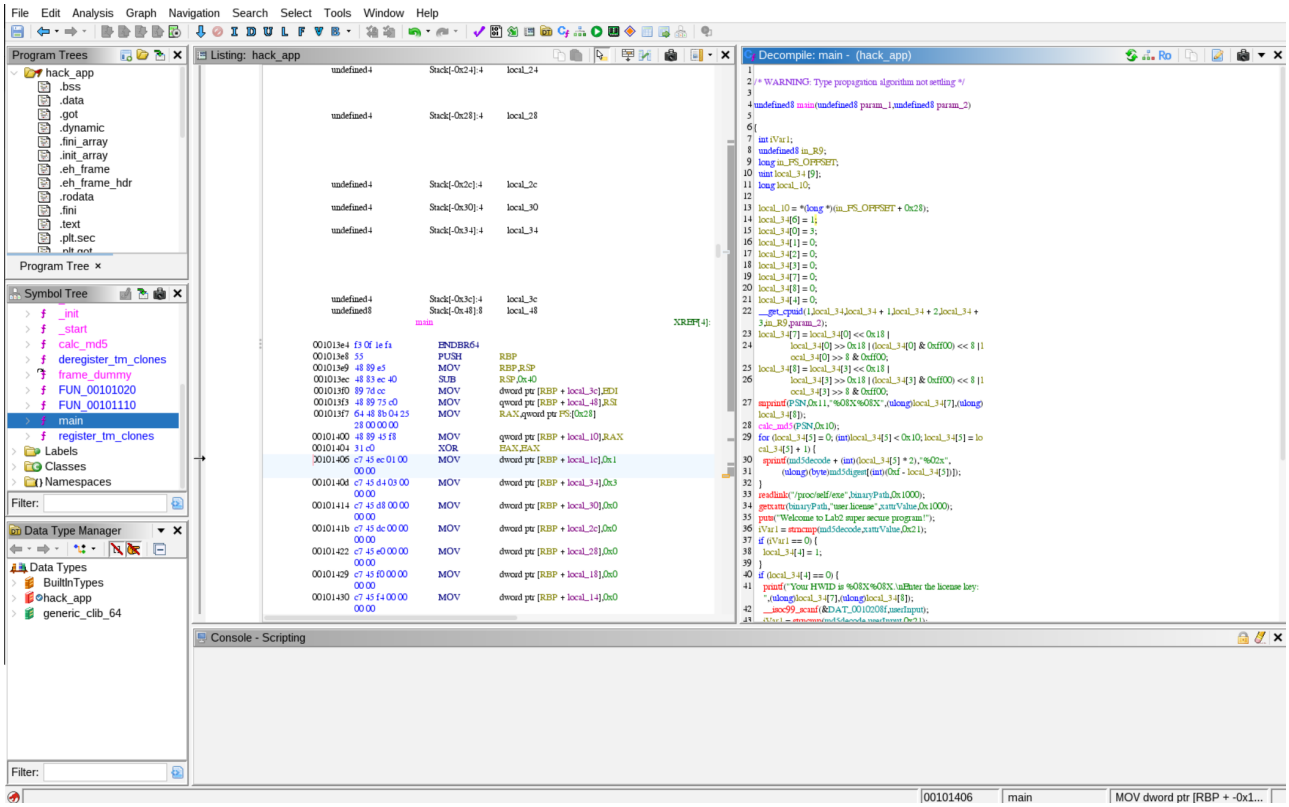
- all the dependencies are resolved now

```
ammar@ubuntu:~/Desktop/lab2$ ldd hack_app
        linux-vdso.so.1 (0x00007c2da563b000)
        libcrypto.so.1.1 => /lib/x86_64-linux-gnu/libcrypto.so.1.1 (0x00007c2da5000000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007c2da4c00000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007c2da5613000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007c2da560e000)
        /lib64/ld-linux-x86-64.so.2 (0x00007c2da563d000)
```

- now let's try to run the program again:

```
ammar@ubuntu:~/Desktop/lab2$ ./hack_app
Welcome to Lab2 super secure program!
Your HWID is 810F8100FFFB8B17.
Enter the license key: asdfasdf
Provided key is wrong! App is closing!
Press Enter to continue...
```

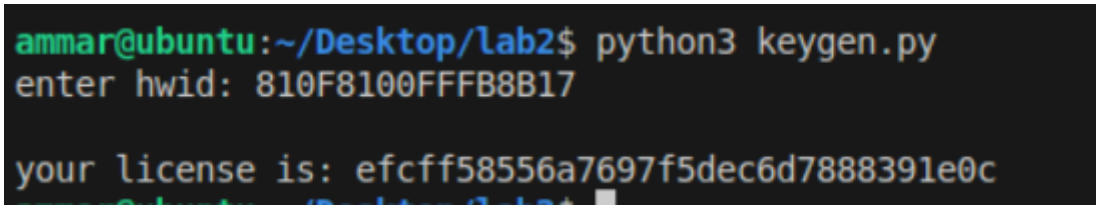
- let's analyze the program using `Ghidra`:

- 

- now let's create a python keygen:

```
import hashlib

def generate_key(hwid):
    md5 = hashlib.md5(hwid.encode()).digest()

    # reverse md5
    reversed_md5 = md5[::-1]

    return reversed_md5.hex()

def main():
    hwid = input("enter hwid: ").strip()

    license_key = generate_key(hwid)
    print(f"\nyour license is: {license_key}")

if __name__ == "__main__":
    main()
```

- let's run the keygen and get the license:

- 

- let's check the license generated by the keygen:

```
ammar@ubuntu:~/Desktop/lab2$ ./hack_app
Welcome to Lab2 super secure program!
Your HWID is 810F8100FFFB8B17.
Enter the license key: efcff58556a7697f5dec6d7888391e0c
Now you app is activated! Thanks for purchasing!
Press Enter to continue...
```

- now let's patch the following instruction so that `iVar1 = 0` to enforce setting `local_34[4]` to 1

```
iVar1 = strncmp(md5decode,xattrValue,0x21);
if (iVar1 == 0) {
  local_34[4] = 1;
}
```

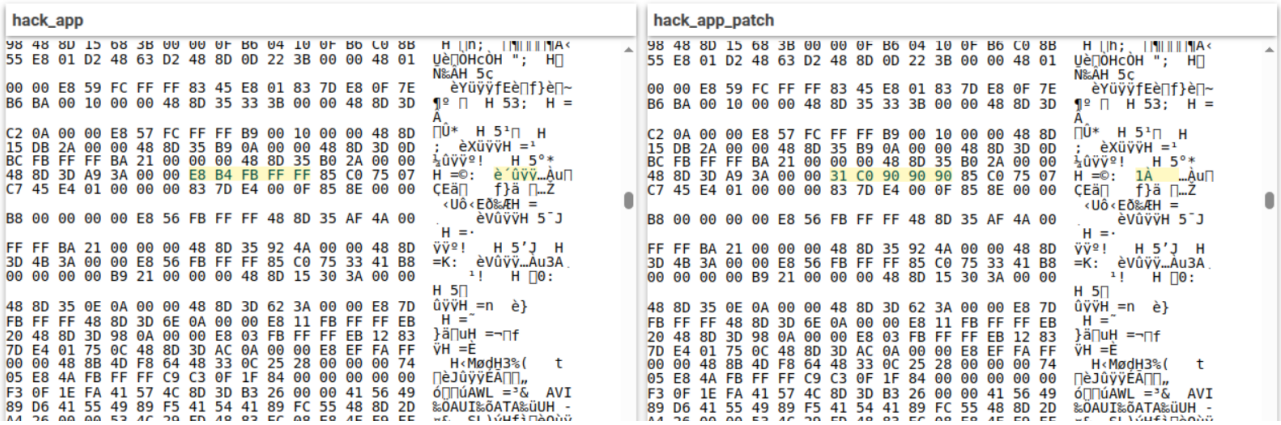
- let's use `XOR EAX, EAX` which is fast operation which guarantees that `EAX` will be set to 0 because of the nature of `XOR` operation

```
00101597 31 c0          XOR     EAX,EAX
00101599 90             NOP
0010159a 90             NOP
0010159b 90             NOP
0010159c 85 c0          TEST    EAX,EAX
0010159e 75 07          JNZ     LAB_001015a7
001015a0 c7 45 e4 01 00 MOV     dword ptr [RBP + local_24],0x1
         00 00
```

- now let's export and run the new patched program:

```
ammar@ubuntu:~/Desktop/lab2$ chmod +x hack_app_patch
ammar@ubuntu:~/Desktop/lab2$ ./hack_app_patch
Welcome to Lab2 super secure program!
Your app is licensed to this PC!
Press Enter to continue...
ammar@ubuntu:~/Desktop/lab2$
```

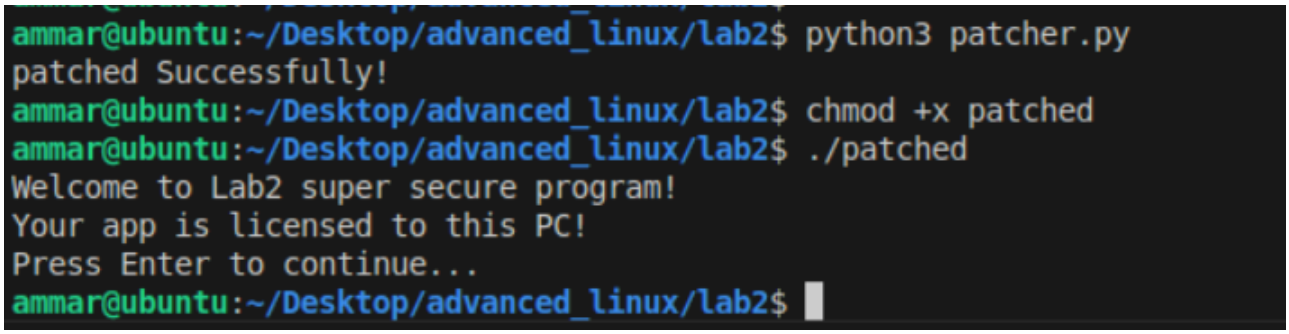
- now let's create a python script which will patch given `hack_app` program, to do this let's compare the binaries of the patched and the original version of `hack_app`:



- now let's create `patcher.py` which will apply the modification shown in the previous comparison:

```
def patch_binary_file(input_file, output_file, original_line,
patched_line):
    # hex strings to bytes
    if isinstance(original_line, str):
        original_line = bytes.fromhex(original_line.replace(" ", ""))
    if isinstance(patched_line, str):
        patched_line = bytes.fromhex(patched_line.replace(" ", ""))

    # open first file
    with open(input_file, 'rb') as f:
        data = f.read()

    # replace line
    modified_data = data.replace(original_line, patched_line)

    if modified_data == data:
        print("warning: the sequence was not found")
    else:
        print("patched Successfully!")

    # update the second file
    with open(output_file, 'wb') as f:
        f.write(modified_data)

# original line
original = "48 8D 3D A9 3A 00 00 E8 B4 FB FF FF 85 C0 75 07"
# patched line
patched = "48 8D 3D A9 3A 00 00 31 C0 90 90 90 85 C0 75 07"


patch_binary_file("hack_app", "patched", original, patched)
```

- let's run it and try running the generated program

- 

- it works successfully!