LAB 2

Decompiling the binary through Ghidra we find the following logic in the main function of the program

```
undefined8 main(undefined8 param_1,undefined8 param_2)

{
  int iVar1;
  undefined8 in_R9;
  long in_FS_OFFSET;
  uint local_34;
  undefined4 local_30;
  undefined4 local_2c;
  uint local_28;
  int local_24;
  int local_20;
  undefined4 local_1c;
  uint local_18;
  uint local_14;
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_1c = 1;
  local_34 = 3;
  local_30 = 0;
  local_2c = 0;
  local_28 = 0;
  local_18 = 0;
  local_14 = 0;
  local_24 = 0;
  __get_cpuid(1,&local_34,&local_30,&local_2c,&local_28,in_R9,param_2);
  local_18 = local_34 << 0x18 | local_34 >> 0x18 | (local_34 & 0xff00) << 8 | local_34 >> 8 & 0xff00;
  local_14 = local_28 << 0x18 | local_28 >> 0x18 | (local_28 & 0xff00) << 8 | local_28 >> 8 & 0xff00;
  snprintf(PSN,0x11,"%08X%08X",(ulong)local_18,(ulong)local_14);
  calc_md5(PSN,0x10);
  for (local_20 = 0; local_20 < 0x10; local_20 = local_20 + 1) {
    sprintf(md5decode + local_20 * 2,"%02x",(ulong)(byte)md5digest[0xf - local_20]);
  }
  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_24 = 1;
  }
  if (local_24 == 0) {
    printf("Your HWID is %08X%08X.\nEnter the license key: ",(ulong)local_18,(ulong)local_14);
    __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_24 == 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 */
    __stack_chk_fail();
  }
  return 0;
}
```

Keygen

By analyzing the main function, we can see MD5 mentioned in the code. MD5 is a hashing algorithm and it was clearly used to generate the key for out program

```
snprintf(PSN,0x11,"%08X%08X",(ulong)local_18,(ulong)local_14);
calc_md5(PSN,0x10);
for (local_20 = 0; local_20 < 0x10; local_20 = local_20 + 1) {
    sprintf(md5decode + local_20 * 2,"%02x",(ulong)(byte)md5digest[0xf - local_20]);
}
```

we can clearly see that `local_18` and `local_14` represent the HWID

```
printf("Your HWID is %08X%08X.\nEnter the license key: ",(ulong)local_18,(ulong)local_14);
```

Concluding, MD5 was used to hash the HWID and then it was further convoluted.

Patch

Analyzing the decompiled code we can clearly see a conditional that can be exploited

```
iVar1 = strncmp(md5decode,xattrValue,0x21);
    if (iVar1 == 0) {
    local_24 = 1;
}
```

here local_24 clearly represents a conditional for the presence of a license.

should be changed to

```
0010159e 74 07          JZ          LAB_001015a7
001015a0 c7 45 e4       MOV         dword ptr [RBP + local_24],0x1
         01 00 00 00

                    LAB_001015a7                                XREF[1]:    0010159e(j)
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_001020...  = "Your HWID is %08X%08X.\nEnte
         a0 0a 00 00
001015c0 b8 00 00       MOV         EAX,0x0
         00 00
001015c5 e8 56 fb       CALL        <EXTERNAL>::printf                  int printf(char * __format, ...
         ff ff
001015ca 48 8d 35       LEA         RSI,[userInput]                     = ??
```

```
42  if (iVar1 != 0) {
43    local_24 = 1;
44  }
45  if (local_24 == 0) {
46    printf("Your HWID is %08X%08X.\nEnter the license key: ",(ulo
47    __isoc99_scanf(&DAT_0010208f,userInput);
48    iVar1 = strncmp(md5decode,userInput,0x21);
49    if (iVar1 == 0) {
50      setxattr(binaryPath,"user.license",md5decode,0x21,0);
51      puts("Now you app is activated! Thanks for purchasing!");
52    }
53    else {
54      puts("Provided key is wrong! App is closing!");
55    }
56  }
57  else if (local_24 == 1) {
58    puts("Your app is licensed to this PC!");
59  }
```

Let's find the exact address of the conditional jump using GDB

```
   0x0000000000001597 <+435>:    call   0x1150 <strncmp@plt>
   0x000000000000159c <+440>:    test   eax,eax
   0x000000000000159e <+442>:    jne    0x15a7 <main+451>
   0x00000000000015a0 <+444>:    mov    DWORD PTR [rbp-0x1c],0x1
   0x00000000000015a7 <+451>:    cmp    DWORD PTR [rbp-0x1c],0x0
   0x00000000000015ab <+455>:    jne    0x163f <main+603>
   0x00000000000015b1 <+461>:    mov    edx,DWORD PTR [rbp-0xc]
   0x00000000000015b4 <+464>:    mov    eax,DWORD PTR [rbp-0x10]
```

Jump for **not equal** is represented by **0x75**, whereas jump for **equal** is **0x74**. We should change the byte as shown below

```
00001580: bcfb ffff ba21 0000 0048 8d35 b02a 0000  .....!...H.5.*..
00001590: 488d 3da9 3a00 00e8 b4fb ffff 85c0 7407  H.=.:.........u.
000015a0: c745 e401 0000 0083 7de4 000f 858e 0000  .E......}.......
```