Lab 2: GDB

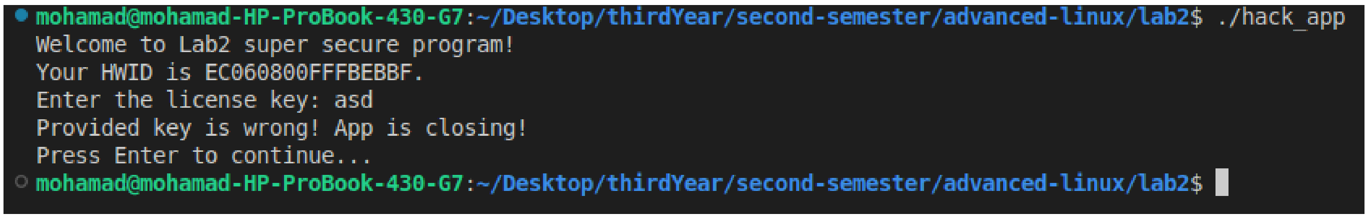
Task Overview

In this lab, we explored various debugging and reverse engineering tools, including GDB, strace, GHIDRA, and ldd. The main objectives were to:

- Analyze a given application that requires a hardware-based license.
- Implement two artifacts:
    1. A keygen to generate valid licenses.
    2. A binary patch to disable licensing checks completely.

Step 1: Running the Application

We start by executing the provided hack application.



```
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/lab2$ ./hack_app
Welcome to Lab2 super secure program!
Your HWID is EC060800FFFBEBBF.
Enter the license key: asd
Provided key is wrong! App is closing!
Press Enter to continue...
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/lab2$
```

Step 2: Installing Required Libraries

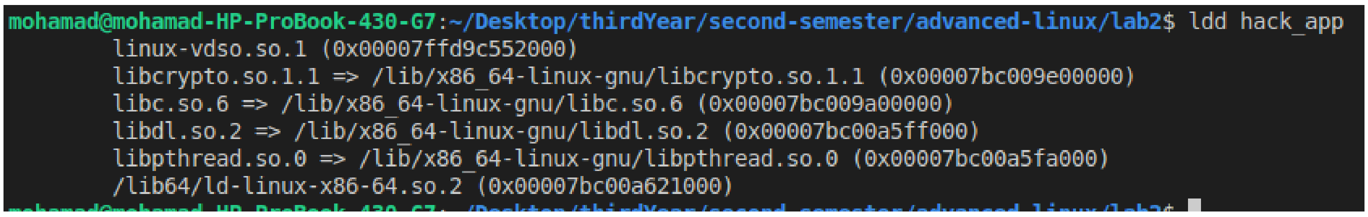
The application depends on `libcrypto.so.1.1`, which must be installed manually. This is done using the following command:

```
wget http://archive.ubuntu.com/ubuntu/pool/main/o/openssl/libssl1.1_1.1.0g-2ubuntu4_amd64.deb
```

After installation, we verify that all required libraries are correctly installed using `ldd`:

```
ldd hack_app
```

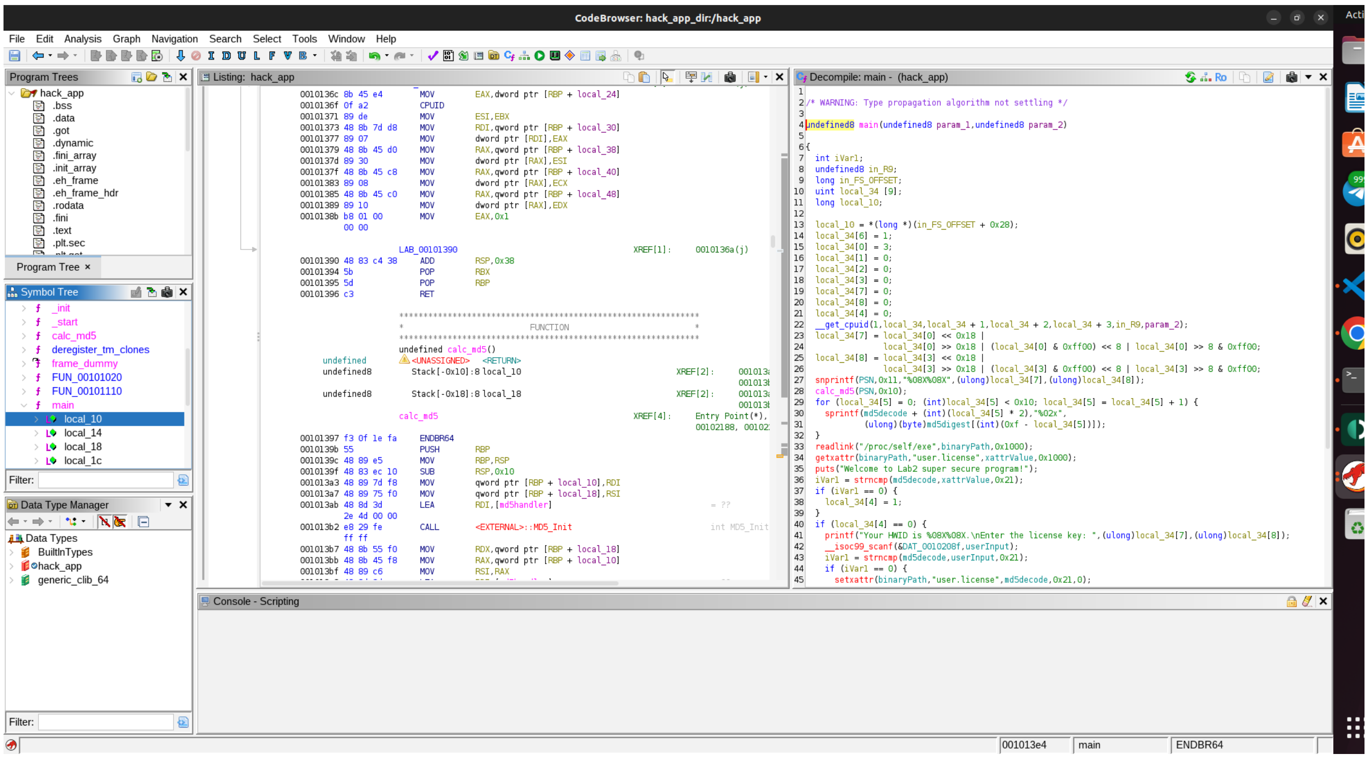
Output confirms all dependencies are resolved:



```
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/lab2$ ldd hack_app
        linux-vdso.so.1 (0x00007ffd9c552000)
        libcrypto.so.1.1 => /lib/x86_64-linux-gnu/libcrypto.so.1.1 (0x00007bc009e00000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007bc009a00000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007bc00a5ff000)
        libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007bc00a5fa000)
        /lib64/ld-linux-x86-64.so.2 (0x00007bc00a621000)
```

Step 3: Reverse Engineering with GHIDRA

After installing GHIDRA, we create a project and import `hack_app` for analysis.



```
0010136c 8b 45 e4        MOV        EAX,dword ptr [RBP + local_24]
0010136f 0f a2           CPUID
00101371 89 de           MOV        ESI,EBX
00101373 48 8b 7d d8     MOV        RDI,qword ptr [RBP + local_30]
00101377 89 07           MOV        dword ptr [RDI],EAX
00101379 48 8b 45 d0     MOV        RAX,qword ptr [RBP + local_38]
0010137d 89 30           MOV        dword ptr [RAX],ESI
0010137f 48 8b 45 c8     MOV        RAX,qword ptr [RBP + local_40]
00101383 89 08           MOV        dword ptr [RAX],ECX
00101385 48 8b 45 c0     MOV        RAX,qword ptr [RBP + local_48]
00101389 89 10           MOV        dword ptr [RAX],EDX
0010138b b8 01 00        MOV        EAX,0x1
         00 00

                     LAB_00101390                           XREF[1]:     0010136a(j)
00101390 48 83 c4 38     ADD        RSP,0x38
00101394 5b              POP        RBX
00101395 5d              POP        RBP
00101396 c3              RET

                     **************************************************************
                     *                          FUNCTION                          *
                     **************************************************************
                     undefined calc_md5()
     undefined         <UNASSIGNED>   <RETURN>
     undefined8          Stack[-0x10]:8 local_10                    XREF[2]:     001013a
                                                                                 001013b
     undefined8          Stack[-0x18]:8 local_18                    XREF[2]:     001013a
                                                                                 001013b
                     calc_md5                               XREF[4]:     Entry Point(*),
                                                                         00102188, 00102
00101397 f3 0f 1e fa     ENDBR64
0010139b 55              PUSH       RBP
0010139c 48 89 e5        MOV        RBP,RSP
0010139f 48 83 ec 10     SUB        RSP,0x10
001013a3 48 89 7d f8     MOV        qword ptr [RBP + local_10],RDI
001013a7 48 89 75 f0     MOV        qword ptr [RBP + local_18],RSI
001013ab 48 8d 3d        LEA        RDI,[md5handler]                              = ??
         2e 4d 00 00
001013b2 e8 29 fe        CALL       <EXTERNAL>::MD5_Init                          int MD5_Init
         ff ff
001013b7 48 8b 55 f0     MOV        RDX,qword ptr [RBP + local_18]
001013bb 48 8b 45 f8     MOV        RAX,qword ptr [RBP + local_10]
001013bf 48 89 c6        MOV        RSI,RAX
```

```c

/* WARNING: Type propagation algorithm not settling */

undefined8 main(undefined8 param_1,undefined8 param_2)

{
  int iVar1;
  undefined8 in_R9;
  long in_FS_OFFSET;
  uint local_34 [9];
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_34[6] = 1;
  local_34[0] = 3;
  local_34[1] = 0;
  local_34[2] = 0;
  local_34[3] = 0;
  local_34[7] = 0;
  local_34[8] = 0;
  local_34[4] = 0;
  __get_cpuid(1,local_34,local_34 + 1,local_34 + 2,local_34 + 3,in_R9,param_2);
  local_34[7] = local_34[0] << 0x18 |
                local_34[0] >> 0x18 | (local_34[0] & 0xff00) << 8 | local_34[0] >> 8 & 0xff00;
  local_34[8] = local_34[3] << 0x18 |
                local_34[3] >> 0x18 | (local_34[3] & 0xff00) << 8 | local_34[3] >> 8 & 0xff00;
  snprintf(PSN,0x11,"%08X%08X",(ulong)local_34[7],(ulong)local_34[8]);
  calc_md5(PSN,0x10);
  for (local_34[5] = 0; (int)local_34[5] < 0x10; local_34[5] = local_34[5] + 1) {
    sprintf(md5decode + (int)(local_34[5] * 2),"%02x",
            (ulong)(byte)md5digest[(int)(0xf - local_34[5])]);
  }
  readlink("/proc/self/exe",binaryPath,0x1000);
  getxattr(binaryPath,"user.license",xattrValue,0x1000);
  puts("Welcome to Lab2 super secure program!");
  iVar1 = strncmp(md5decode,xattrValue,0x21);
  if (iVar1 == 0) {
    local_34[4] = 1;
  }
  if (local_34[4] == 0) {
    printf("Your HWID is %08X%08X.\nEnter the license key: ",(ulong)local_34[7],(ulong)local_34[8]);
    __isoc99_scanf(&DAT_0010208f,userInput);
    iVar1 = strncmp(md5decode,userInput,0x21);
    if (iVar1 == 0) {
      setxattr(binaryPath,"user.license",md5decode,0x21,0);
```

Keygen Implementation

We create a Python script to generate valid licenses based on the hardware ID:

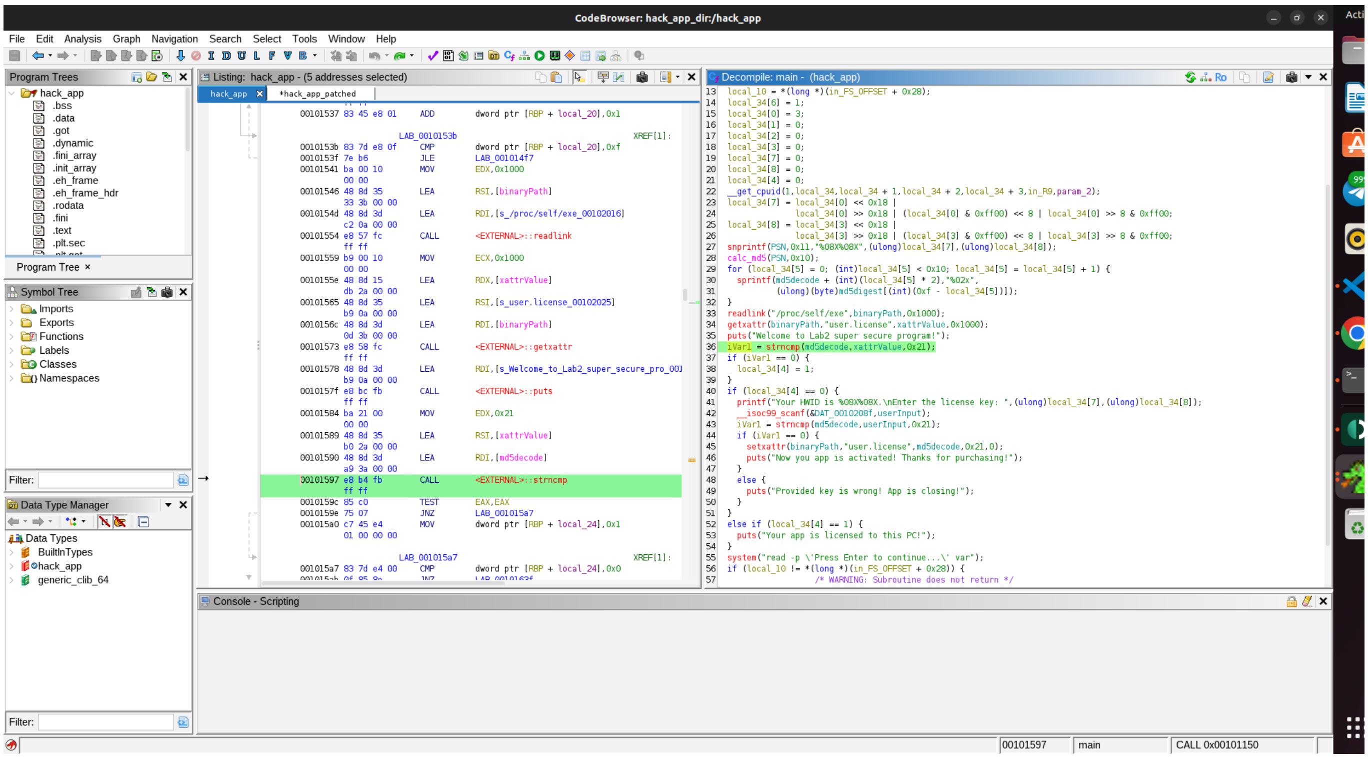
```python
import hashlib

def hash_hardware_id(hardware_id):
    hashed_id = hashlib.md5(hardware_id.encode()).digest()[::-1].hex()
    print("Your hashed id is: ", hashed_id)
    return hashed_id

if __name__ == "__main__":
    hardware_id = input("Enter your HWID: ")
    if len(hardware_id) < 16:
        print("Invalid HWID. Please enter a valid HWID.")
    else:
        print("Your HWID is valid.")
        hash_hardware_id(hardware_id)
```

Modifying the Binary

Using GHIDRA, we locate the variable `iVAr1` and modify its value to zero. This change ensures that `local_34[4]` is set to `1`, triggering the following message:

```
puts("Your app is licensed to this PC!");
```

Before modification

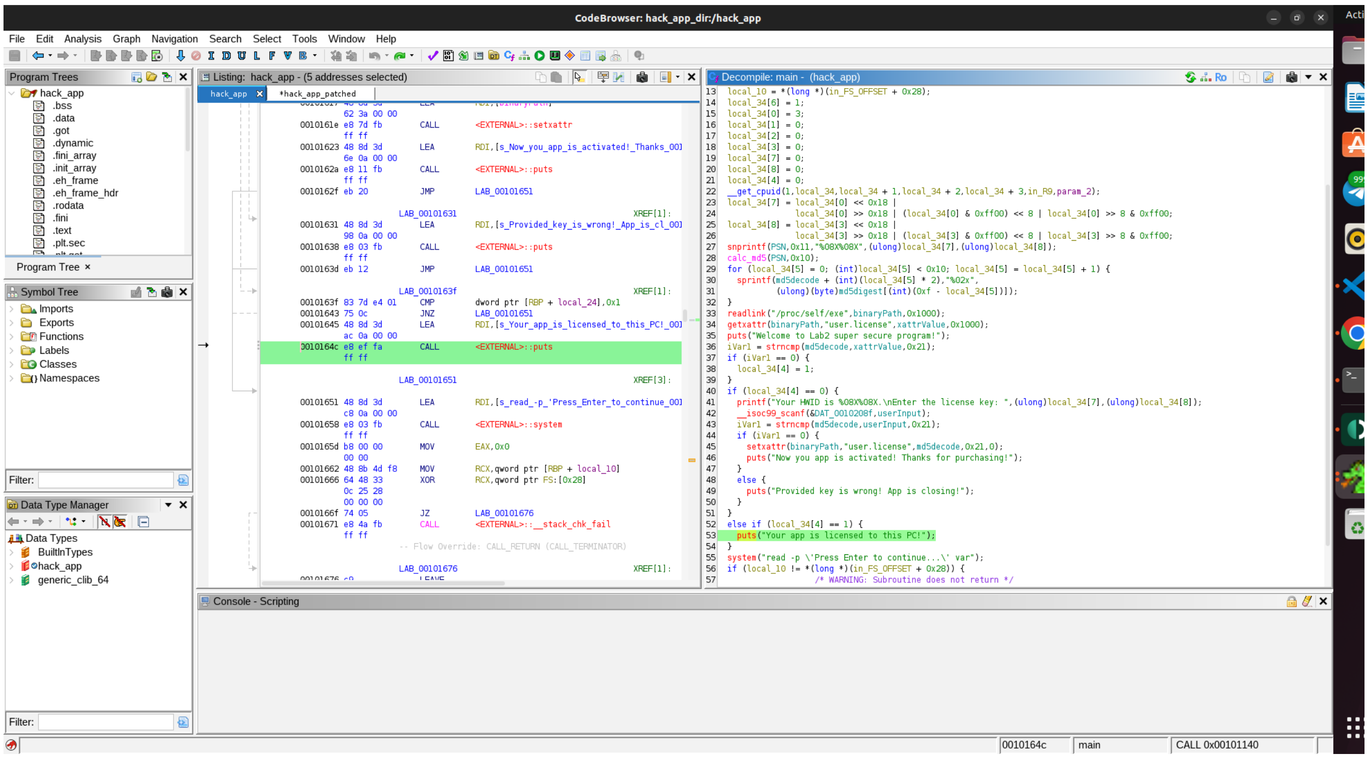


```
00101537 83 45 e8 01     ADD       dword ptr [RBP + local_20],0x1

                   LAB_0010153b                              XREF[1]:
0010153b 83 7d e8 0f     CMP       dword ptr [RBP + local_20],0xf
0010153f 7e b6           JLE       LAB_001014f7
00101541 ba 00 10        MOV       EDX,0x1000
         00 00
00101546 48 8d 35        LEA       RSI,[binaryPath]
         33 3b 00 00
0010154d 48 8d 3d        LEA       RDI,[s_/proc/self/exe_00102016]
         c2 0a 00 00
00101554 e8 57 fc        CALL      <EXTERNAL>::readlink
         ff ff
00101559 b9 00 10        MOV       ECX,0x1000
         00 00
0010155e 48 8d 15        LEA       RDX,[xattrValue]
         db 2a 00 00
00101565 48 8d 35        LEA       RSI,[s_user.license_00102025]
         b9 0a 00 00
0010156c 48 8d 3d        LEA       RDI,[binaryPath]
         0d 3b 00 00
00101573 e8 58 fc        CALL      <EXTERNAL>::getxattr
         ff ff
00101578 48 8d 3d        LEA       RDI,[s_Welcome_to_Lab2_super_secure_pro_00]
         b9 0a 00 00
0010157f e8 bc fb        CALL      <EXTERNAL>::puts
         ff ff
00101584 ba 21 00        MOV       EDX,0x21
         00 00
00101589 48 8d 35        LEA       RSI,[xattrValue]
         b0 2a 00 00
00101590 48 8d 3d        LEA       RDI,[md5decode]
         a9 3a 00 00
00101597 e8 b4 fb        CALL      <EXTERNAL>::strncmp
         ff ff
0010159c 85 c0           TEST      EAX,EAX
0010159e 75 07           JNZ       LAB_001015a7
001015a0 c7 45 e4        MOV       dword ptr [RBP + local_24],0x1
         01 00 00 00

                   LAB_001015a7                              XREF[1]:
001015a7 83 7d e4 00     CMP       dword ptr [RBP + local_24],0x0
```

```c
  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_34[6] = 1;
  local_34[0] = 3;
  local_34[1] = 0;
  local_34[2] = 0;
  local_34[3] = 0;
  local_34[7] = 0;
  local_34[8] = 0;
  local_34[4] = 0;
  __get_cpuid(1,local_34,local_34 + 1,local_34 + 2,local_34 + 3,in_R9,param_2);
  local_34[7] = local_34[0] << 0x18 |
                local_34[0] >> 0x18 | (local_34[0] & 0xff00) << 8 | local_34[0] >> 8 & 0xff00;
  local_34[8] = local_34[3] << 0x18 |
                local_34[3] >> 0x18 | (local_34[3] & 0xff00) << 8 | local_34[3] >> 8 & 0xff00;
  snprintf(PSN,0x11,"%08X%08X",(ulong)local_34[7],(ulong)local_34[8]);
  calc_md5(PSN,0x10);
  for (local_34[5] = 0; (int)local_34[5] < 0x10; local_34[5] = local_34[5] + 1) {
    sprintf(md5decode + (int)(local_34[5] * 2),"%02x",
            (ulong)(byte)md5digest[(int)(0xf - local_34[5])]);
  }
  readlink("/proc/self/exe",binaryPath,0x1000);
  getxattr(binaryPath,"user.license",xattrValue,0x1000);
  puts("Welcome to Lab2 super secure program!");
  iVar1 = strncmp(md5decode,xattrValue,0x21);
  if (iVar1 == 0) {
    local_34[4] = 1;
  }
  if (local_34[4] == 0) {
    printf("Your HWID is %08X%08X.\nEnter the license key: ",(ulong)local_34[7],(ulong)local_34[8]);
    __isoc99_scanf(&DAT_0010208f,userInput);
    iVar1 = strncmp(md5decode,userInput,0x21);
    if (iVar1 == 0) {
      setxattr(binaryPath,"user.license",md5decode,0x21,0);
      puts("Now you app is activated! Thanks for purchasing!");
    }
    else {
      puts("Provided key is wrong! App is closing!");
    }
  }
  else if (local_34[4] == 1) {
    puts("Your app is licensed to this PC!");
  }
  system("read -p \'Press Enter to continue...\' var");
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
```

```
00101597    main    CALL 0x00101150
```

After modification

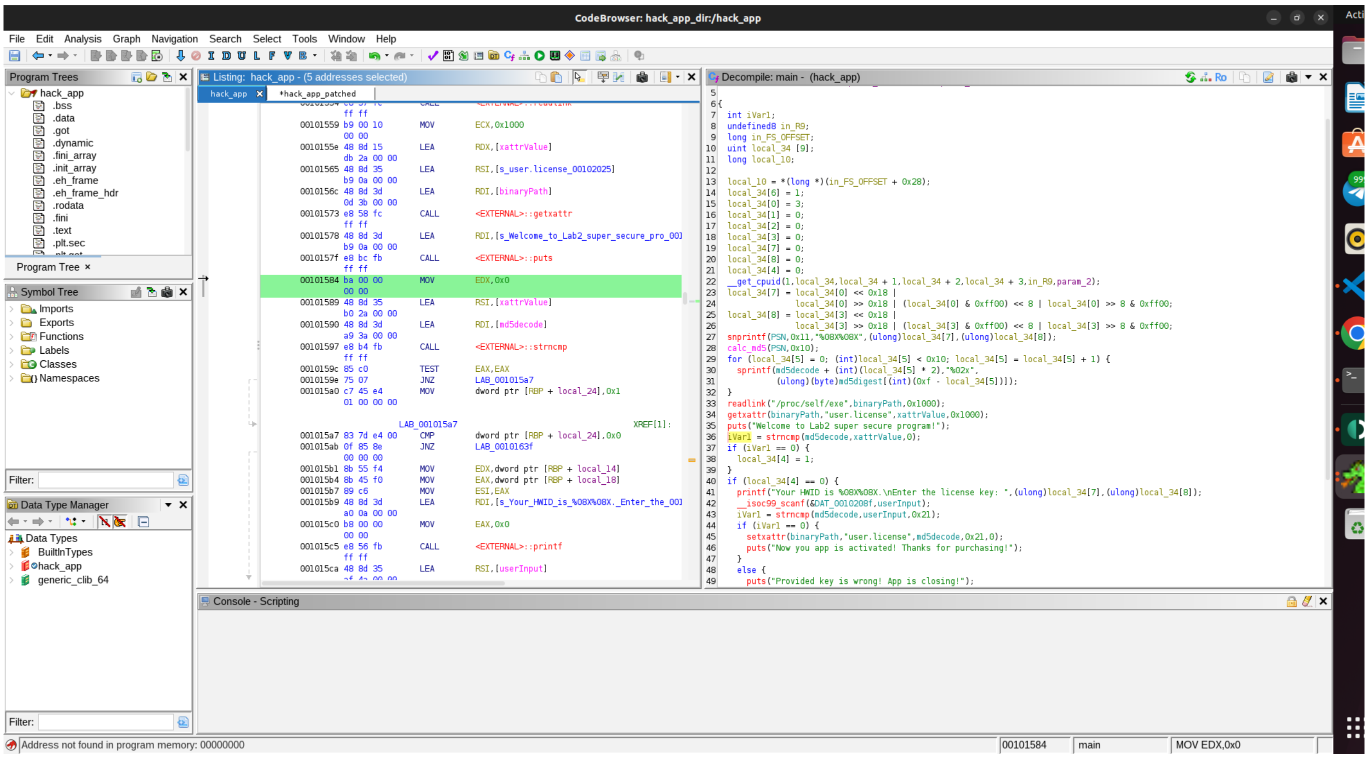


```
         62 3a 00 00
0010161e e8 7d fb        CALL      <EXTERNAL>::setxattr
         ff ff
00101623 48 8d 3d        LEA       RDI,[s_Now_you_app_is_activated!_Thanks_00]
         6e 0a 00 00
0010162a e8 11 fb        CALL      <EXTERNAL>::puts
         ff ff
0010162f eb 20           JMP       LAB_00101651

                   LAB_00101631                              XREF[1]:
00101631 48 8d 3d        LEA       RDI,[s_Provided_key_is_wrong!_App_is_cl_00]
         98 0a 00 00
00101638 e8 03 fb        CALL      <EXTERNAL>::puts
         ff ff
0010163d eb 12           JMP       LAB_00101651

                   LAB_0010163f                              XREF[1]:
0010163f 83 7d e4 01     CMP       dword ptr [RBP + local_24],0x1
00101643 75 0c           JNZ       LAB_00101651
00101645 48 8d 3d        LEA       RDI,[s_Your_app_is_licensed_to_this_PC!_00]
         ac 0a 00 00
0010164c e8 ef fa        CALL      <EXTERNAL>::puts
         ff ff

                   LAB_00101651                              XREF[3]:

00101651 48 8d 3d        LEA       RDI,[s_read_-p_'Press_Enter_to_continue_00]
         c8 0a 00 00
00101658 e8 03 fb        CALL      <EXTERNAL>::system
         ff ff
0010165d b8 00 00        MOV       EAX,0x0
         00 00
00101662 48 8b 4d f8     MOV       RCX,qword ptr [RBP + local_10]
00101666 64 48 33        XOR       RCX,qword ptr FS:[0x28]
         0c 25 28
         00 00 00
0010166f 74 05           JZ        LAB_00101676
00101671 e8 4a fb        CALL      <EXTERNAL>::__stack_chk_fail
         ff ff
                   -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                   LAB_00101676                              XREF[1]:
```

```c
  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_34[6] = 1;
  local_34[0] = 3;
  local_34[1] = 0;
  local_34[2] = 0;
  local_34[3] = 0;
  local_34[7] = 0;
  local_34[8] = 0;
  local_34[4] = 0;
  __get_cpuid(1,local_34,local_34 + 1,local_34 + 2,local_34 + 3,in_R9,param_2);
  local_34[7] = local_34[0] << 0x18 |
                local_34[0] >> 0x18 | (local_34[0] & 0xff00) << 8 | local_34[0] >> 8 & 0xff00;
  local_34[8] = local_34[3] << 0x18 |
                local_34[3] >> 0x18 | (local_34[3] & 0xff00) << 8 | local_34[3] >> 8 & 0xff00;
  snprintf(PSN,0x11,"%08X%08X",(ulong)local_34[7],(ulong)local_34[8]);
  calc_md5(PSN,0x10);
  for (local_34[5] = 0; (int)local_34[5] < 0x10; local_34[5] = local_34[5] + 1) {
    sprintf(md5decode + (int)(local_34[5] * 2),"%02x",
            (ulong)(byte)md5digest[(int)(0xf - local_34[5])]);
  }
  readlink("/proc/self/exe",binaryPath,0x1000);
  getxattr(binaryPath,"user.license",xattrValue,0x1000);
  puts("Welcome to Lab2 super secure program!");
  iVar1 = strncmp(md5decode,xattrValue,0x21);
  if (iVar1 == 0) {
    local_34[4] = 1;
  }
  if (local_34[4] == 0) {
    printf("Your HWID is %08X%08X.\nEnter the license key: ",(ulong)local_34[7],(ulong)local_34[8]);
    __isoc99_scanf(&DAT_0010208f,userInput);
    iVar1 = strncmp(md5decode,userInput,0x21);
    if (iVar1 == 0) {
      setxattr(binaryPath,"user.license",md5decode,0x21,0);
      puts("Now you app is activated! Thanks for purchasing!");
    }
    else {
      puts("Provided key is wrong! App is closing!");
    }
  }
  else if (local_34[4] == 1) {
    puts("Your app is licensed to this PC!");
  }
  system("read -p \'Press Enter to continue...\' var");
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
```

```
0010164c    main    CALL 0x00101140
```

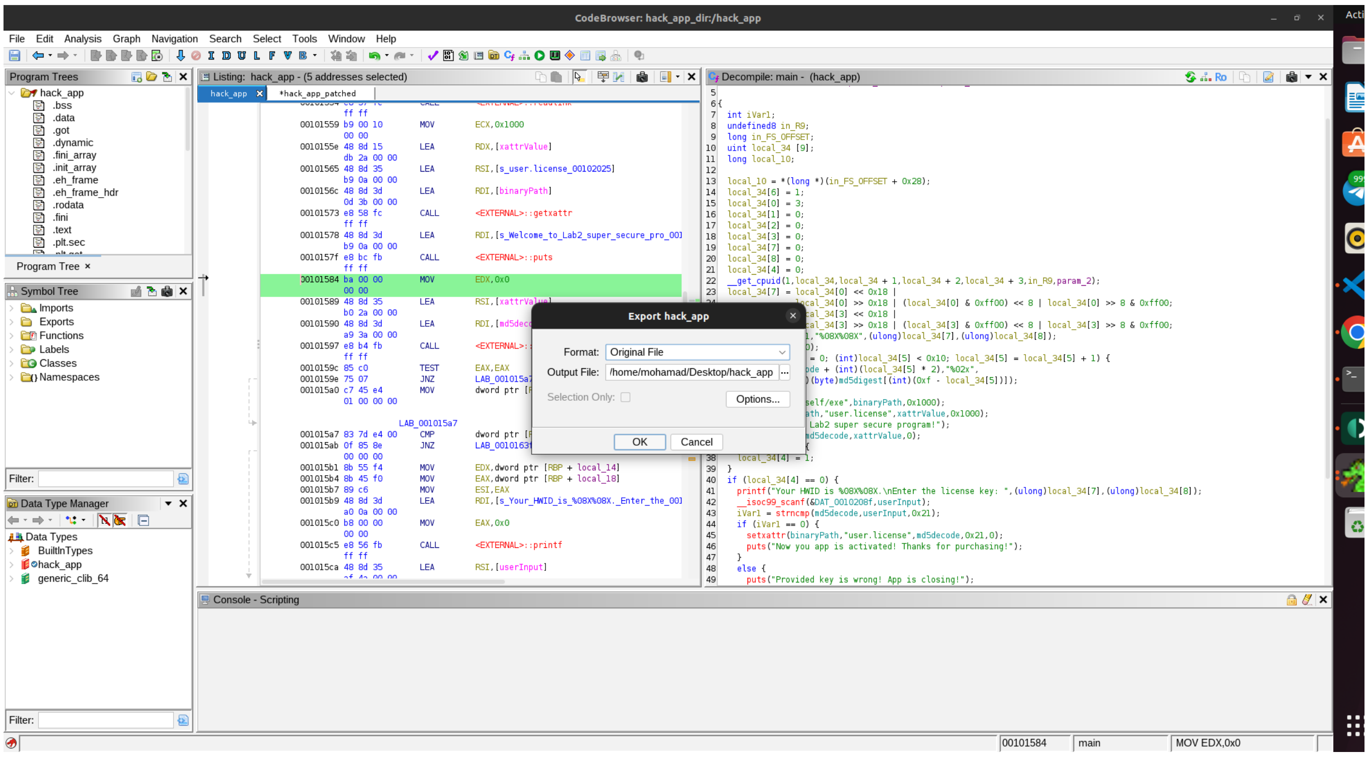
We update the instruction at `0x21` to `0x0` using a patch:



```
00101559 b9 00 10       MOV     ECX,0x1000
         00 00
0010155e 48 8d 15       LEA     RDX,[xattrValue]
         db 2a 00 00
00101565 48 8d 35       LEA     RSI,[s_user.license_00102025]
         b9 0a 00 00
0010156c 48 8d 3d       LEA     RDI,[binaryPath]
         0d 3b 00 00
00101573 e8 58 fc       CALL    <EXTERNAL>::getxattr
         ff ff
00101578 48 8d 3d       LEA     RDI,[s_Welcome_to_Lab2_super_secure_pro_001
         b9 0a 00 00
0010157f e8 bc fb       CALL    <EXTERNAL>::puts
         ff ff
00101584 ba 00 00       MOV     EDX,0x0
         00 00
00101589 48 8d 35       LEA     RSI,[xattrValue]
         b0 2a 00 00
00101590 48 8d 3d       LEA     RDI,[md5decode]
         a9 3a 00 00
00101597 e8 b4 fb       CALL    <EXTERNAL>::strncmp
         ff ff
0010159c 85 c0          TEST    EAX,EAX
0010159e 75 07          JNZ     LAB_001015a7
001015a0 c7 45 e4       MOV     dword ptr [RBP + local_24],0x1
         01 00 00 00

                  LAB_001015a7                          XREF[1]:
001015a7 83 7d e4 00    CMP     dword ptr [RBP + local_24],0x0
001015ab 0f 85 8e       JNZ     LAB_0010163f
         00 00 00
001015b1 8b 55 f4       MOV     EDX,dword ptr [RBP + local_14]
001015b4 8b 45 f0       MOV     EAX,dword ptr [RBP + local_18]
001015b7 89 c6          MOV     ESI,EAX
001015b9 48 8d 3d       LEA     RDI,[s_Your_HWID_is_%08X%08X._Enter_the_001
         a0 0a 00 00
001015c0 b8 00 00       MOV     EAX,0x0
         00 00
001015c5 e8 56 fb       CALL    <EXTERNAL>::printf
         ff ff
001015ca 48 8d 35       LEA     RSI,[userInput]
```

```c
{
  int iVar1;
  undefined8 in_R9;
  long in_FS_OFFSET;
  uint local_34 [9];
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_34[6] = 1;
  local_34[0] = 3;
  local_34[1] = 0;
  local_34[2] = 0;
  local_34[3] = 0;
  local_34[7] = 0;
  local_34[8] = 0;
  local_34[4] = 0;
  __get_cpuid(1,local_34,local_34 + 1,local_34 + 2,local_34 + 3,in_R9,param_2);
  local_34[7] = local_34[0] << 0x18 |
                local_34[0] >> 0x18 | (local_34[0] & 0xff00) << 8 | local_34[0] >> 8 & 0xff00;
  local_34[8] = local_34[3] << 0x18 |
                local_34[3] >> 0x18 | (local_34[3] & 0xff00) << 8 | local_34[3] >> 8 & 0xff00;
  snprintf(PSN,0x11,"%08X%08X",(ulong)local_34[7],(ulong)local_34[8]);
  calc_md5(PSN,0x10);
  for (local_34[5] = 0; (int)local_34[5] < 0x10; local_34[5] = local_34[5] + 1) {
    sprintf(md5decode + (int)(local_34[5] * 2),"%02x",
            (ulong)(byte)md5digest[(int)(0xf - local_34[5])]);
  }
  readlink("/proc/self/exe",binaryPath,0x1000);
  getxattr(binaryPath,"user.license",xattrValue,0x1000);
  puts("Welcome to Lab2 super secure program!");
  iVar1 = strncmp(md5decode,xattrValue,0);
  if (iVar1 == 0) {
    local_34[4] = 1;
  }
  if (local_34[4] == 0) {
    printf("Your HWID is %08X%08X.\nEnter the license key: ",(ulong)local_34[7],(ulong)local_34[8]);
    __isoc99_scanf(&DAT_0010208f,userInput);
    iVar1 = strncmp(md5decode,userInput,0x21);
    if (iVar1 == 0) {
      setxattr(binaryPath,"user.license",md5decode,0x21,0);
      puts("Now you app is activated! Thanks for purchasing!");
    }
    else {
      puts("Provided key is wrong! App is closing!");
```

After making these changes, we export the patched binary:

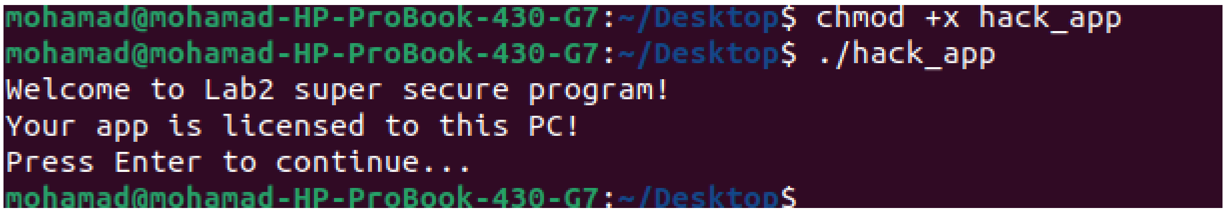


Export hack_app

Format: Original File
Output File: /home/mohamad/Desktop/hack_app
Selection Only: ☐

Finally, we make the modified application executable and run it:

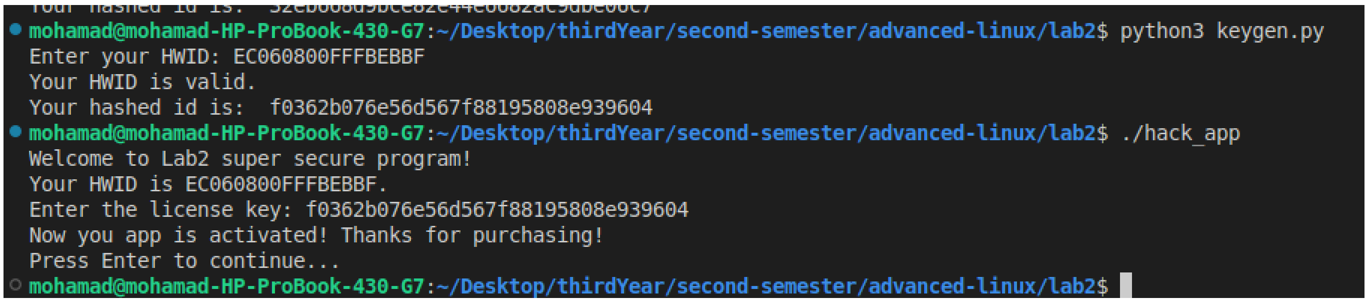
```
chmod +x hack_app
./hack_app
```



```
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop$ chmod +x hack_app
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop$ ./hack_app
Welcome to Lab2 super secure program!
Your app is licensed to this PC!
Press Enter to continue...
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop$
```

Step 4: Testing the Keygen

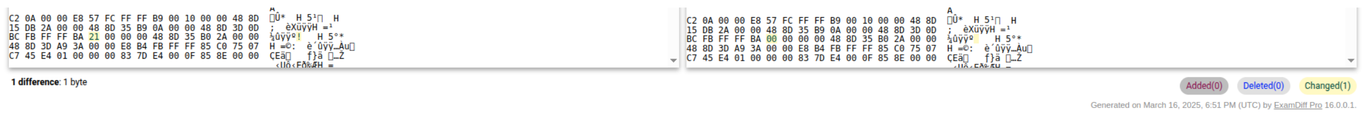
We now test our keygen script with the original application:



```
Your hashed id is:
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/lab2$ python3 keygen.py
Enter your HWID: EC060800FFFBEBBF
Your HWID is valid.
Your hashed id is:  f0362b076e56d567f88195808e939604
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/lab2$ ./hack_app
Welcome to Lab2 super secure program!
Your HWID is EC060800FFFBEBBF.
Enter the license key: f0362b076e56d567f88195808e939604
Now you app is activated! Thanks for purchasing!
Press Enter to continue...
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/lab2$
```

Step 5: Creating a Binary Patch

To automate patching the original binary, we compare the original and patched files using `diffnow.com`:



```
C2 0A 00 00 E8 57 FC FF FF B9 00 10 00 00 48 8D
15 DB 2A 00 00 48 8D 35 B9 0A 00 00 48 8D 3D 0D
BC FB FF FF BA 21 00 00 00 48 8D 35 B0 2A 00 00
48 8D 3D A9 3A 00 00 E8 B4 FB FF FF 85 C0 75 07
C7 45 E4 01 00 00 00 83 7D E4 00 0F 85 8E 00 00
```

1 difference: 1 byte

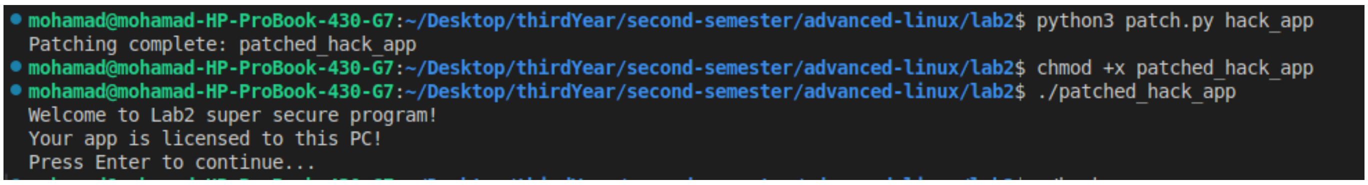
```
C2 0A 00 00 E8 57 FC FF FF B9 00 10 00 00 48 8D
15 DB 2A 00 00 48 8D 35 B9 0A 00 00 48 8D 3D 0D
BC FB FF FF BA 00 00 00 00 48 8D 35 B0 2A 00 00
48 8D 3D A9 3A 00 00 E8 B4 FB FF FF 85 C0 75 07
C7 45 E4 01 00 00 00 83 7D E4 00 0F 85 8E 00 00
```

Generated on March 16, 2025, 6:51 PM (UTC) by ExamDiff Pro 16.0.0.1.

Using this difference, we write a Python script to modify the binary dynamically:

```python
import sys

# Define original and replacement byte sequences
old_bytes = b'\xBA\x21\x00'
new_bytes = b'\xBA\x00\x00'

def patch_binary(filename):
    with open(filename, "rb") as f:
        data = f.read()

    # Find first occurrence
    index = data.find(old_bytes)
    if index == -1:
        print("Pattern BA 21 00 not found!")
        return

    # Replace only the first occurrence
    patched_data = data[:index] + new_bytes + data[index + len(old_bytes):]

    # Save the modified file
    patched_filename = "patched_" + filename
    with open(patched_filename, "wb") as f:
        f.write(patched_data)

    print(f"Patching complete: {patched_filename}")

# Usage: python script.py hack_app
if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python patch.py <filename>")
        sys.exit(1)

    patch_binary(sys.argv[1])
```

running the script :



```
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/lab2$ python3 patch.py hack_app
Patching complete: patched_hack_app
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/lab2$ chmod +x patched_hack_app
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/lab2$ ./patched_hack_app
Welcome to Lab2 super secure program!
Your app is licensed to this PC!
Press Enter to continue...
```

Github Lab02 link