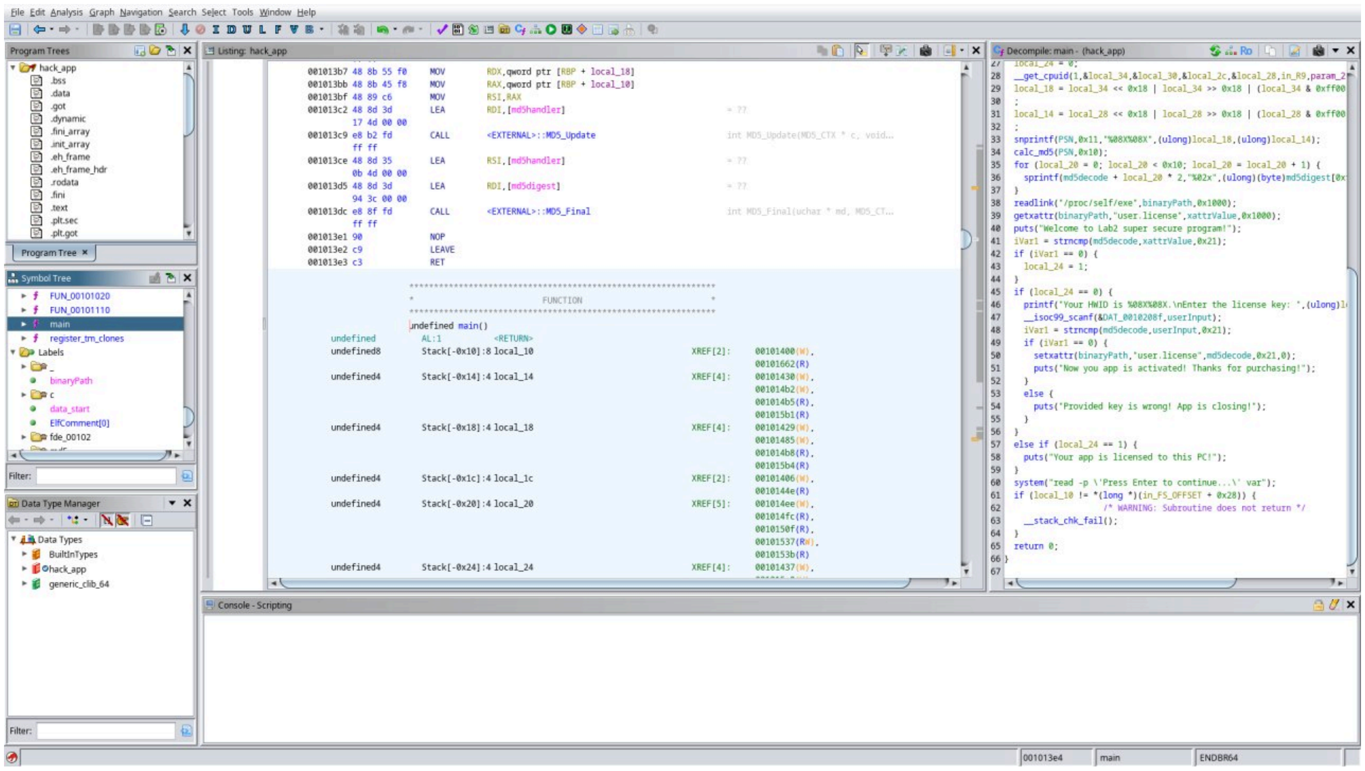
**Lab 2**

**repo: GitHub**

---

## Ghidra exploring



```
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
001013c2 48 8d 3d       LEA     RDI,[md5handler]                  = ??
         17 4d 00 00
001013c9 e8 b2 fd       CALL    <EXTERNAL>::MD5_Update            int MD5_Update(MD5_CTX * c, void...
         ff ff
001013ce 48 8d 35       LEA     RSI,[md5handler]                  = ??
         0b 4d 00 00
001013d5 48 8d 3d       LEA     RDI,[md5digest]                   = ??
         94 3c 00 00
001013dc e8 8f fd       CALL    <EXTERNAL>::MD5_Final             int MD5_Final(uchar * md, MD5_CT...
         ff ff
001013e1 90             NOP
001013e2 c9             LEAVE
001013e3 c3             RET

                 **************************************************************
                 *                          FUNCTION                          *
                 **************************************************************
                 undefined main()
     undefined         AL:1           <RETURN>
     undefined8        Stack[-0x10]:8 local_10            XREF[2]:     00101400(W),
                                                                       00101662(R)
     undefined4        Stack[-0x14]:4 local_14            XREF[4]:     00101430(W),
                                                                       001014b2(W),
                                                                       001014b5(R),
                                                                       001015b1(R)
     undefined4        Stack[-0x18]:4 local_18            XREF[4]:     00101429(W),
                                                                       00101485(W),
                                                                       001014b8(R),
                                                                       001015b4(R)
     undefined4        Stack[-0x1c]:4 local_1c            XREF[2]:     00101406(W),
                                                                       0010144e(R)
     undefined4        Stack[-0x20]:4 local_20            XREF[5]:     001014ee(W),
                                                                       001014fc(R),
                                                                       0010150f(R),
                                                                       00101537(RW),
                                                                       0010153b(R)
     undefined4        Stack[-0x24]:4 local_24            XREF[4]:     00101437(W),
```

```c
  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
  ;
  local_14 = local_28 << 0x18 | local_28 >> 0x18 | (local_28 & 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[0x
  }
  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)l
    __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

### 1 — Algorithm reverse-engineered from `main`

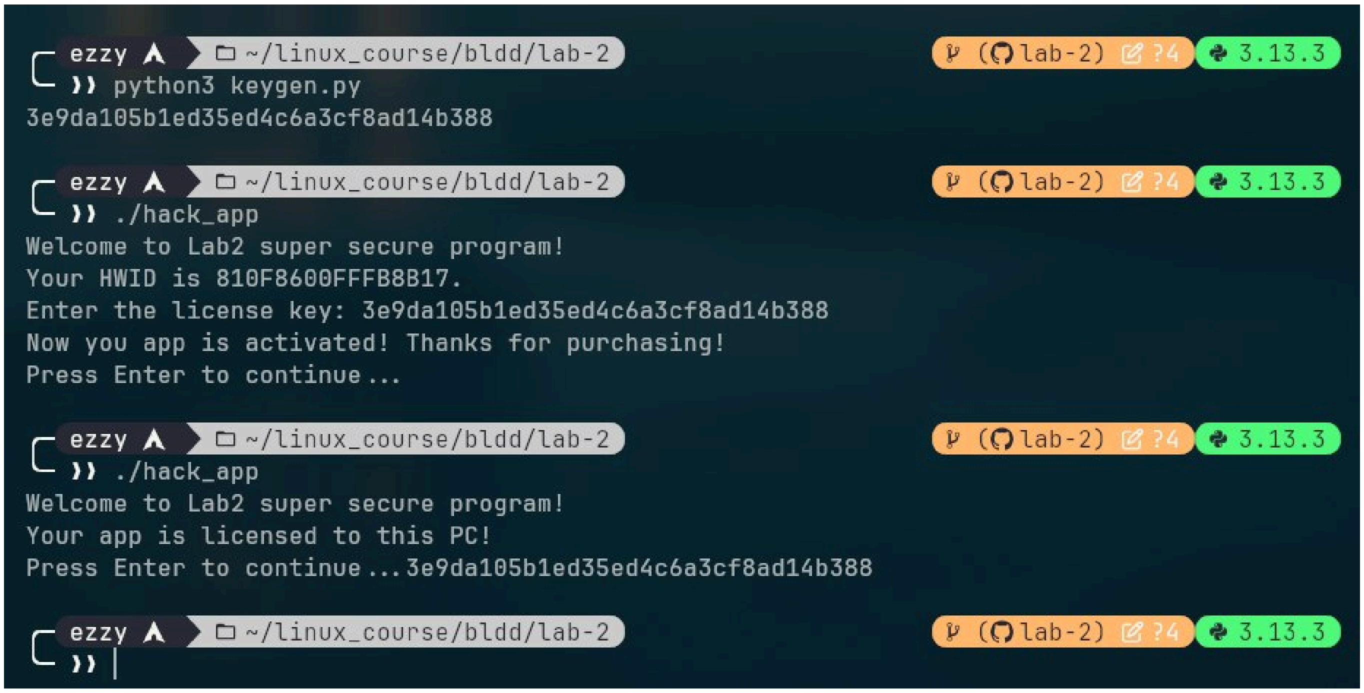
| Step | Extracted from code |
|---|---|
| 1 | `__get_cpuid(1, …)` returns `EAX`, `EDX` |
| 2 | Byte-swap both registers → `swap32` |
| 3 | Format: `"%08X%08X"` ⇒ **HWID** (16 upper-case hex) |
| 4 | `MD5(HWID_ASCII)` |
| 5 | Reverse order of the 16 MD5 bytes |
| 6 | Output lowercase hex ⇒ **license key** (32 chars) |

## 2 — Implementation ( `keygen.py` )

```
#!/usr/bin/env python3
import argparse, hashlib, re, subprocess, sys

def swap32(v): return ((v<<24)&0xFF000000)|((v<<8)&0x00FF0000)|
((v>>8)&0x0000FF00)|((v>>24)&0xFF)

def cpuid():
    out = subprocess.check_output(["cpuid", "-r", "-l", "1"], text=True)
    eax = int(re.search(r"eax=0x([0-9a-f]{8})", out, re.I).group(1), 16)
    edx = int(re.search(r"edx=0x([0-9a-f]{8})", out, re.I).group(1), 16)
    return eax, edx

def hwid():
    eax, edx = cpuid()
    return f"{swap32(eax):08X}{swap32(edx):08X}"

def key(h): return "".join(f"{b:02x}" for b in
hashlib.md5(h.encode()).digest()[::-1])

if __name__ == "__main__":
    a = argparse.ArgumentParser()
    a.add_argument("--id", help="override HWID (16 hex)")
    args = a.parse_args()
    hw = args.id or hwid()
    if not re.fullmatch(r"[0-9A-F]{16}", hw): sys.exit("HWID missing; use --
id")
    print(key(hw))
```

*No comments, < 100 LOC, single dependency — `cpuid` CLI.*

## 3 — Usage & result



```
ezzy ~/linux_course/bldd/lab-2                    (lab-2) ?4  3.13.3
)) python3 keygen.py
3e9da105b1ed35ed4c6a3cf8ad14b388

ezzy ~/linux_course/bldd/lab-2                    (lab-2) ?4  3.13.3
)) ./hack_app
Welcome to Lab2 super secure program!
Your HWID is 810F8600FFFB8B17.
Enter the license key: 3e9da105b1ed35ed4c6a3cf8ad14b388
Now you app is activated! Thanks for purchasing!
Press Enter to continue...

ezzy ~/linux_course/bldd/lab-2                    (lab-2) ?4  3.13.3
)) ./hack_app
Welcome to Lab2 super secure program!
Your app is licensed to this PC!
Press Enter to continue...3e9da105b1ed35ed4c6a3cf8ad14b388

ezzy ~/linux_course/bldd/lab-2                    (lab-2) ?4  3.13.3
))
```

# Binary-patch

## 1 — Locate the check

```
objdump -d hack_app | grep -n -A3 -B1 strncmp
103-
104:0000000000001150 <strncmp@plt>:
105-    1150:  f3 0f 1e fa             endbr64
106:    1154:  f2 ff 25 25 2e 00 00    bnd jmp *0x2e25(%rip)        # 3f80
<strncmp@GLIBC_2.2.5>
107-    115b:  0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)
108-
109-0000000000001160 <system@plt>:
--
420-    1590:  48 8d 3d a9 3a 00 00    lea    0x3aa9(%rip),%rdi        # 5040
<md5decode>
421:    1597:  e8 b4 fb ff ff          call   1150 <strncmp@plt>
422-    159c:  85 c0                   test   %eax,%eax
423-    159e:  75 07                   jne    15a7 <main+0x1c3>
424-    15a0:  c7 45 e4 01 00 00 00    movl   $0x1,-0x1c(%rbp)
--
439-    15ee:  48 8d 3d 4b 3a 00 00    lea    0x3a4b(%rip),%rdi        # 5040
<md5decode>
440:    15f5:  e8 56 fb ff ff          call   1150 <strncmp@plt>
441-    15fa:  85 c0                   test   %eax,%eax
442-    15fc:  75 33                   jne    1631 <main+0x24d>
443-    15fe:  41 b8 00 00 00 00       mov    $0x0,%r8d
```

```
1597:  call   strncmp@plt        ; compare md5decode vs xattr
159c:  test   eax,eax
159e:  jne    15a7               ; if not equal → ask for key
```

- Virtual address of the conditional jump: **0x159e**
- File offset (ELF base is 0): **0x159e**
- Bytes: `75 07` ( `JNE +0x07` )

---

## 2 — Patch decision

Replace `JNE` with two `NOP` s ⇒ `test eax,eax` is preserved, but branch is neutralised → execution falls through as if comparison passed.

`75 07` -> `90 90`

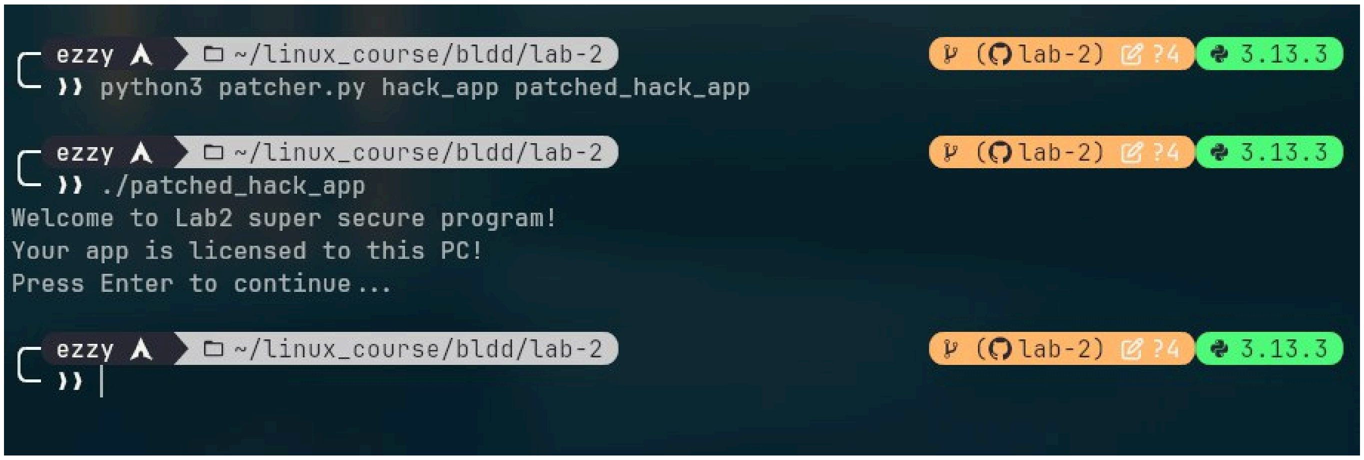
---

## 3 — Patcher script

```python
#!/usr/bin/env python3
import argparse, os, stat, sys

OFF, ORIG, PATCH = 0x159e, b"\x75\x07", b"\x90\x90"

def patch(src, dst):
    data = bytearray(open(src, "rb").read())
    if data[OFF:OFF+2] != ORIG:
        sys.exit("unexpected opcode")
    data[OFF:OFF+2] = PATCH
    open(dst, "wb").write(data)
    os.chmod(dst, os.stat(src).st_mode | stat.S_IXUSR)

if __name__ == "__main__":
    a = argparse.ArgumentParser()
    a.add_argument("infile"), a.add_argument("outfile")
    args = a.parse_args()
    patch(args.infile, args.outfile)
```

---

## 4 — Usage & result



```
╭ ezzy  ~/linux_course/bldd/lab-2                         (lab-2) ?4  3.13.3
╰ ❯❯ python3 patcher.py hack_app patched_hack_app

╭ ezzy  ~/linux_course/bldd/lab-2                         (lab-2) ?4  3.13.3
╰ ❯❯ ./patched_hack_app
Welcome to Lab2 super secure program!
Your app is licensed to this PC!
Press Enter to continue...

╭ ezzy  ~/linux_course/bldd/lab-2                         (lab-2) ?4  3.13.3
╰ ❯❯ |
```

Program starts licensed; no key, no xattr — licensing logic disabled.