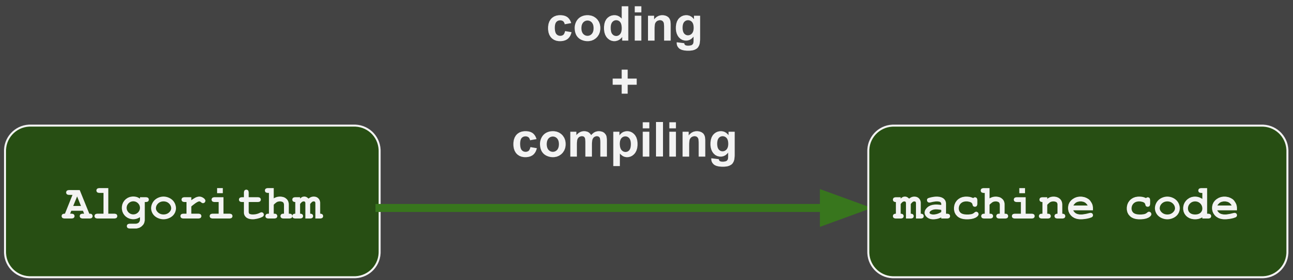
Introduction to 8086 Assembly

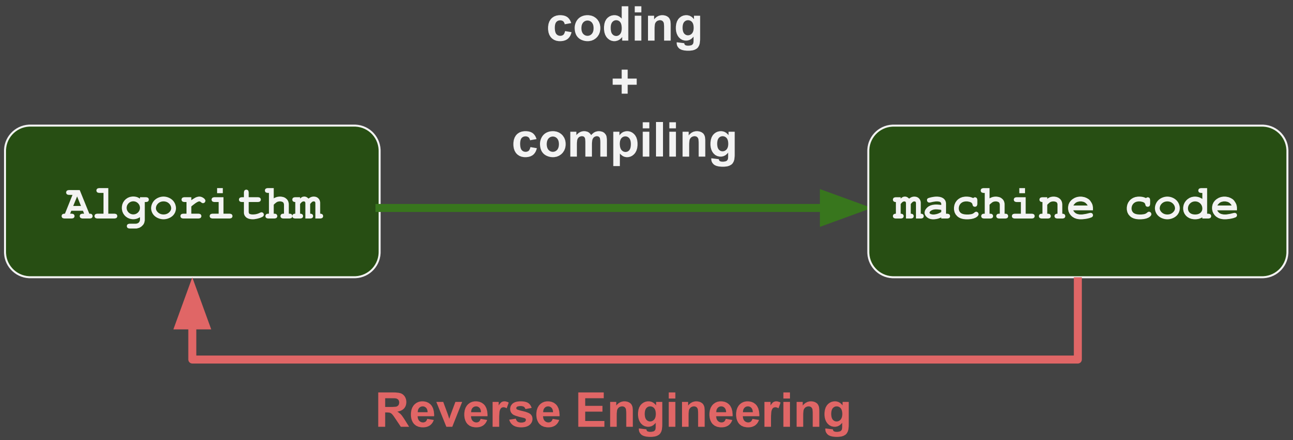
Lecture RevEng

Introduction to Reverse Engineering

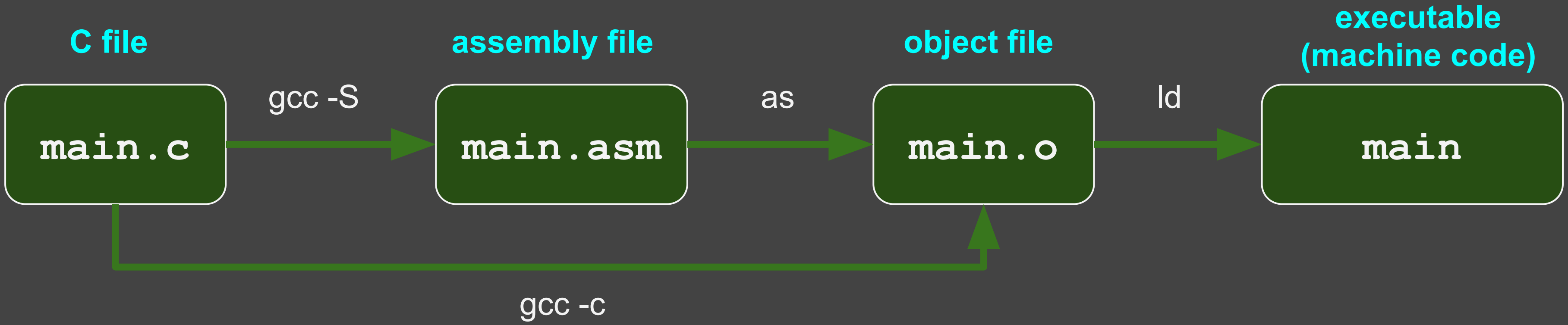
Reverse Engineering



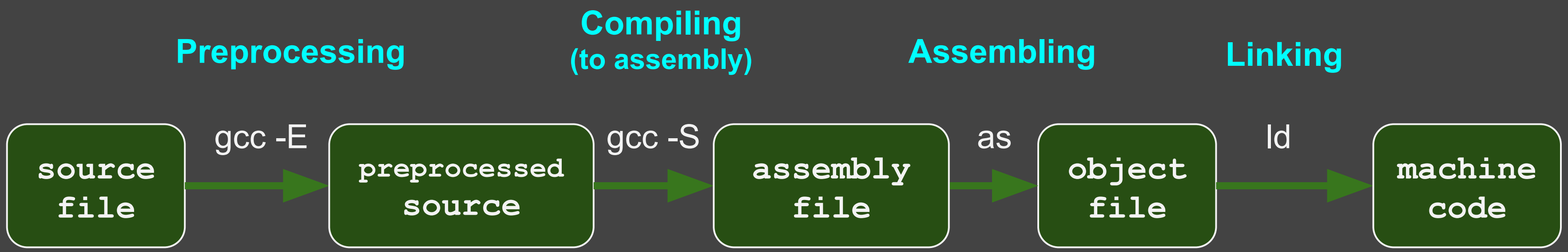
Reverse Engineering



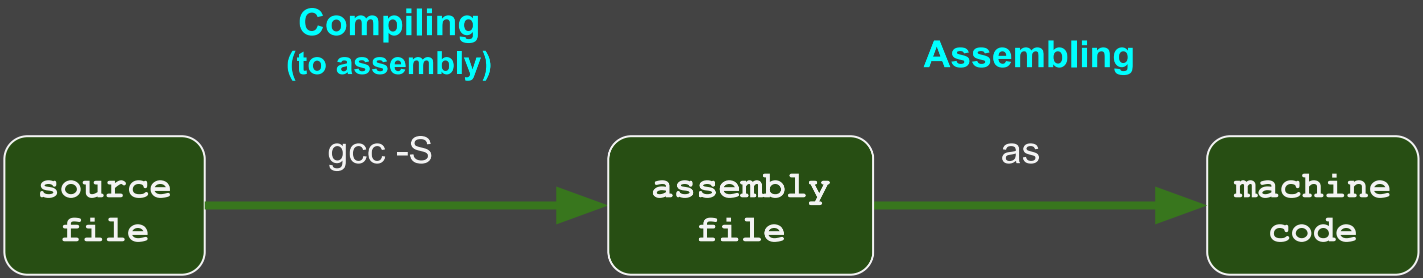
Remember: high-level to low-level hierarchy



Remember: high-level to low-level hierarchy



Remember: high-level to low-level hierarchy



Why learn Reverse Engineering?

- ?

Why learn Reverse Engineering?

- Modify software, add features to closed-source software (legal?)
- Find and/or fix bugs in closed-source software
- better understand important concepts
- Assess software security, identify vulnerabilities
- Learn how to protect your software against reversing
- Understand/detect/fix malware (viruses, worms, trojans, spyware, adware, etc.)
    - get a job in an antivirus company
-

Why learn Reverse Engineering?

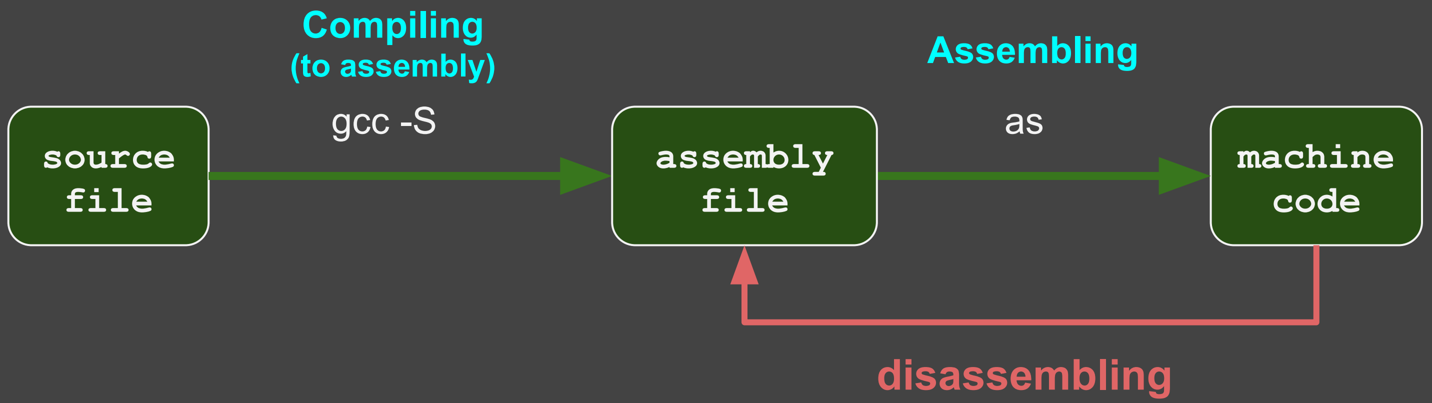
- Modify software, add features to closed-source software (legal?)
- Find and/or fix bugs in closed-source software
- better understand important concepts
- Assess software security, identify vulnerabilities
- Learn how to protect your software against reversing
- Understand/detect/fix malware (viruses, worms, trojans, spyware, adware, etc.)
    - get a job in an antivirus company
- It's fun!

Basic Tools

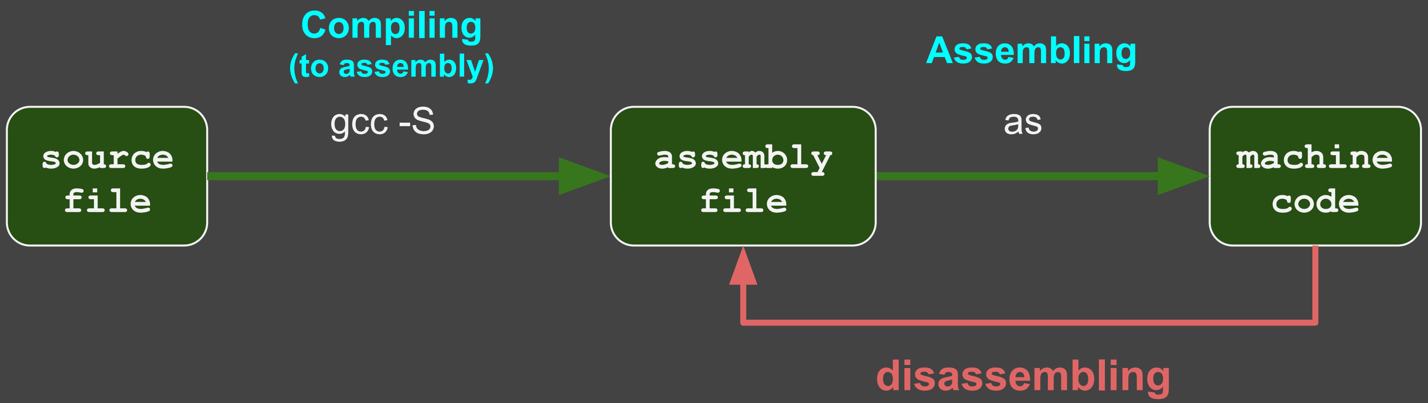
- Disassembler
- Debugger (e.g. GDB)
- Hex Editor

Disassemblers

K. N. Toosi
University of Technology

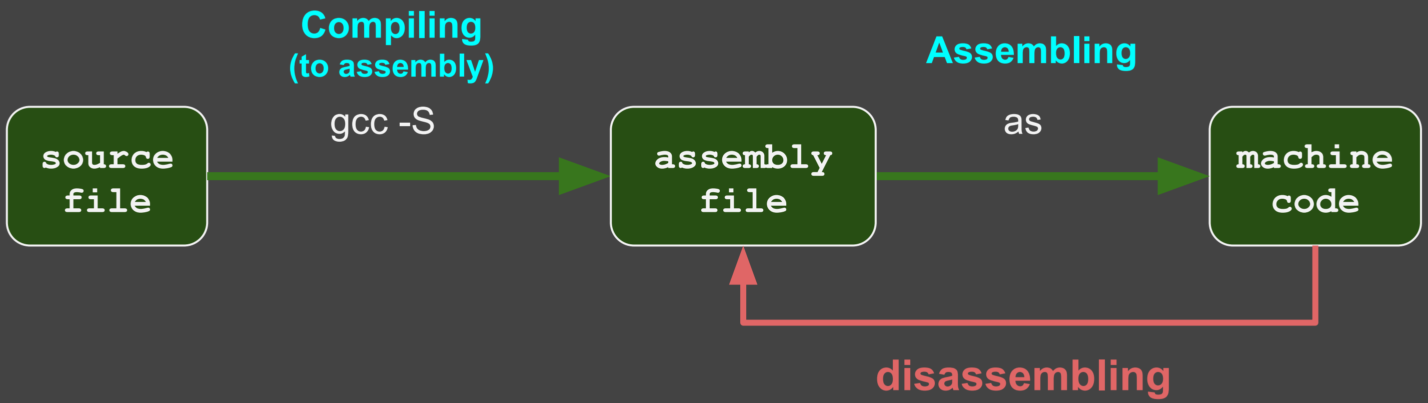


Disassemblers



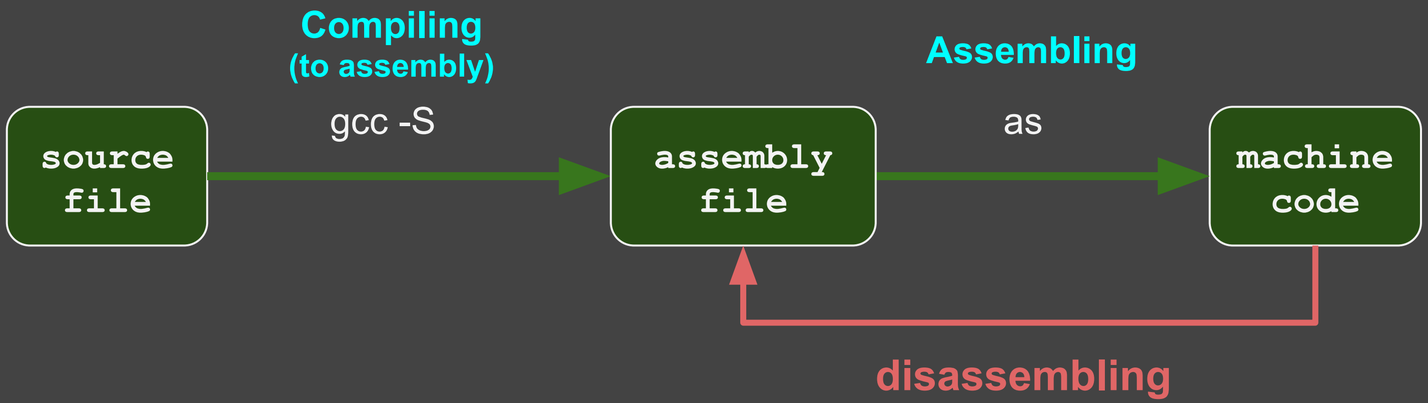
- One-to-one correspondence between assembly and machine code (almost)
-

Disassemblers



- One-to-one correspondence between assembly and machine code (almost)
- Distinguish code from data
    - Data may reside in code section, Code may get stored in data section
    - code crawling
-

Disassemblers



- One-to-one correspondence between assembly and machine code (almost)
- Distinguish code from data
  - Data may reside in code section, Code may get stored in data section
  - code crawling
- Usually, debuggers can also disassemble
- https://en.wikibooks.org/wiki/X86_Disassembly/Disassemblers_and_Decompilers

Debuggers

- Execute, test, debug, trace
- High-level language vs. low-level/machine language debugging
- step-by-step running
- breaking, break points
- Interface (GUI vs command line)
- Observe
  - variable values, expressions (high-level debugging)
  - memory contents
  - register values (processor state)

Debuggers

K. N. Toosi
University of Technology

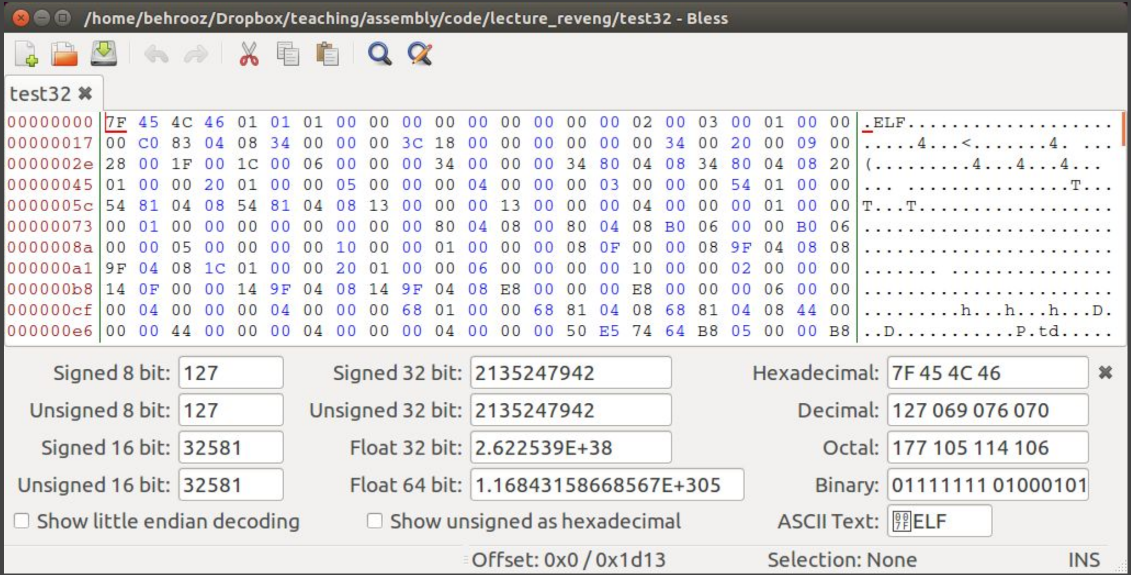
- Many debuggers can also
  - **disassemble**
  - **modify** code while running
  - **skip** code

Debuggers

- Many debuggers can also
  - **disassemble**
  - **modify** code while running
  - **skip** code
- Example
  - The GNU Debugger (GDB)
  - DBX
  - LLDB
  - Microsoft Visual Studio Debugger

HEX Editors

- View/Edit binary files
- Modify executable files (patching)



/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/test32 - Bless

test32

```
00000000 7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00 02 00 03 00 01 00 00 .ELF....................
00000017 00 C0 83 04 08 34 00 00 00 3C 18 00 00 00 00 00 00 34 00 20 00 09 00 .....4...<.......4. ...
0000002e 28 00 1F 00 1C 00 06 00 00 00 34 00 00 00 34 80 04 08 34 80 04 08 20 (.........4...4...4...
00000045 01 00 00 20 01 00 00 05 00 00 00 04 00 00 00 03 00 00 00 54 01 00 00 ... ...............T...
0000005c 54 81 04 08 54 81 04 08 13 00 00 00 13 00 00 00 04 00 00 00 01 00 00 T...T..................
00000073 00 01 00 00 00 00 00 00 00 00 80 04 08 00 80 04 08 B0 06 00 00 B0 06 .......................
0000008a 00 00 05 00 00 00 00 10 00 00 01 00 00 00 08 0F 00 00 08 9F 04 08 08 .......................
000000a1 9F 04 08 1C 01 00 00 20 01 00 00 06 00 00 00 00 10 00 00 02 00 00 00 ....... ...............
000000b8 14 0F 00 00 14 9F 04 08 14 9F 04 08 E8 00 00 00 E8 00 00 00 06 00 00 .......................
000000cf 00 04 00 00 00 04 00 00 00 68 01 00 00 68 81 04 08 68 81 04 08 44 00 .........h...h...h...D.
000000e6 00 00 44 00 00 00 04 00 00 00 04 00 00 00 50 E5 74 64 B8 05 00 00 B8 ..D...........P.td.....
```

Signed 8 bit: 127
Unsigned 8 bit: 127
Signed 16 bit: 32581
Unsigned 16 bit: 32581
Signed 32 bit: 2135247942
Unsigned 32 bit: 2135247942
Float 32 bit: 2.622539E+38
Float 64 bit: 1.16843158668567E+305
Hexadecimal: 7F 45 4C 46
Decimal: 127 069 076 070
Octal: 177 105 114 106
Binary: 01111111 01000101
ASCII Text: ELF

☐ Show little endian decoding
☐ Show unsigned as hexadecimal

Offset: 0x0 / 0x1d13   Selection: None   INS

Getting started

K. N. Toosi
University of Technology

checkpass1.c

```
int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

```
char password[] = "?????????????";

int check_password(char *input) {
  return strcmp(input,password)== 0;
}
```

Getting started

checkpass1.c

```
int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

I hide this for now!

```
char password[] = "?????????????";

int check_password(char *input) {
  return strcmp(input,password)== 0;
}
```

Getting started

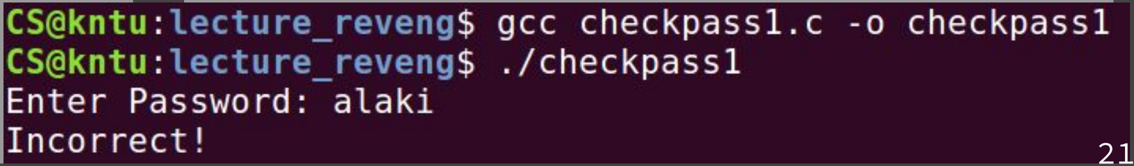
checkpass1.c

```c
int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

I hide this for now!

```c
char password[] = "??????????????";

int check_password(char *input) {
  return strcmp(input,password)== 0;
}
```



```
CS@kntu:lecture_reveng$ gcc checkpass1.c -o checkpass1
CS@kntu:lecture_reveng$ ./checkpass1
Enter Password: alaki
Incorrect!
```

Step 1: Collect info about target program

- Hardware platform
- OS
- architecture (16-, 32- or 64- bit)
- system calls
- library calls
- compiler (+version)
- meta data
  - debug info.
  - labels/variables (stripped?)
- opened files
- opened sockets / network connections

Getting started

```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

```
$ gcc -m64    checkpass1.c -o checkpass64

$ gcc -m64 -s checkpass1.c -o checkpass64s


$ gcc -m64 -g checkpass1.c -o checkpass64g
```

Getting started

K. N. Toosi
University of Technology

```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

```
$ gcc -m64    checkpass1.c -o checkpass64

$ gcc -m64 -s checkpass1.c -o checkpass64s
```

strip symbols

```
$ gcc -m64 -g checkpass1.c -o checkpass64g
```

add debug info

Getting started

```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

```
$ gcc -m64    checkpass1.c -o checkpass64

$ gcc -m64 -s checkpass1.c -o checkpass64s
```

strip symbols

```
$ gcc -m64 -g checkpass1.c -o checkpass64g
```

add debug info

```
# create 32-bit executables
$ gcc -m32    checkpass1.c -o checkpass32

$ gcc -m32 -s checkpass1.c -o checkpass32s

$ gcc -m32 -g checkpass1.c -o checkpass32g
```

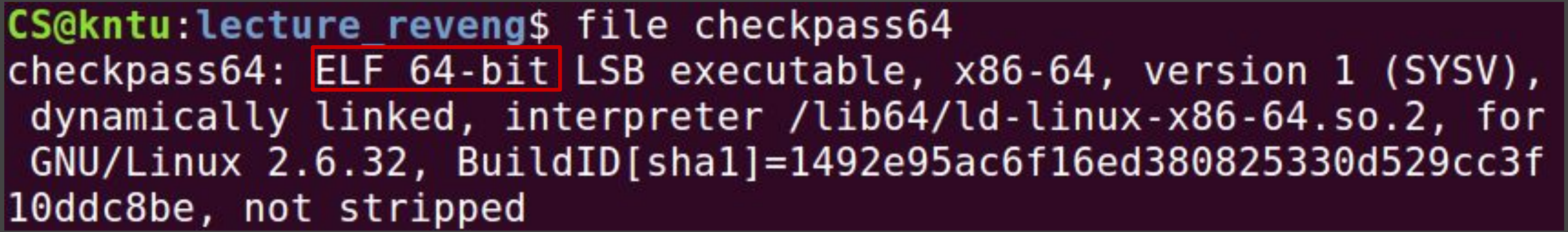
strip

- remove the symbols (**symbol table**) from object files/executables
    - code labels, function names
    - data labels, global variables
- `gcc -s`
- also a UNIX command
    - `$ strip checkpass32`

add debug info

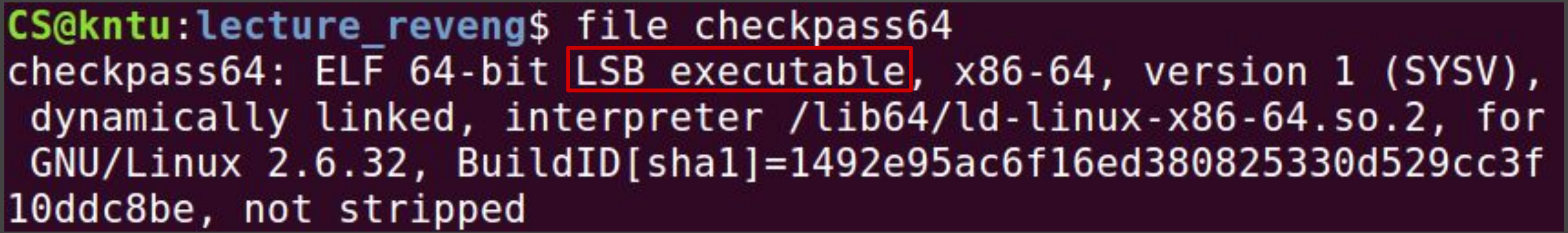
- add debug info to the object file/executable
  - line number maps
  - global variable names
  - local variable names
  - etc.
- `gcc -g`

Collect info: the `file` command



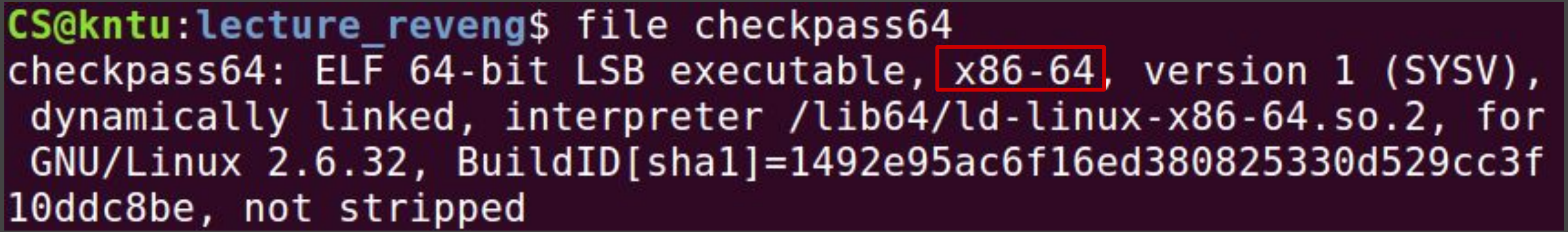
```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```

Collect info: the `file` command



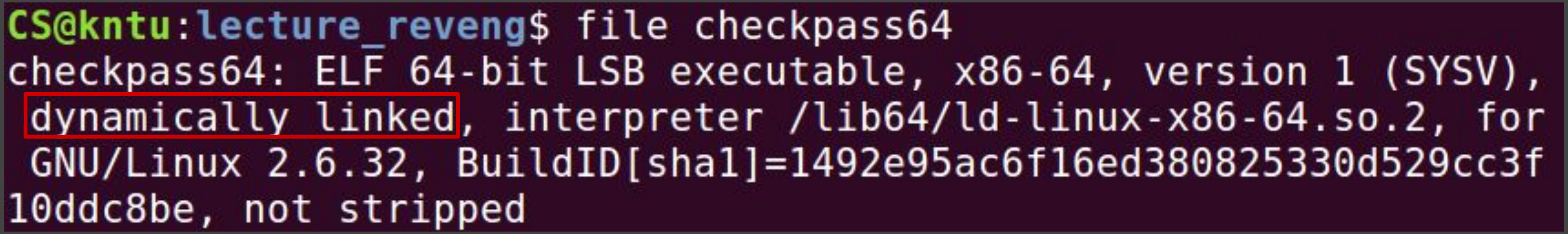
```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```

Collect info: the `file` command



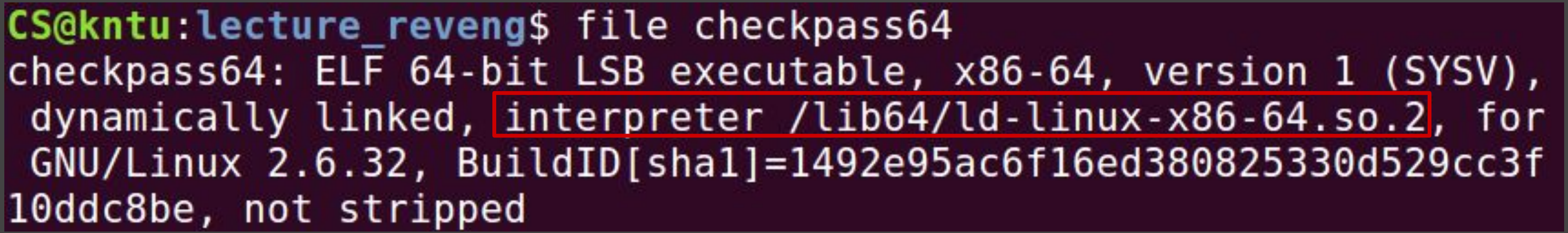
```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```

Collect info: the `file` command



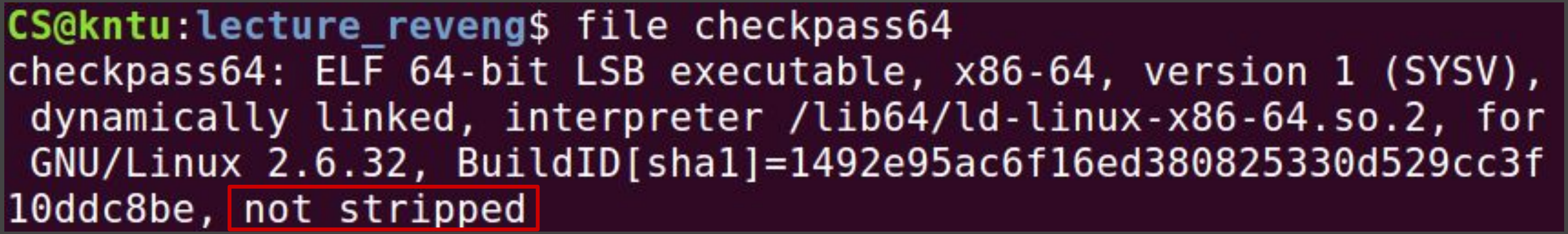
```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```

Collect info: the `file` command



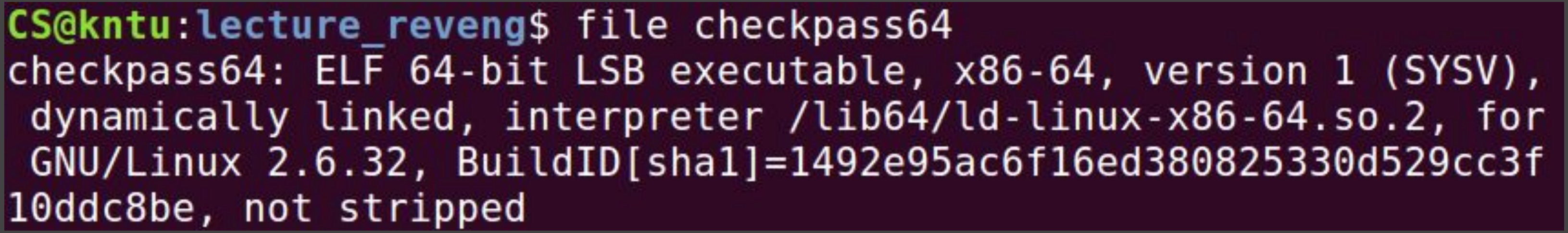
```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```

Collect info: the `file` command

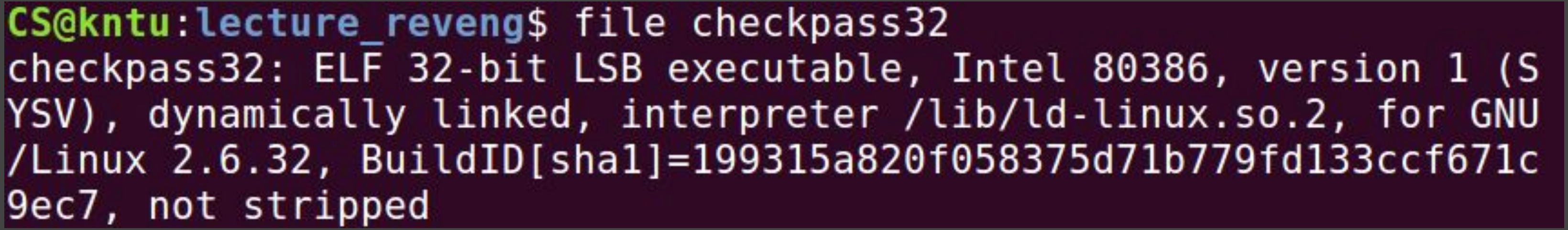


```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```

Collect info: the `file` command

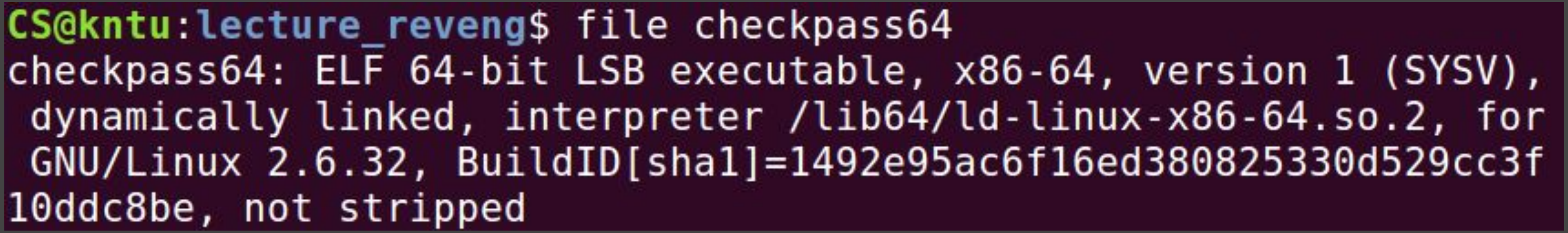


```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```

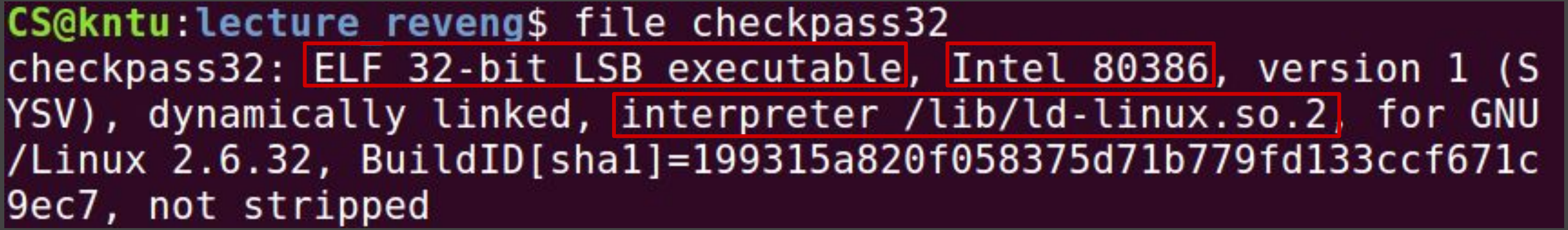


```
CS@kntu:lecture_reveng$ file checkpass32
checkpass32: ELF 32-bit LSB executable, Intel 80386, version 1 (S
YSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU
/Linux 2.6.32, BuildID[sha1]=199315a820f058375d71b779fd133ccf671c
9ec7, not stripped
```

Collect info: the `file` command

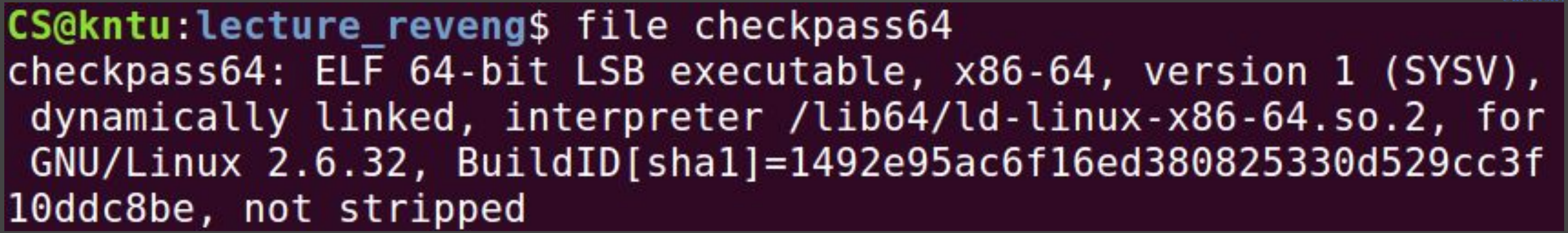


```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```

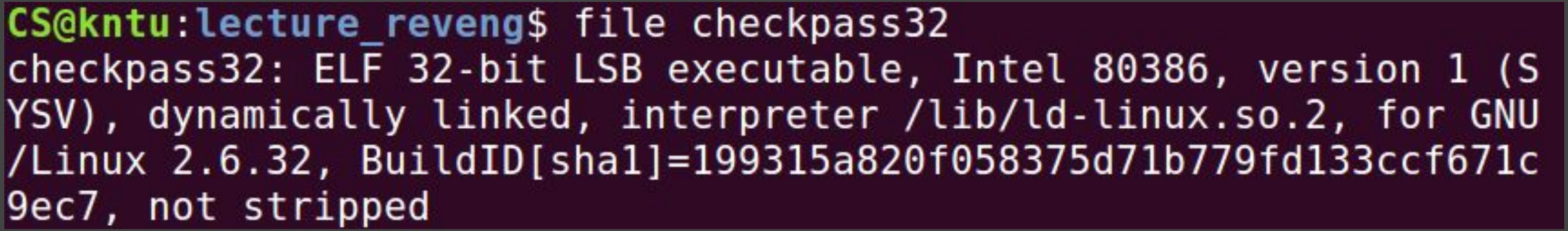


```
CS@kntu:lecture_reveng$ file checkpass32
checkpass32: ELF 32-bit LSB executable, Intel 80386, version 1 (S
YSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU
/Linux 2.6.32, BuildID[sha1]=199315a820f058375d71b779fd133ccf671c
9ec7, not stripped
```

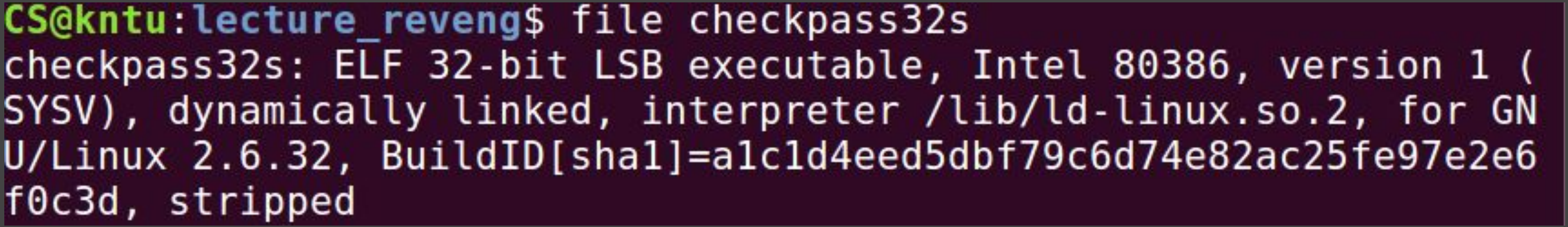
Collect info: the `file` command



```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```

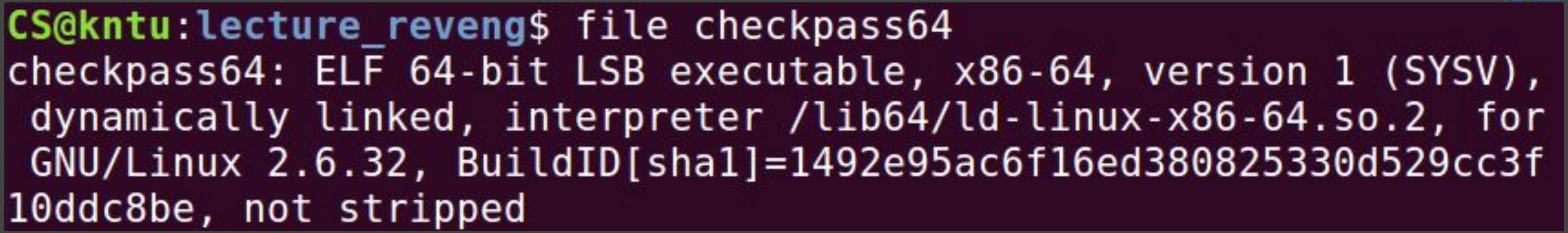


```
CS@kntu:lecture_reveng$ file checkpass32
checkpass32: ELF 32-bit LSB executable, Intel 80386, version 1 (S
YSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU
/Linux 2.6.32, BuildID[sha1]=199315a820f058375d71b779fd133ccf671c
9ec7, not stripped
```

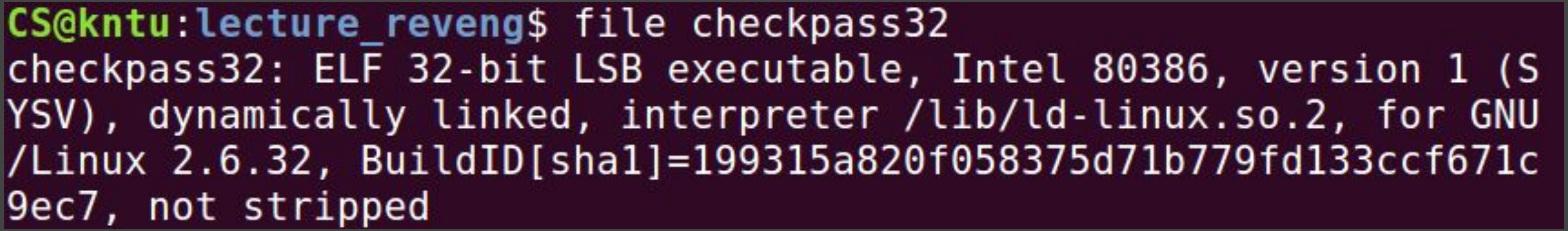


```
CS@kntu:lecture_reveng$ file checkpass32s
checkpass32s: ELF 32-bit LSB executable, Intel 80386, version 1 (
SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GN
U/Linux 2.6.32, BuildID[sha1]=a1c1d4eed5dbf79c6d74e82ac25fe97e2e6
f0c3d, stripped
```

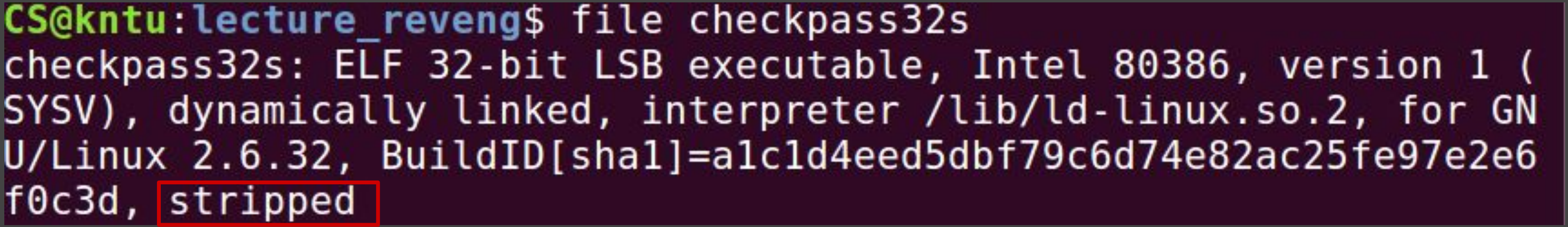
Collect info: the `file` command



```
CS@kntu:lecture_reveng$ file checkpass64
checkpass64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
 dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=1492e95ac6f16ed380825330d529cc3f
10ddc8be, not stripped
```



```
CS@kntu:lecture_reveng$ file checkpass32
checkpass32: ELF 32-bit LSB executable, Intel 80386, version 1 (S
YSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU
/Linux 2.6.32, BuildID[sha1]=199315a820f058375d71b779fd133ccf671c
9ec7, not stripped
```



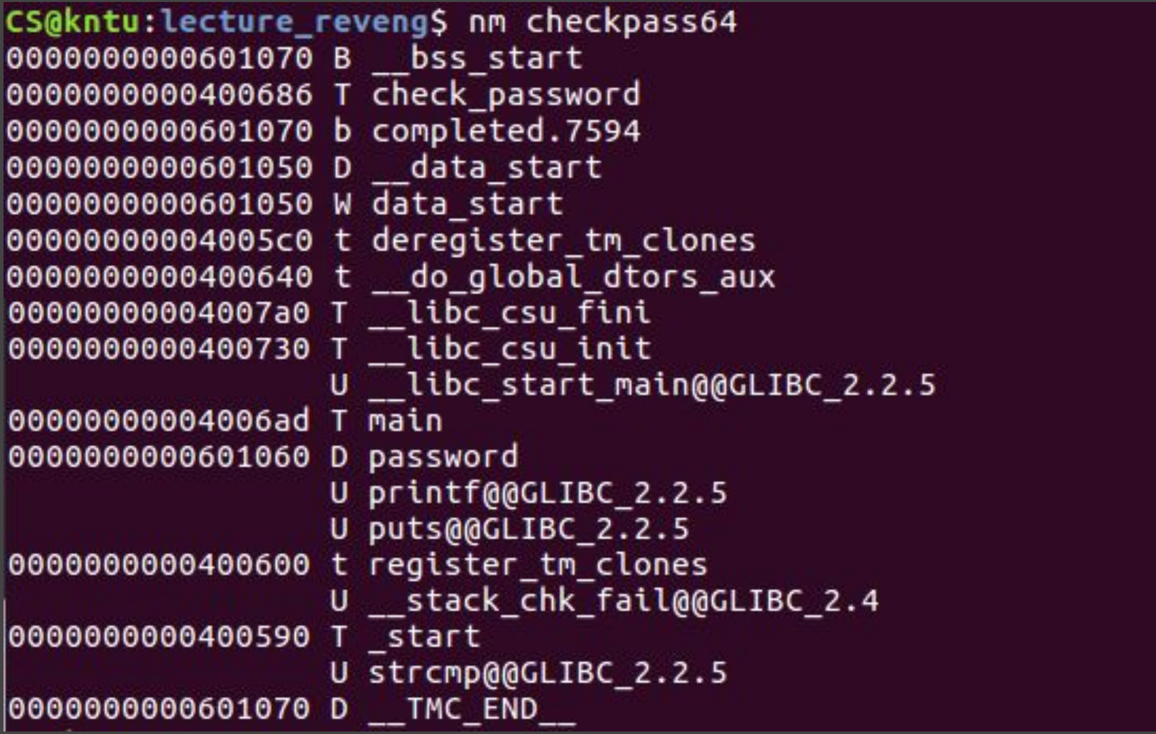
```
CS@kntu:lecture_reveng$ file checkpass32s
checkpass32s: ELF 32-bit LSB executable, Intel 80386, version 1 (
SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GN
U/Linux 2.6.32, BuildID[sha1]=a1c1d4eed5dbf79c6d74e82ac25fe97e2e6
f0c3d, stripped
```

Collect info: symbols, the nm command

K. N. Toosi
University of Technology

```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

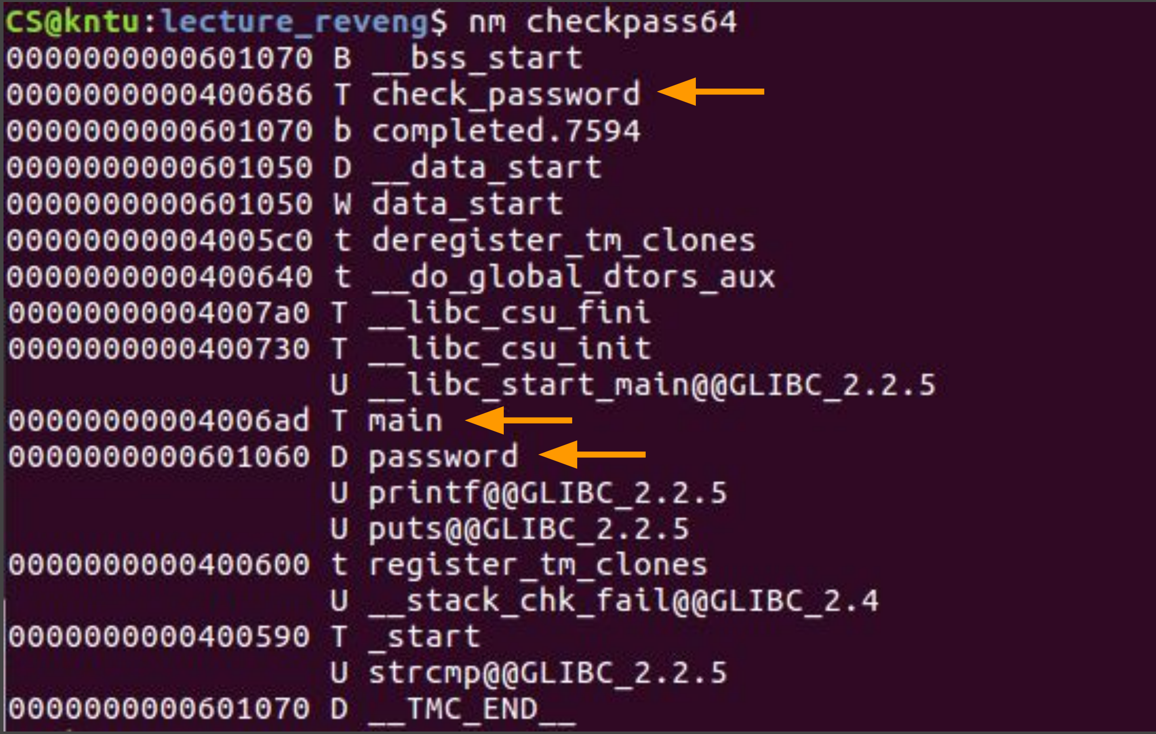


```
CS@kntu:lecture_reveng$ nm checkpass64
0000000000601070 B __bss_start
0000000000400686 T check_password
0000000000601070 b completed.7594
0000000000601050 D __data_start
0000000000601050 W data_start
00000000004005c0 t deregister_tm_clones
0000000000400640 t __do_global_dtors_aux
00000000004007a0 T __libc_csu_fini
0000000000400730 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
00000000004006ad T main
0000000000601060 D password
                 U printf@@GLIBC_2.2.5
                 U puts@@GLIBC_2.2.5
0000000000400600 t register_tm_clones
                 U __stack_chk_fail@@GLIBC_2.4
0000000000400590 T _start
                 U strcmp@@GLIBC_2.2.5
0000000000601070 D __TMC_END__
```

Collect info: symbols, the nm command

```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

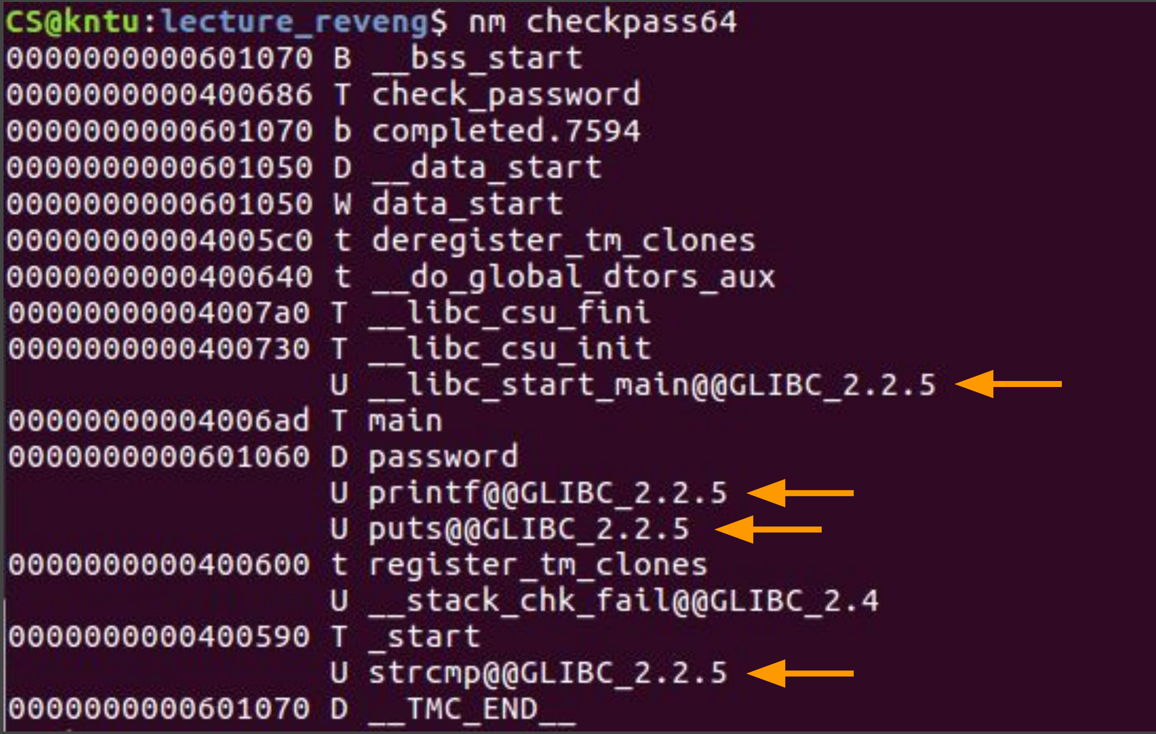


```
CS@kntu:lecture_reveng$ nm checkpass64
0000000000601070 B __bss_start
0000000000400686 T check_password
0000000000601070 b completed.7594
0000000000601050 D __data_start
0000000000601050 W data_start
00000000004005c0 t deregister_tm_clones
0000000000400640 t __do_global_dtors_aux
00000000004007a0 T __libc_csu_fini
0000000000400730 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
00000000004006ad T main
0000000000601060 D password
                 U printf@@GLIBC_2.2.5
                 U puts@@GLIBC_2.2.5
0000000000400600 t register_tm_clones
                 U __stack_chk_fail@@GLIBC_2.4
0000000000400590 T _start
                 U strcmp@@GLIBC_2.2.5
0000000000601070 D __TMC_END__
```

Collect info: symbols, the nm command

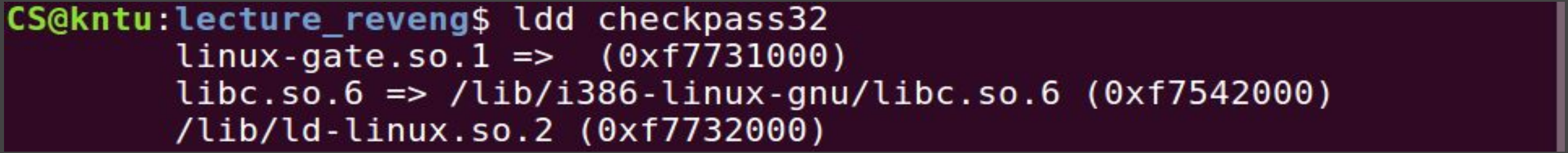
```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c



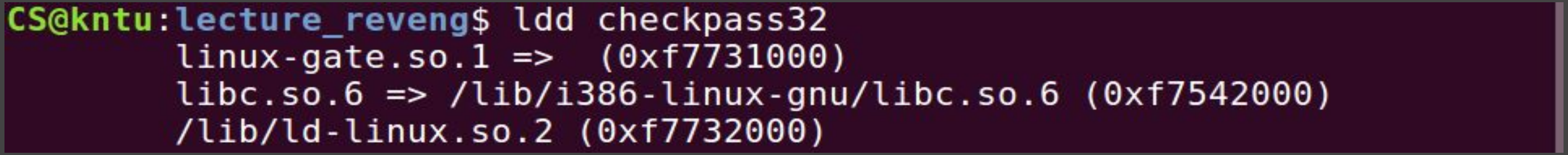
```
CS@kntu:lecture_reveng$ nm checkpass64
0000000000601070 B __bss_start
0000000000400686 T check_password
0000000000601070 b completed.7594
0000000000601050 D __data_start
0000000000601050 W data_start
00000000004005c0 t deregister_tm_clones
0000000000400640 t __do_global_dtors_aux
00000000004007a0 T __libc_csu_fini
0000000000400730 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
00000000004006ad T main
0000000000601060 D password
                 U printf@@GLIBC_2.2.5
                 U puts@@GLIBC_2.2.5
0000000000400600 t register_tm_clones
                 U __stack_chk_fail@@GLIBC_2.4
0000000000400590 T _start
                 U strcmp@@GLIBC_2.2.5
0000000000601070 D __TMC_END__
```

Collect info: shared object dependencies

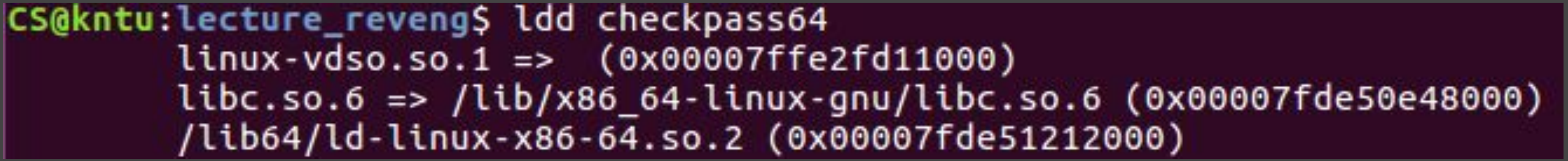


```
CS@kntu:lecture_reveng$ ldd checkpass32
        linux-gate.so.1 =>  (0xf7731000)
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xf7542000)
        /lib/ld-linux.so.2 (0xf7732000)
```

Collect info: shared object dependencies

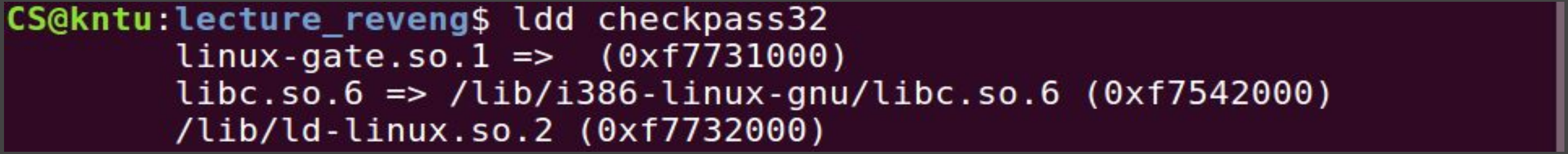


```
CS@kntu:lecture_reveng$ ldd checkpass32
        linux-gate.so.1 =>  (0xf7731000)
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xf7542000)
        /lib/ld-linux.so.2 (0xf7732000)
```

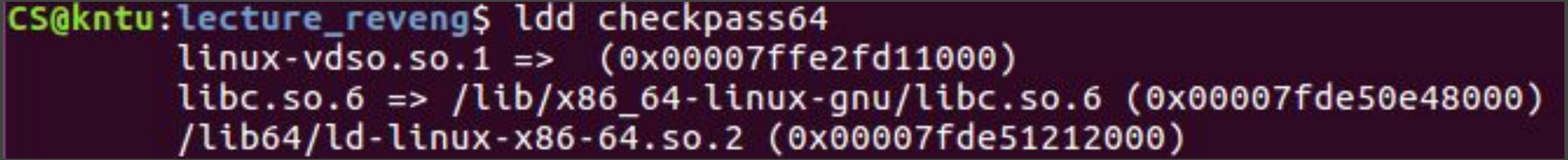


```
CS@kntu:lecture_reveng$ ldd checkpass64
        linux-vdso.so.1 =>  (0x00007ffe2fd11000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fde50e48000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fde51212000)
```

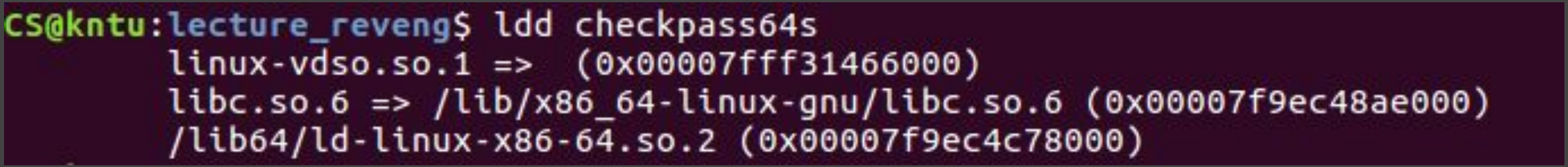
Collect info: shared object dependencies



```
CS@kntu:lecture_reveng$ ldd checkpass32
        linux-gate.so.1 =>  (0xf7731000)
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xf7542000)
        /lib/ld-linux.so.2 (0xf7732000)
```



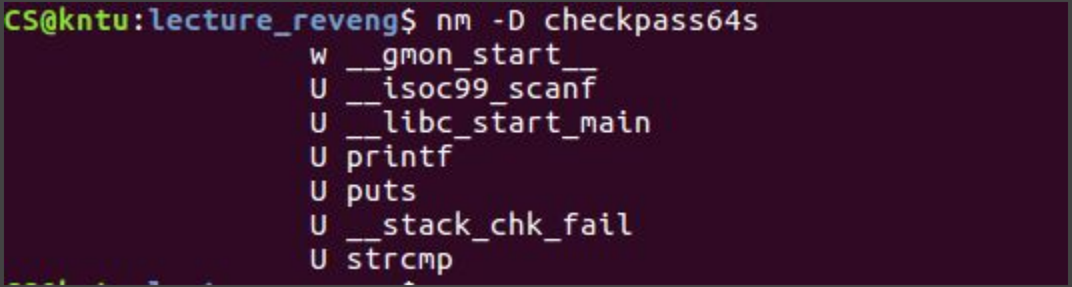
```
CS@kntu:lecture_reveng$ ldd checkpass64
        linux-vdso.so.1 =>  (0x00007ffe2fd11000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fde50e48000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fde51212000)
```



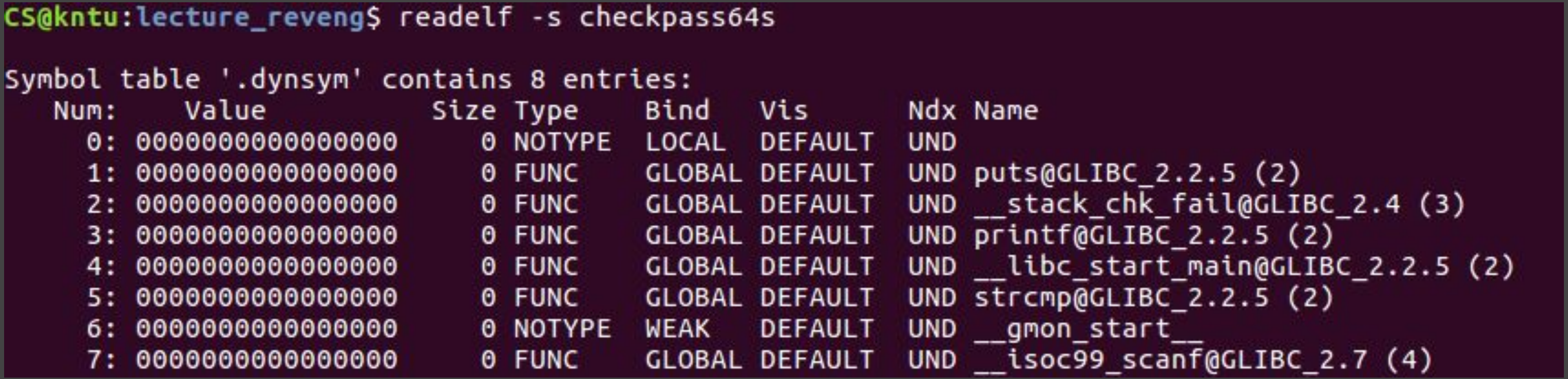
```
CS@kntu:lecture_reveng$ ldd checkpass64s
        linux-vdso.so.1 =>  (0x00007fff31466000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f9ec48ae000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f9ec4c78000)
```

Collect info: library functions

K. N. Toosi
University of Technology

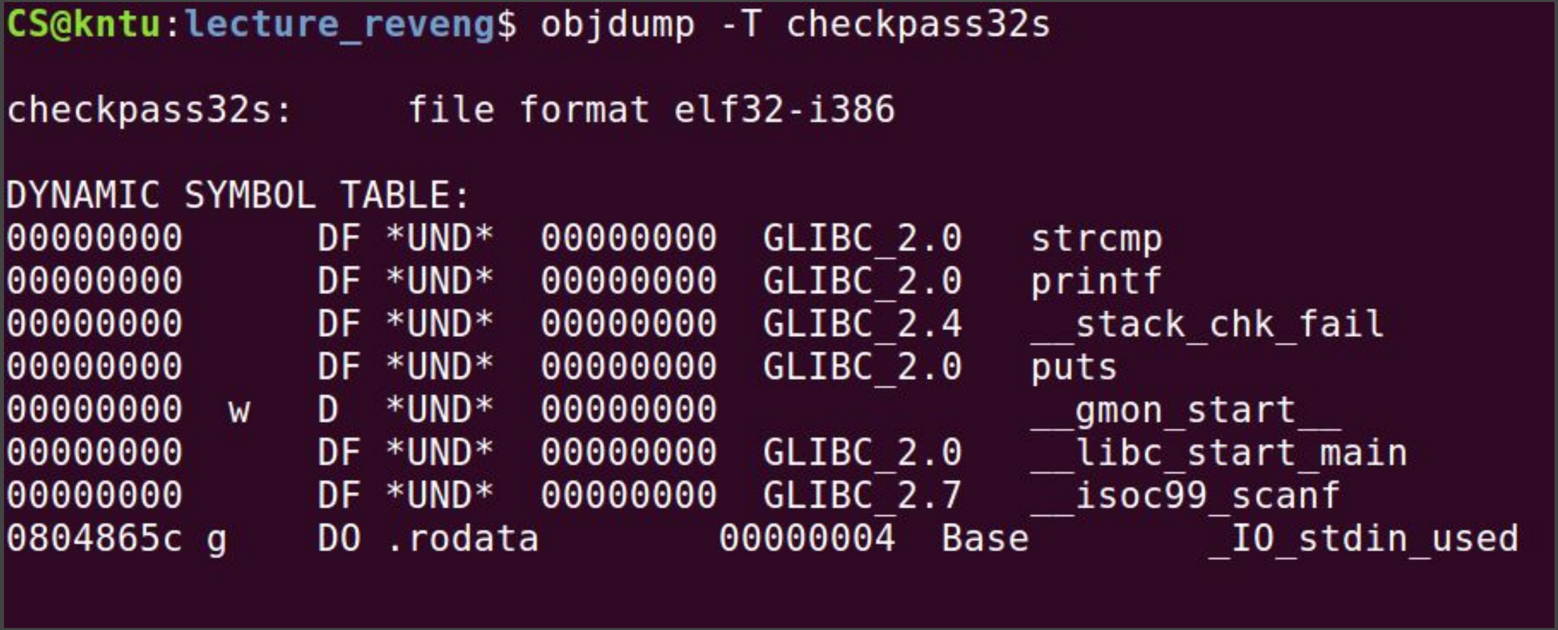


```
CS@kntu:lecture_reveng$ nm -D checkpass64s
                 w __gmon_start__
                 U __isoc99_scanf
                 U __libc_start_main
                 U printf
                 U puts
                 U __stack_chk_fail
                 U strcmp
```



```
CS@kntu:lecture_reveng$ readelf -s checkpass64s

Symbol table '.dynsym' contains 8 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND 
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND puts@GLIBC_2.2.5 (2)
     2: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __stack_chk_fail@GLIBC_2.4 (3)
     3: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND printf@GLIBC_2.2.5 (2)
     4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.2.5 (2)
     5: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND strcmp@GLIBC_2.2.5 (2)
     6: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
     7: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __isoc99_scanf@GLIBC_2.7 (4)
```

Collect info: library functions



```
CS@kntu:lecture_reveng$ objdump -T checkpass32s

checkpass32s:     file format elf32-i386

DYNAMIC SYMBOL TABLE:
00000000      DF *UND*  00000000  GLIBC_2.0   strcmp
00000000      DF *UND*  00000000  GLIBC_2.0   printf
00000000      DF *UND*  00000000  GLIBC_2.4   __stack_chk_fail
00000000      DF *UND*  00000000  GLIBC_2.0   puts
00000000  w   D  *UND*  00000000              __gmon_start__
00000000      DF *UND*  00000000  GLIBC_2.0   __libc_start_main
00000000      DF *UND*  00000000  GLIBC_2.7   __isoc99_scanf
0804865c g    DO .rodata       00000004  Base        _IO_stdin_used
```

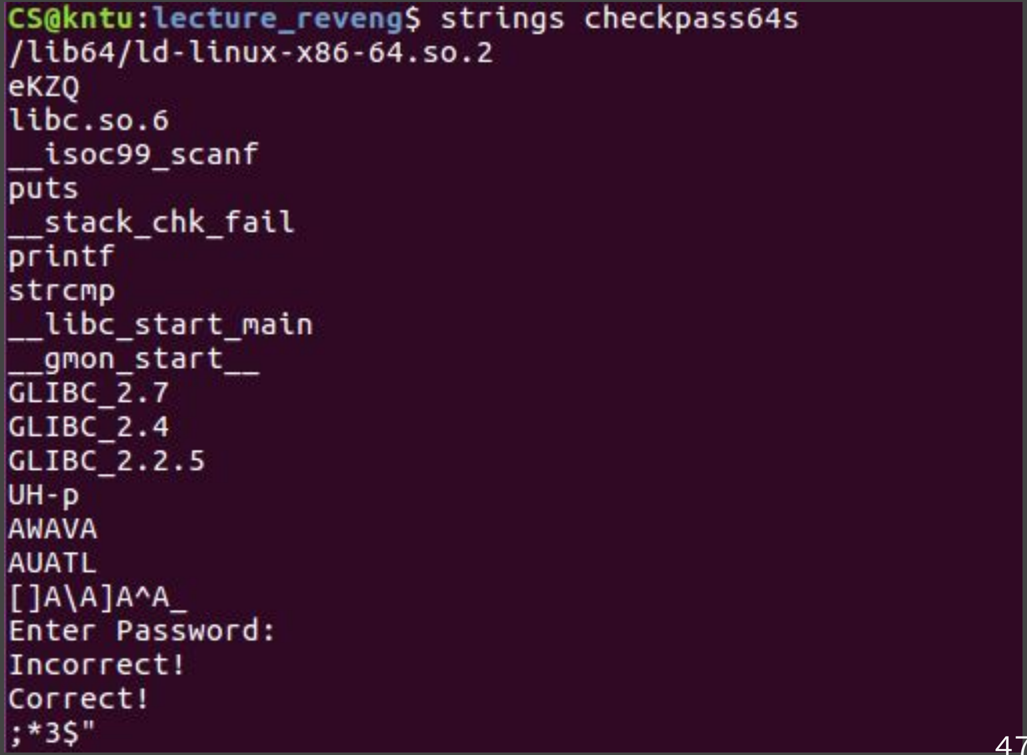
Collect info: the `strings` command

- prints sequences of printable ASCII characters
  - of length **`n`** or more (default **`n=4`**)

Collect info: the strings command

```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

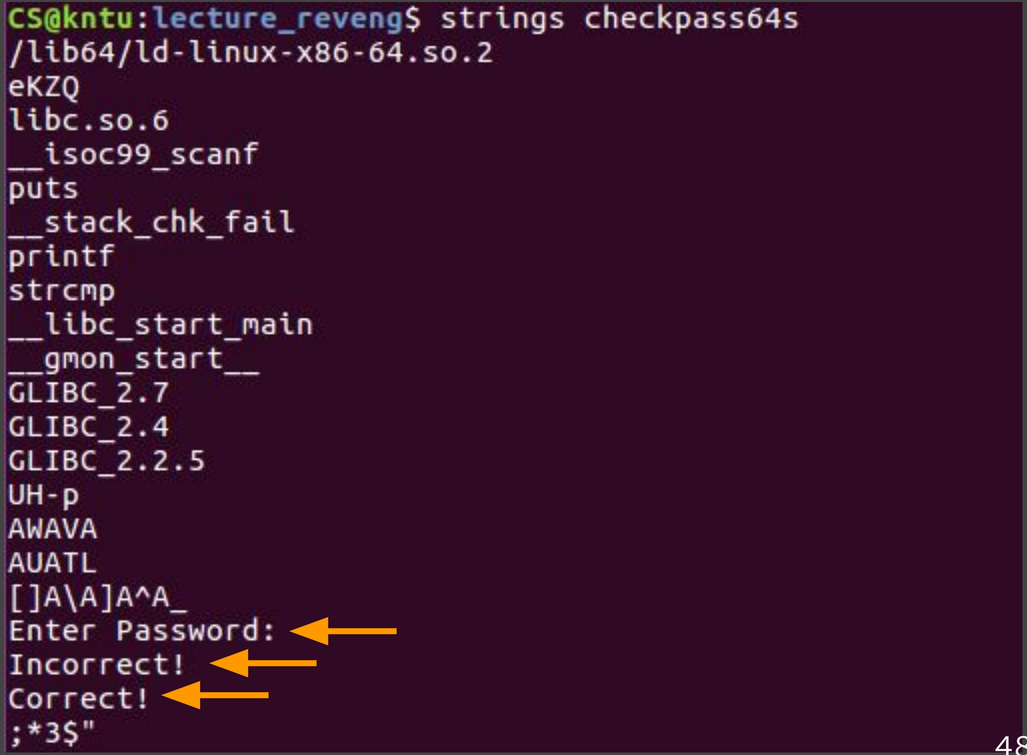


```
CS@kntu:lecture_reveng$ strings checkpass64s
/lib64/ld-linux-x86-64.so.2
eKZQ
libc.so.6
__isoc99_scanf
puts
__stack_chk_fail
printf
strcmp
__libc_start_main
__gmon_start__
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
UH-p
AWAVA
AUATL
[]A\A]A^A_
Enter Password:
Incorrect!
Correct!
;*3$"
```

Collect info: the strings command

```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

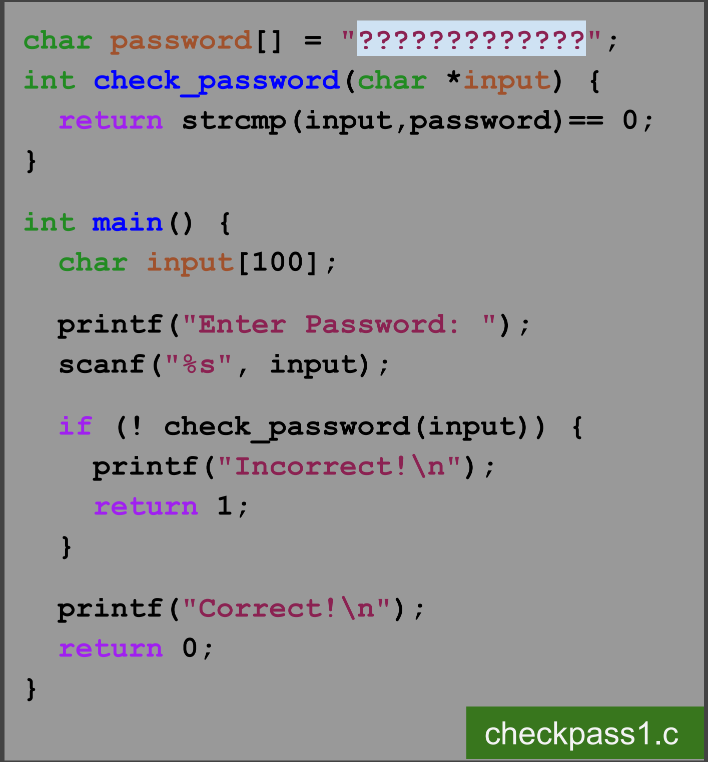
checkpass1.c



```
CS@kntu:lecture_reveng$ strings checkpass64s
/lib64/ld-linux-x86-64.so.2
eKZQ
libc.so.6
__isoc99_scanf
puts
__stack_chk_fail
printf
strcmp
__libc_start_main
__gmon_start__
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
UH-p
AWAVA
AUATL
[]A\A]A^A_
Enter Password:
Incorrect!
Correct!
;*3$"
```

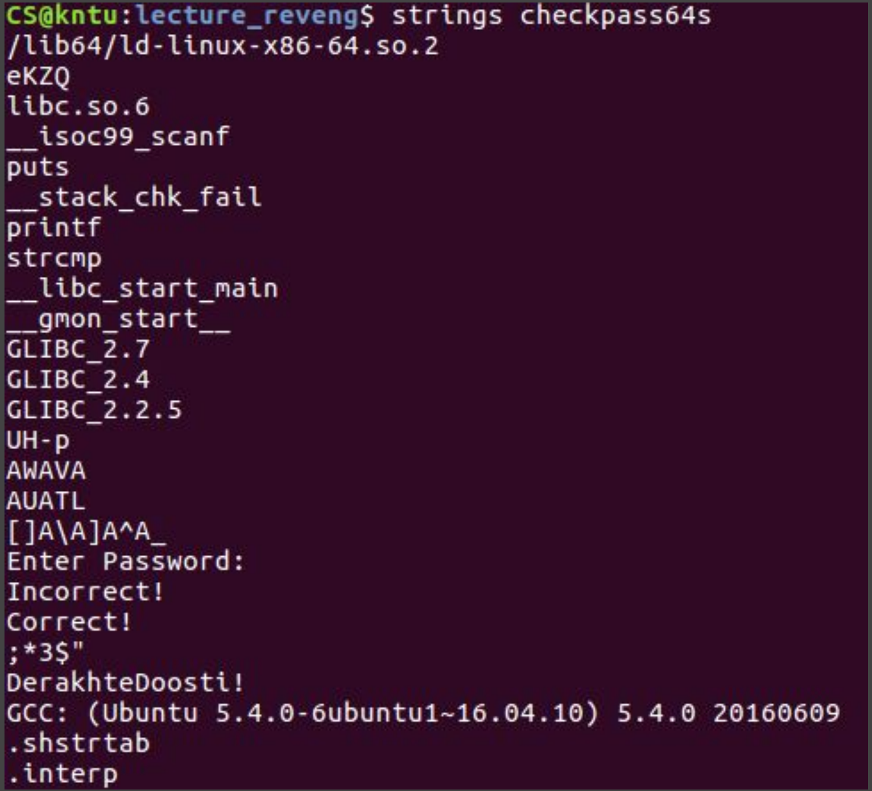
Our first reverse engineering effort

K. N. Toosi
University of Technology



```
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c



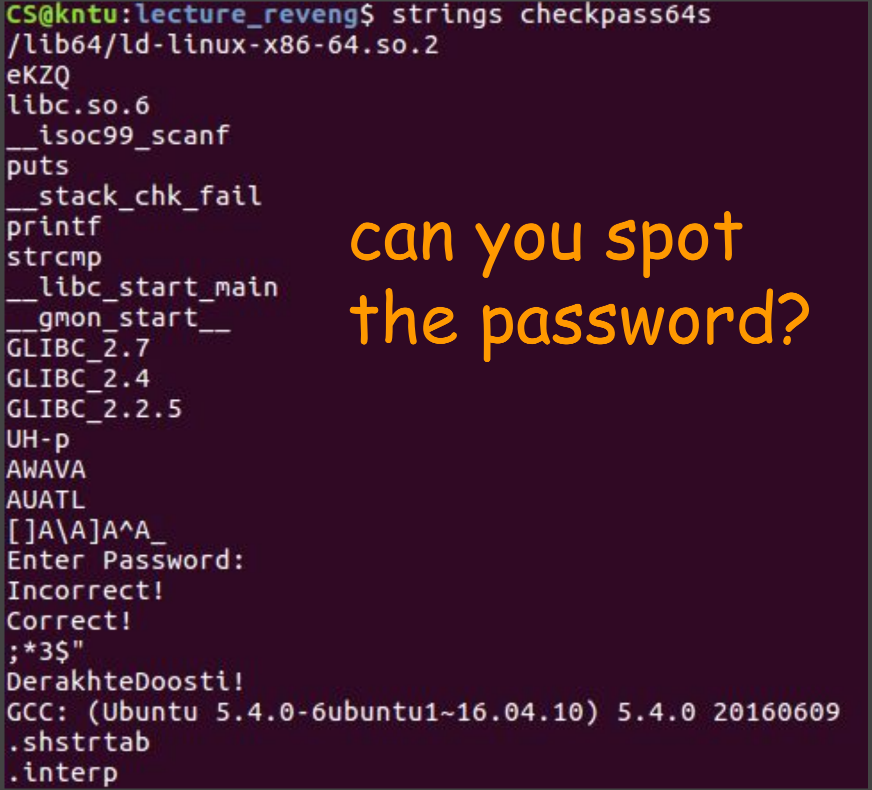
```
cs@kntu:lecture_reveng$ strings checkpass64s
/lib64/ld-linux-x86-64.so.2
eKZQ
libc.so.6
__isoc99_scanf
puts
__stack_chk_fail
printf
strcmp
__libc_start_main
__gmon_start__
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
UH-p
AWAVA
AUATL
[]A\A]A^A_
Enter Password:
Incorrect!
Correct!
;*3$"
DerakhteDoosti!
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.10) 5.4.0 20160609
.shstrtab
.interp
```

Our first reverse engineering effort

K. N. Toosi
University of Technology

```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c



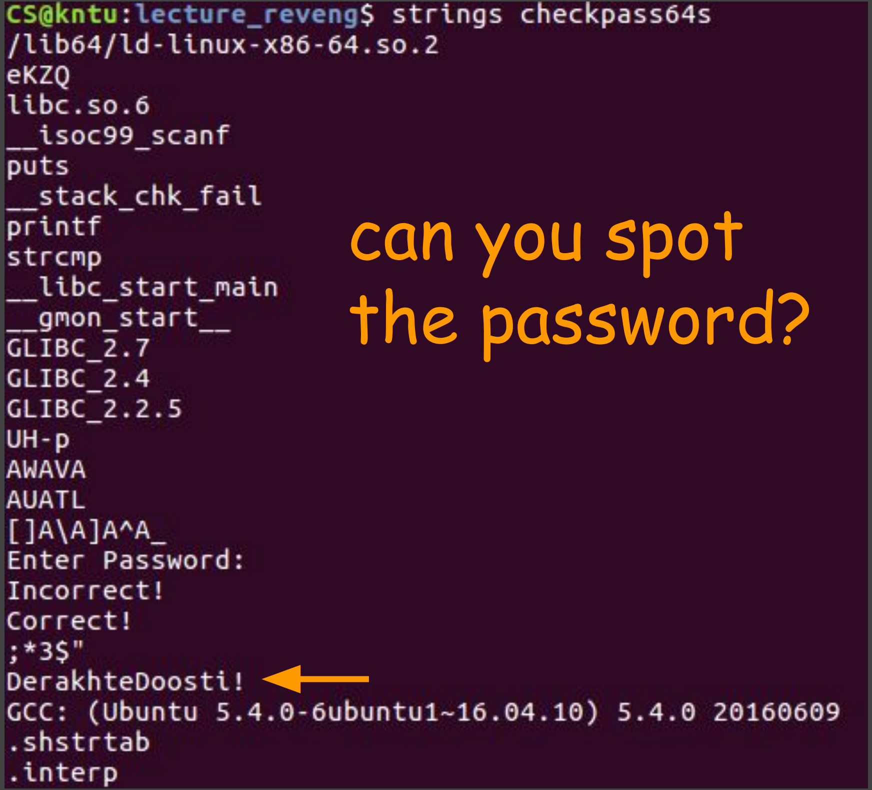
```
cs@kntu:lecture_reveng$ strings checkpass64s
/lib64/ld-linux-x86-64.so.2
eKZQ
libc.so.6
__isoc99_scanf
puts
__stack_chk_fail
printf
strcmp
__libc_start_main
__gmon_start__
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
UH-p
AWAVA
AUATL
[]A\A]A^A_
Enter Password:
Incorrect!
Correct!
;*3$"
DerakhteDoosti!
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.10) 5.4.0 20160609
.shstrtab
.interp
```

can you spot the password?

Our first reverse engineering effort

```c
char password[] = "?????????????";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c



```
CS@kntu:lecture_reveng$ strings checkpass64s
/lib64/ld-linux-x86-64.so.2
eKZQ
libc.so.6
__isoc99_scanf
puts
__stack_chk_fail
printf
strcmp
__libc_start_main
__gmon_start__
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
UH-p
AWAVA
AUATL
[]A\A]A^A_
Enter Password:
Incorrect!
Correct!
;*3$"
DerakhteDoosti!
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.10) 5.4.0 20160609
.shstrtab
.interp
```

can you spot the password?

Our first reverse engineering effort

K. N. Toosi
University of Technology

checkpass1.c

```
int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

I hide this for now!

```
char password[] = "??????????????";

int check_password(char *input) {
  return strcmp(input,password)== 0;
}
```

Our first reverse engineering effort

K. N. Toosi
University of Technology

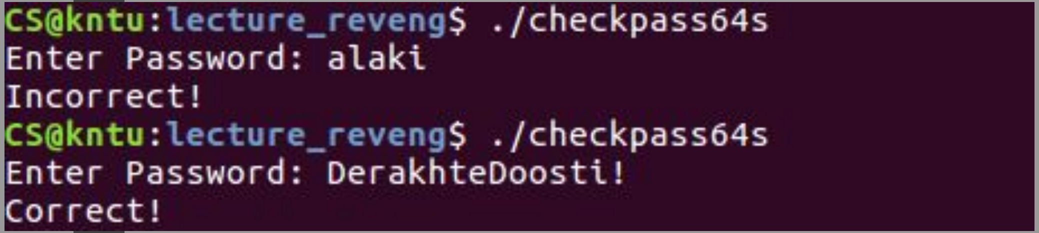
checkpass1.c

```
int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

I hide this for now!

```
char password[] = "??????????????";

int check_password(char *input) {
  return strcmp(input,password)== 0;
}
```



```
CS@kntu:lecture_reveng$ ./checkpass64s
Enter Password: alaki
Incorrect!
CS@kntu:lecture_reveng$ ./checkpass64s
Enter Password: DerakhteDoosti!
Correct!
```

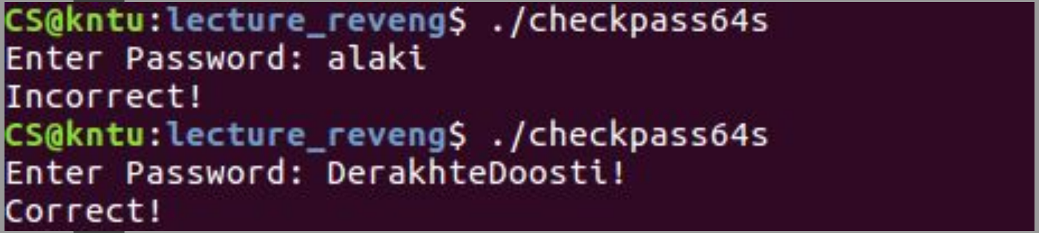
Our first reverse engineering effort

checkpass1.c

```
int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

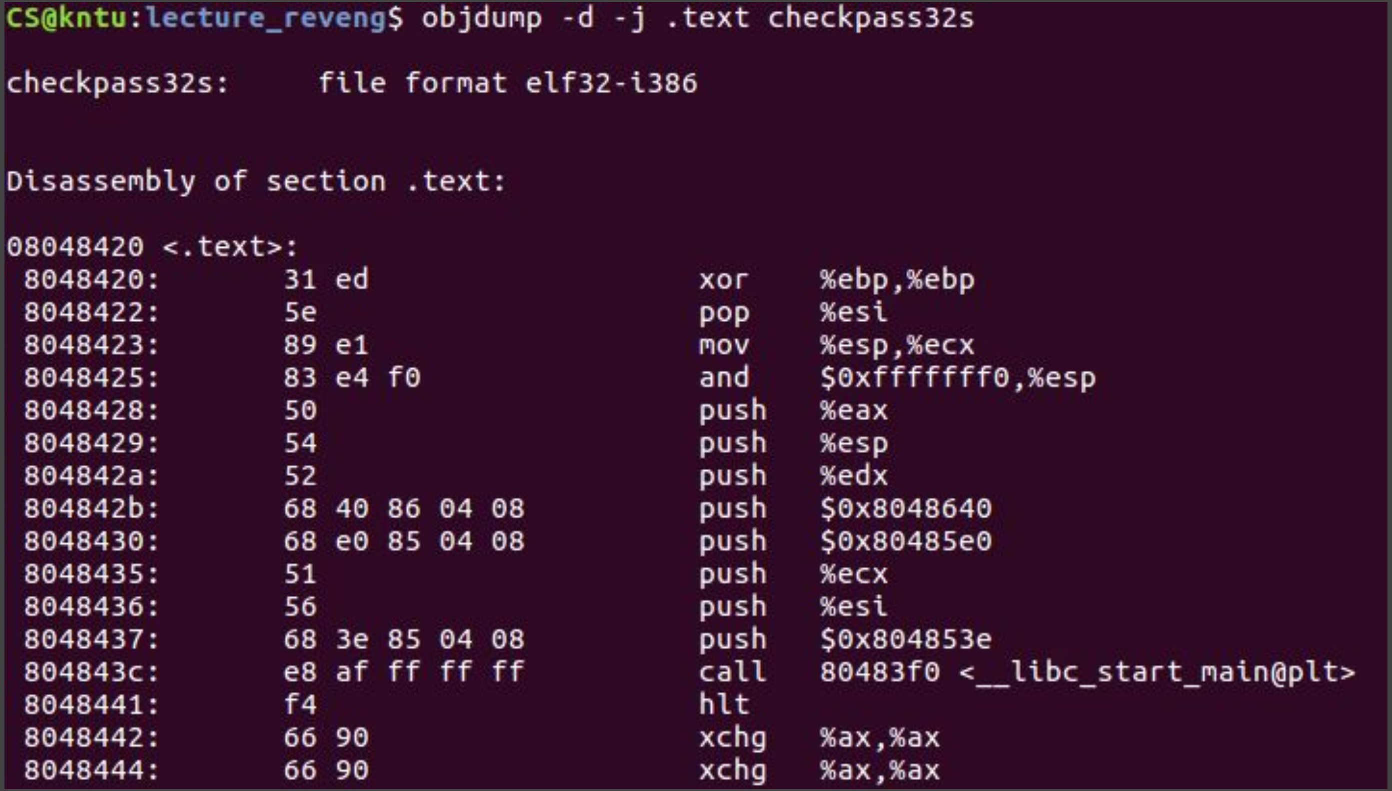
checkpass1.c

```
char password[] = "DerakhteDoosti!";

int check_password(char *input) {
  return strcmp(input,password)== 0;
}
```



```
cs@kntu:lecture_reveng$ ./checkpass64s
Enter Password: alaki
Incorrect!
cs@kntu:lecture_reveng$ ./checkpass64s
Enter Password: DerakhteDoosti!
Correct!
```

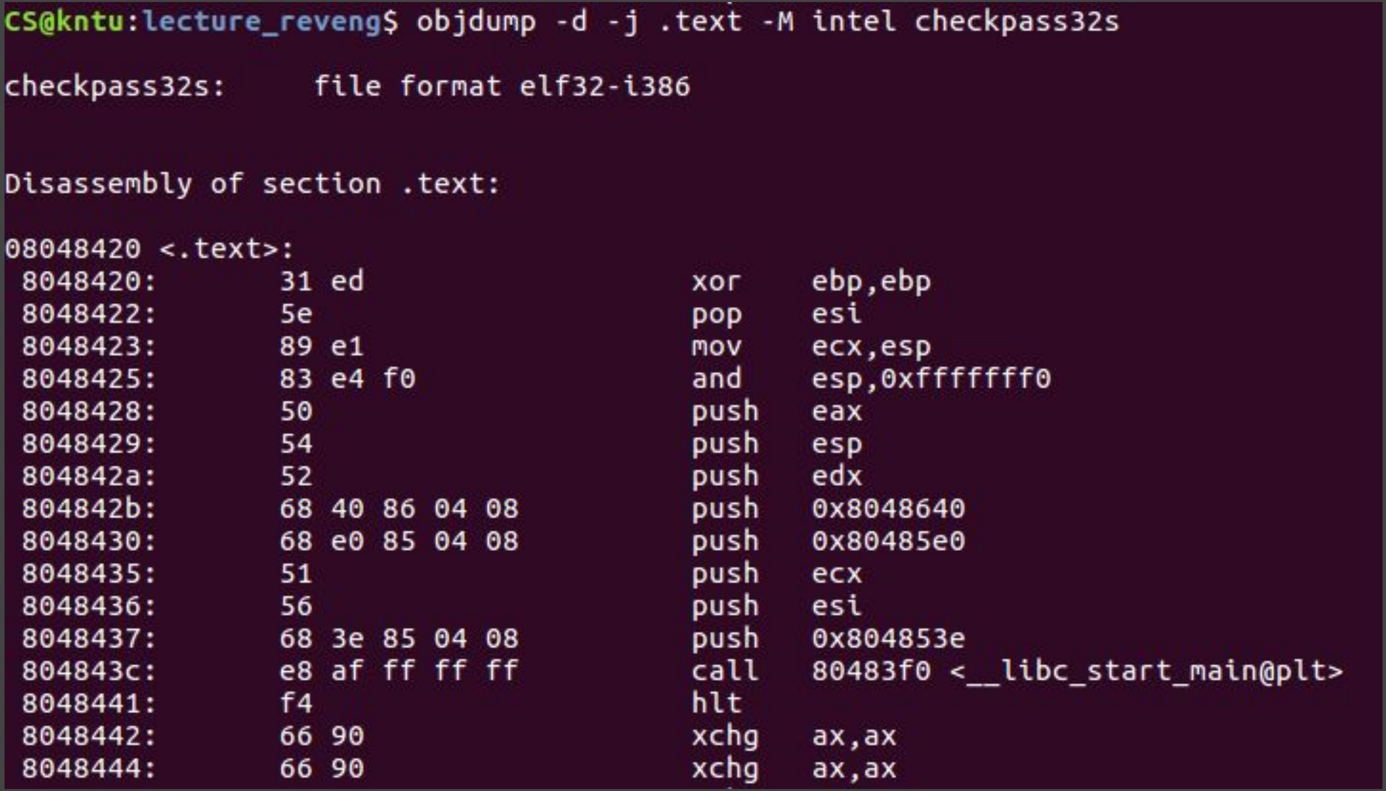
Disassembling: the objdump command



```
CS@kntu:lecture_reveng$ objdump -d -j .text checkpass32s

checkpass32s:     file format elf32-i386


Disassembly of section .text:

08048420 <.text>:
 8048420:       31 ed                   xor    %ebp,%ebp
 8048422:       5e                      pop    %esi
 8048423:       89 e1                   mov    %esp,%ecx
 8048425:       83 e4 f0                and    $0xfffffff0,%esp
 8048428:       50                      push   %eax
 8048429:       54                      push   %esp
 804842a:       52                      push   %edx
 804842b:       68 40 86 04 08          push   $0x8048640
 8048430:       68 e0 85 04 08          push   $0x80485e0
 8048435:       51                      push   %ecx
 8048436:       56                      push   %esi
 8048437:       68 3e 85 04 08          push   $0x804853e
 804843c:       e8 af ff ff ff          call   80483f0 <__libc_start_main@plt>
 8048441:       f4                      hlt
 8048442:       66 90                   xchg   %ax,%ax
 8048444:       66 90                   xchg   %ax,%ax
```

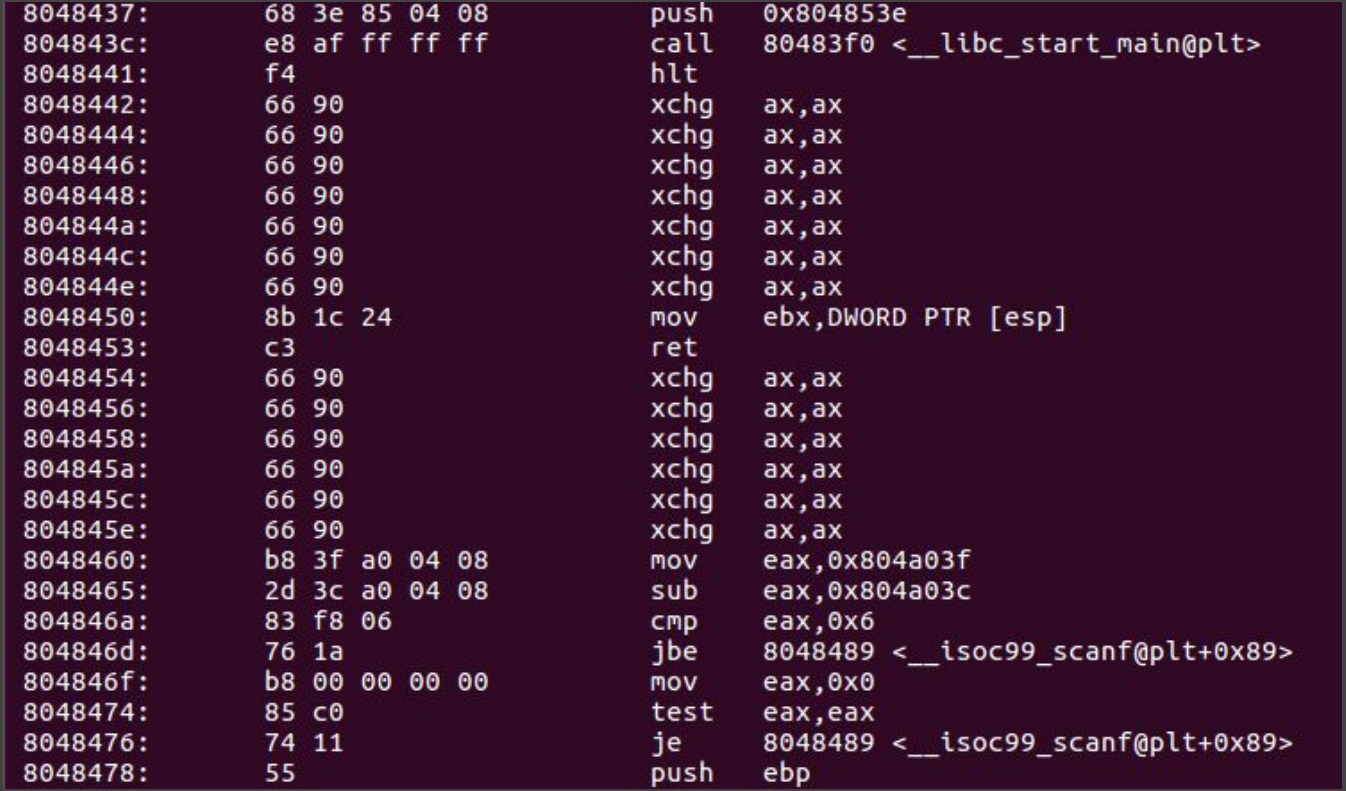
Disassembling: the objdump command

K. N. Toosi
University of Technology

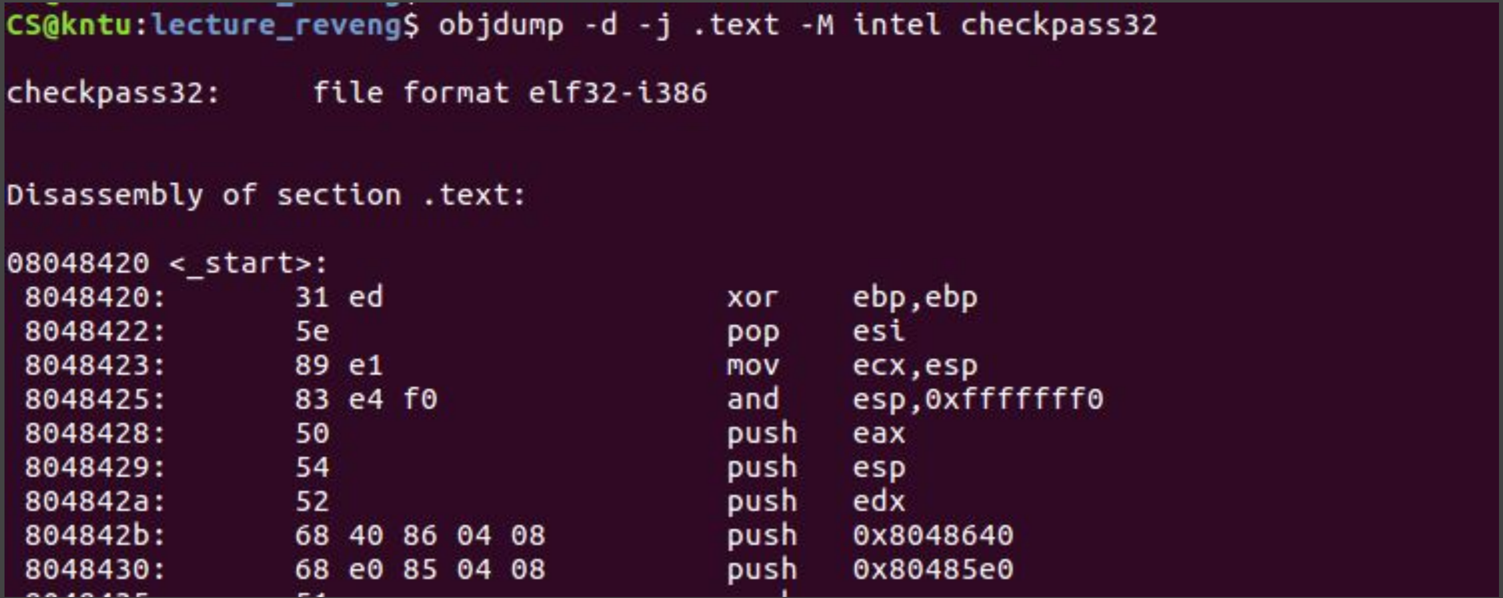


```
CS@kntu:lecture_reveng$ objdump -d -j .text -M intel checkpass32s

checkpass32s:     file format elf32-i386


Disassembly of section .text:

08048420 <.text>:
 8048420:       31 ed                   xor    ebp,ebp
 8048422:       5e                      pop    esi
 8048423:       89 e1                   mov    ecx,esp
 8048425:       83 e4 f0                and    esp,0xfffffff0
 8048428:       50                      push   eax
 8048429:       54                      push   esp
 804842a:       52                      push   edx
 804842b:       68 40 86 04 08          push   0x8048640
 8048430:       68 e0 85 04 08          push   0x80485e0
 8048435:       51                      push   ecx
 8048436:       56                      push   esi
 8048437:       68 3e 85 04 08          push   0x804853e
 804843c:       e8 af ff ff ff          call   80483f0 <__libc_start_main@plt>
 8048441:       f4                      hlt
 8048442:       66 90                   xchg   ax,ax
 8048444:       66 90                   xchg   ax,ax
```

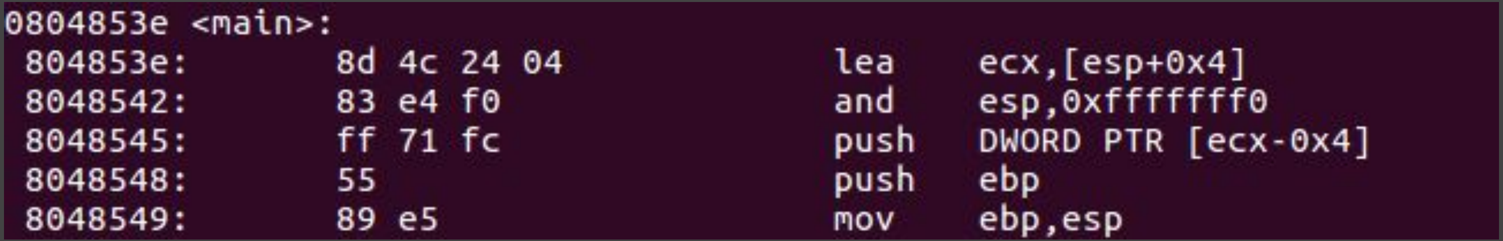
Disassembling: the objdump command


```
8048437:       68 3e 85 04 08          push   0x804853e
804843c:       e8 af ff ff ff          call   80483f0 <__libc_start_main@plt>
8048441:       f4                      hlt
8048442:       66 90                   xchg   ax,ax
8048444:       66 90                   xchg   ax,ax
8048446:       66 90                   xchg   ax,ax
8048448:       66 90                   xchg   ax,ax
804844a:       66 90                   xchg   ax,ax
804844c:       66 90                   xchg   ax,ax
804844e:       66 90                   xchg   ax,ax
8048450:       8b 1c 24                mov    ebx,DWORD PTR [esp]
8048453:       c3                      ret
8048454:       66 90                   xchg   ax,ax
8048456:       66 90                   xchg   ax,ax
8048458:       66 90                   xchg   ax,ax
804845a:       66 90                   xchg   ax,ax
804845c:       66 90                   xchg   ax,ax
804845e:       66 90                   xchg   ax,ax
8048460:       b8 3f a0 04 08          mov    eax,0x804a03f
8048465:       2d 3c a0 04 08          sub    eax,0x804a03c
804846a:       83 f8 06                cmp    eax,0x6
804846d:       76 1a                   jbe    8048489 <__isoc99_scanf@plt+0x89>
804846f:       b8 00 00 00 00          mov    eax,0x0
8048474:       85 c0                   test   eax,eax
8048476:       74 11                   je     8048489 <__isoc99_scanf@plt+0x89>
8048478:       55                      push   ebp
```

Disassembling: unstripped programs



```
CS@kntu:lecture_reveng$ objdump -d -j .text -M intel checkpass32

checkpass32:     file format elf32-i386


Disassembly of section .text:

08048420 <_start>:
 8048420:       31 ed                   xor    ebp,ebp
 8048422:       5e                      pop    esi
 8048423:       89 e1                   mov    ecx,esp
 8048425:       83 e4 f0                and    esp,0xfffffff0
 8048428:       50                      push   eax
 8048429:       54                      push   esp
 804842a:       52                      push   edx
 804842b:       68 40 86 04 08          push   0x8048640
 8048430:       68 e0 85 04 08          push   0x80485e0
```

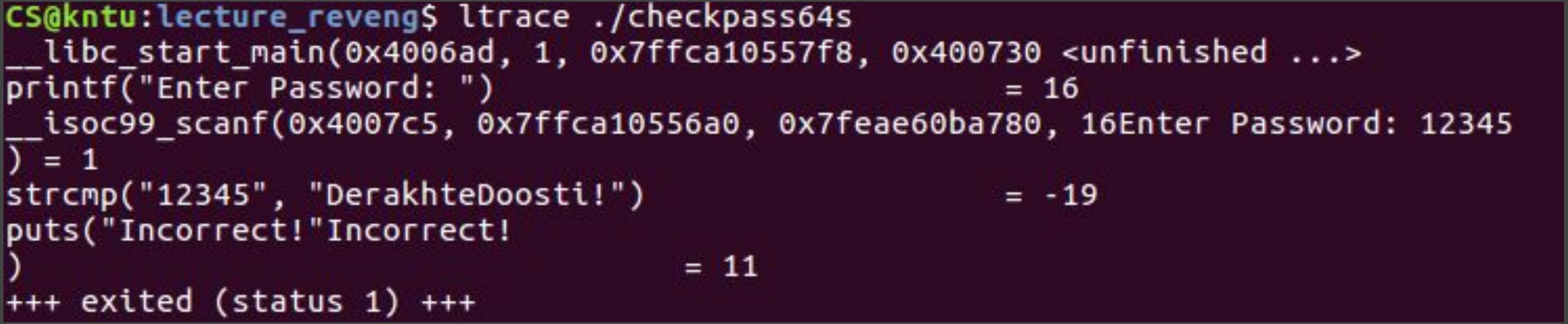


```
0804853e <main>:
 804853e:       8d 4c 24 04             lea    ecx,[esp+0x4]
 8048542:       83 e4 f0                and    esp,0xfffffff0
 8048545:       ff 71 fc                push   DWORD PTR [ecx-0x4]
 8048548:       55                      push   ebp
 8048549:       89 e5                   mov    ebp,esp
```

Code Analysis

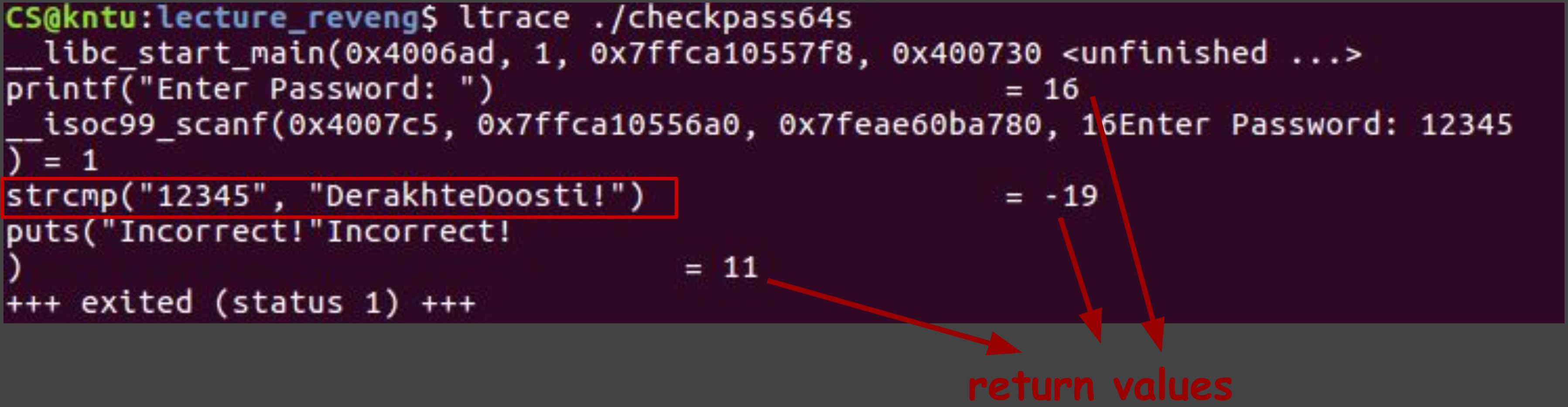
- Static Analysis
- Dynamic Analysis

ltrace: trace library calls



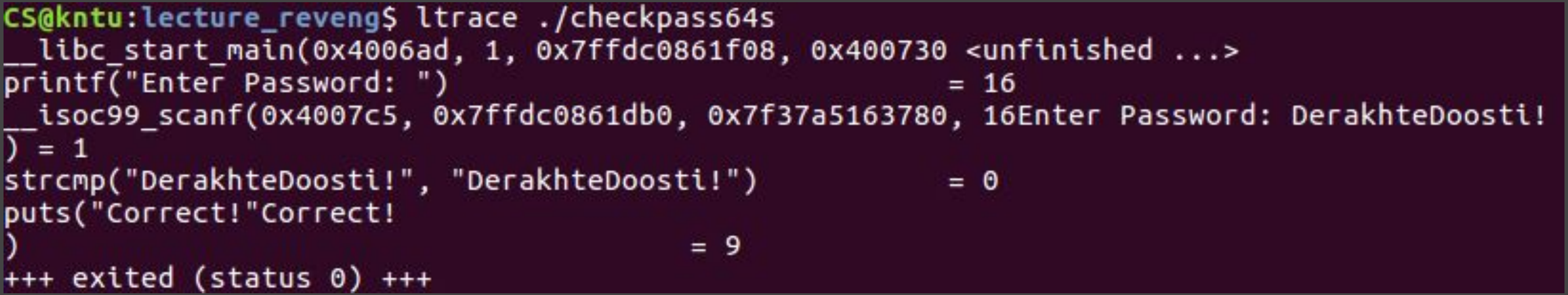
```
CS@kntu:lecture_reveng$ ltrace ./checkpass64s
__libc_start_main(0x4006ad, 1, 0x7ffca10557f8, 0x400730 <unfinished ...>
printf("Enter Password: ")                                   = 16
__isoc99_scanf(0x4007c5, 0x7ffca10556a0, 0x7feae60ba780, 16Enter Password: 12345
) = 1
strcmp("12345", "DerakhteDoosti!")                           = -19
puts("Incorrect!"Incorrect!
)                                        = 11
+++ exited (status 1) +++
```

ltrace: trace library calls



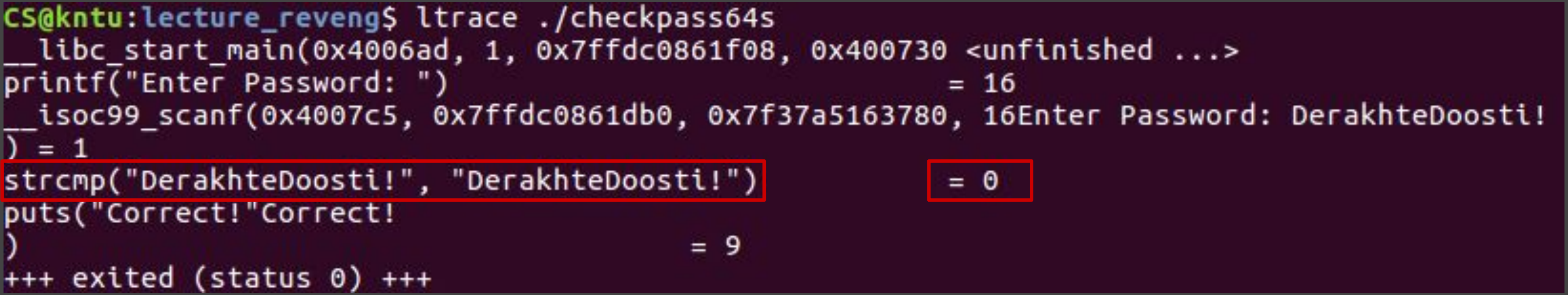
```
CS@kntu:lecture_reveng$ ltrace ./checkpass64s
__libc_start_main(0x4006ad, 1, 0x7ffca10557f8, 0x400730 <unfinished ...>
printf("Enter Password: ")                                     = 16
__isoc99_scanf(0x4007c5, 0x7ffca10556a0, 0x7feae60ba780, 16Enter Password: 12345
) = 1
strcmp("12345", "DerakhteDoosti!")                             = -19
puts("Incorrect!"Incorrect!
)                                      = 11
+++ exited (status 1) +++
```

ltrace: trace library calls



```
CS@kntu:lecture_reveng$ ltrace ./checkpass64s
__libc_start_main(0x4006ad, 1, 0x7ffdc0861f08, 0x400730 <unfinished ...>
printf("Enter Password: ")                                   = 16
__isoc99_scanf(0x4007c5, 0x7ffdc0861db0, 0x7f37a5163780, 16Enter Password: DerakhteDoosti!
) = 1
strcmp("DerakhteDoosti!", "DerakhteDoosti!")                 = 0
puts("Correct!"Correct!
)                                               = 9
+++ exited (status 0) +++
```

ltrace: trace library calls



```
CS@kntu:lecture_reveng$ ltrace ./checkpass64s
__libc_start_main(0x4006ad, 1, 0x7ffdc0861f08, 0x400730 <unfinished ...>
printf("Enter Password: ")                                            = 16
__isoc99_scanf(0x4007c5, 0x7ffdc0861db0, 0x7f37a5163780, 16Enter Password: DerakhteDoosti!
) = 1
strcmp("DerakhteDoosti!", "DerakhteDoosti!")                         = 0
puts("Correct!"Correct!
)                                       = 9
+++ exited (status 0) +++
```

The GNU debugger (GDB)

- Very powerful debugging tool
- Part of the GNU project
- started by **Richard Stallman**
- **low-level** and **high-level** debugging
- debugs high-level languages: C, C++, Java, Objective-C, Ada, Go, etc.
- command line interface
- GUI front ends like DDD, KDbg

Debug using GDB

K. N. Toosi
University of Technology

- Open executable
- set a breakpoint
- run!

Debug using GDB

- Open executable
- set a breakpoint
- run!

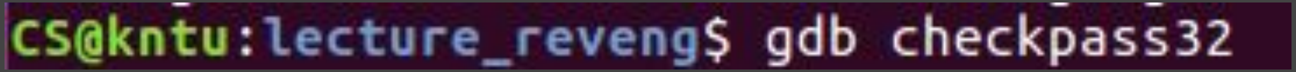
- Here, we only cover low-level debugging.

starting GDB

```
$ gdb program_name
```

starting GDB

```
$ gdb program_name
```

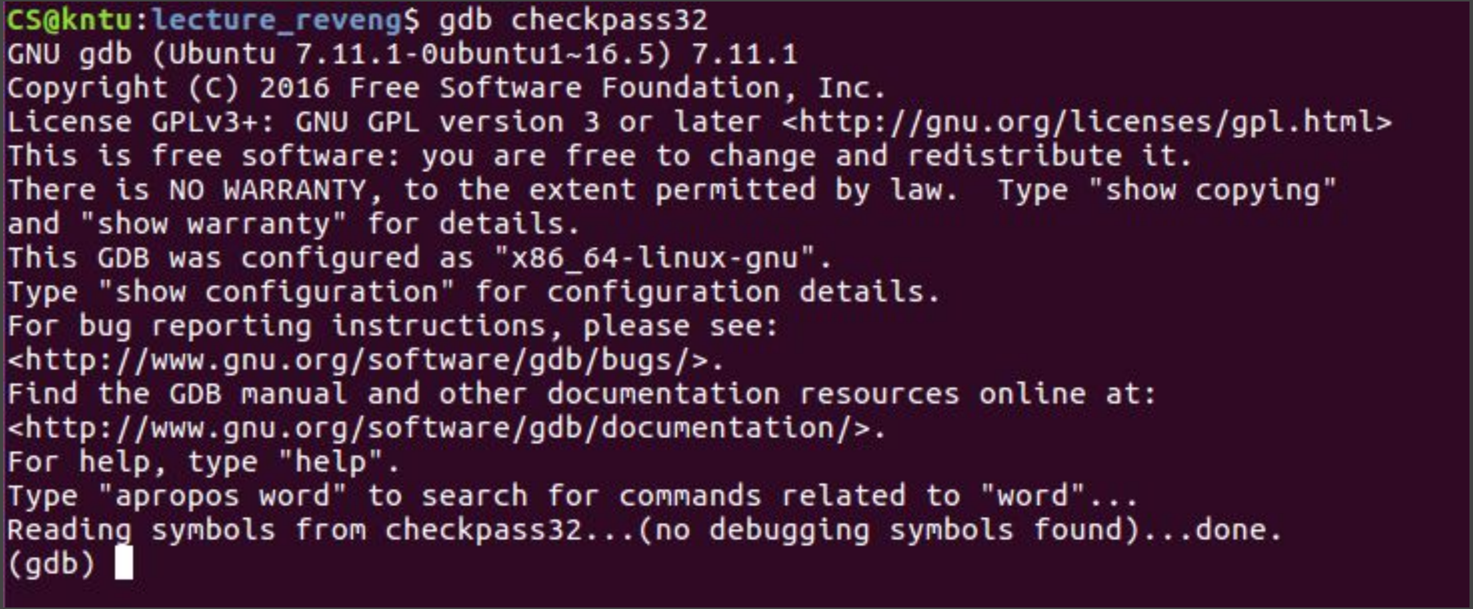


```
CS@kntu:lecture_reveng$ gdb checkpass32
```

starting GDB

K. N. Toosi
University of Technology

```
$ gdb program_name
```



```
cs@kntu:lecture_reveng$ gdb checkpass32
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from checkpass32...(no debugging symbols found)...done.
(gdb) 
```

Basic GDB commands

`h/help`	gdb help
`i/info`	current program info

K. N. Toosi
University of Technology

Basic GDB commands

`b/break`	set breakpoint
`r/run`	run the program
`nexti`	run next instruction (step over calls)
`stepi`	run next instruction (step into calls)
`finish`	exit function
`c/cont/continue`	continue running program

Basic GDB commands

`info registers`	integer register contents
`disass/disassemble`	disassemble
`x`	examine memory

Examine memory

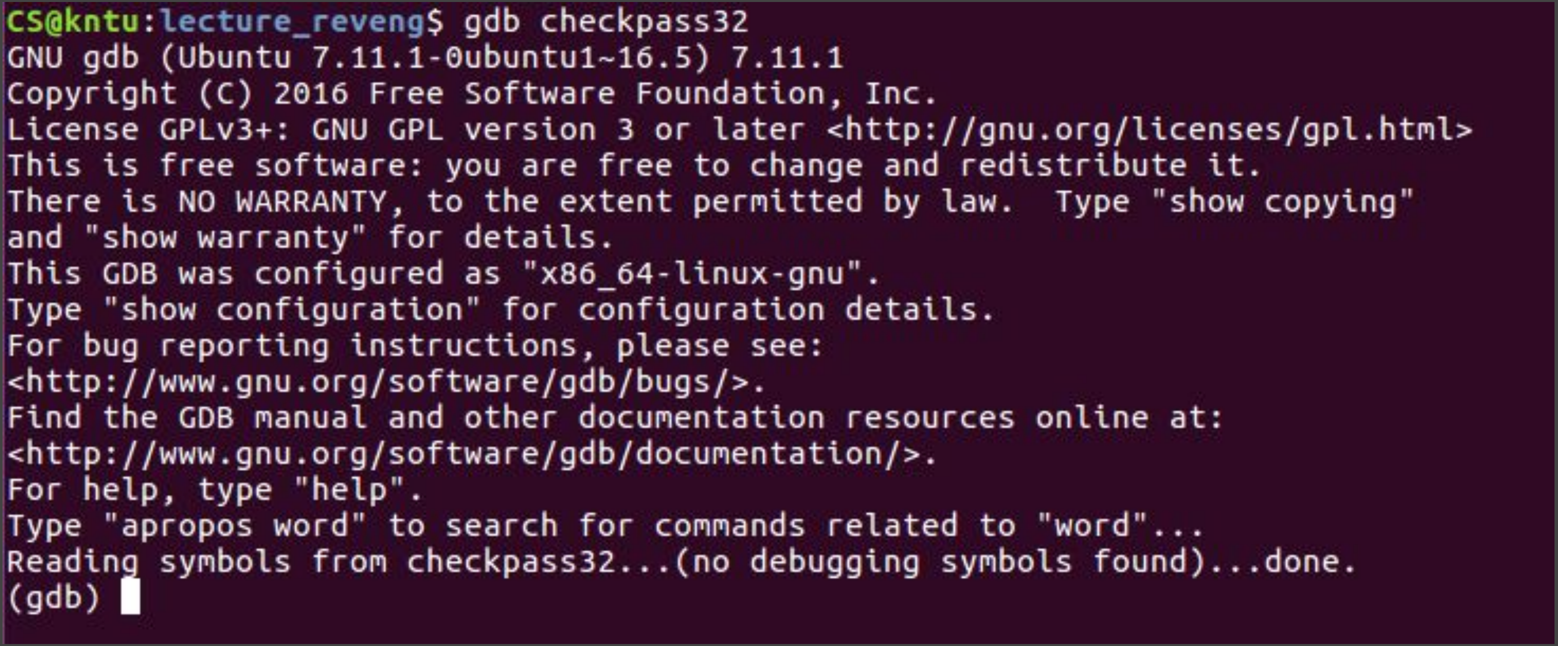
- `x/FMT address`
- `FMT: [num]format[size]`

`x/c ADDRESS`	print character at ADDRESS
`x/4c ADDRESS`	print 4 characters starting at ADDRESS
`x/s ADDRESS`	print (null terminated) string at ADDRESS
`x/i ADDRESS`	print instruction at ADDRESS
`x/hb ADDRESS`	print 1-byte hex number at ADDRESS
`x/10hg ADDRESS`	print 10 64-bits hex numbers at ADDRESS

- `see help x`

Getting started with GDB

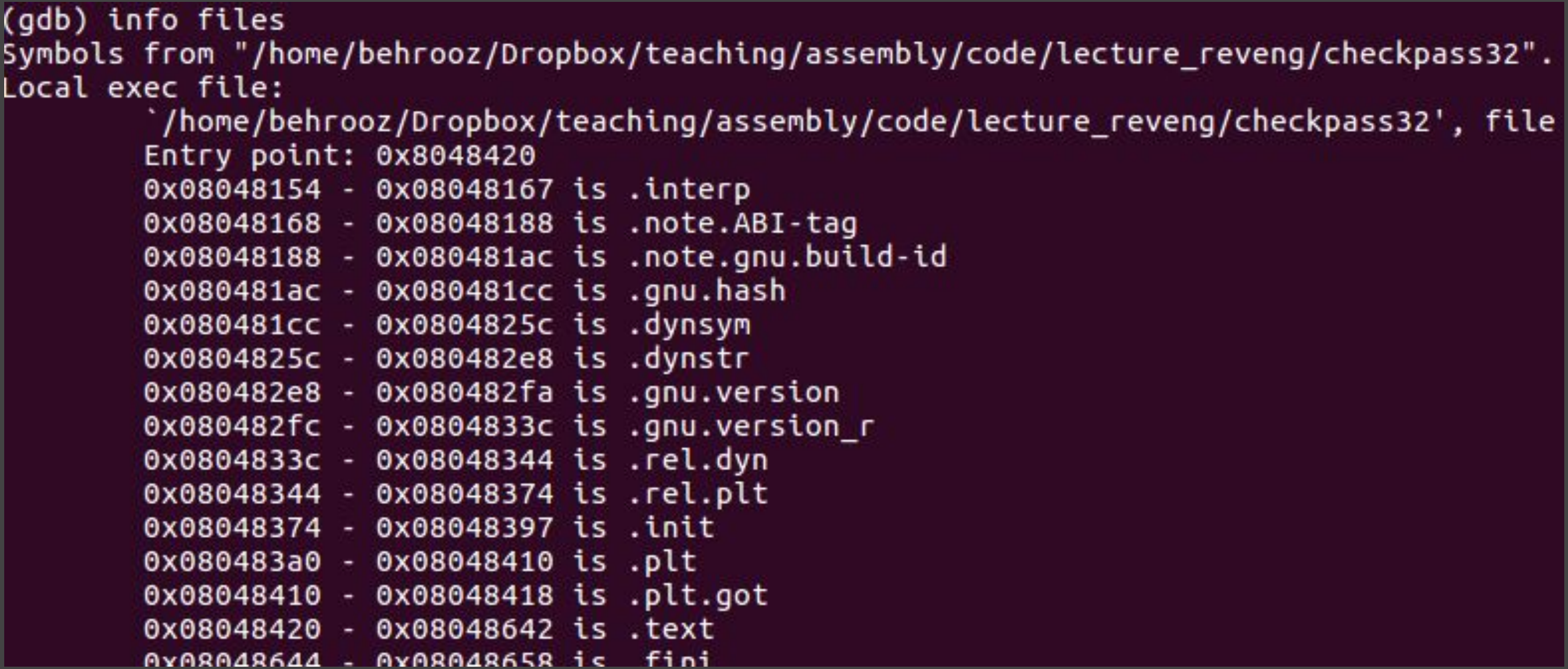
- Let's start with an **unstripped** program



```
CS@kntu:lecture_reveng$ gdb checkpass32
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from checkpass32...(no debugging symbols found)...done.
(gdb) 
```

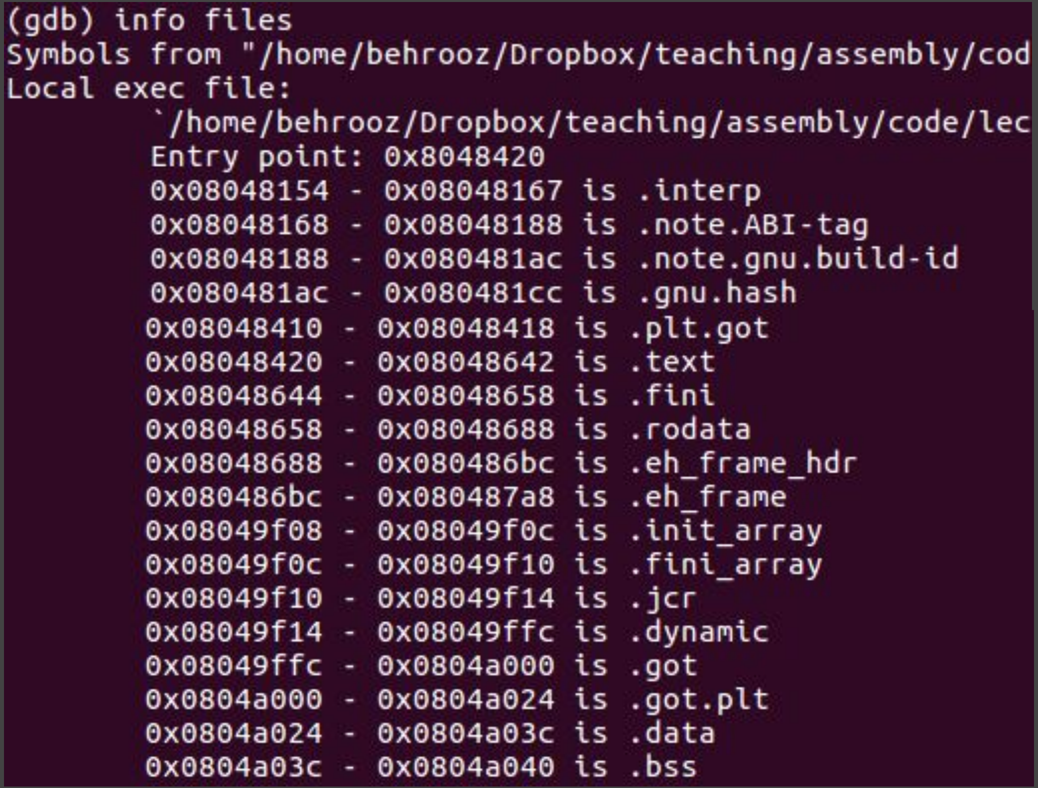
Getting started with GDB: Unstripped

K. N. Toosi
University of Technology



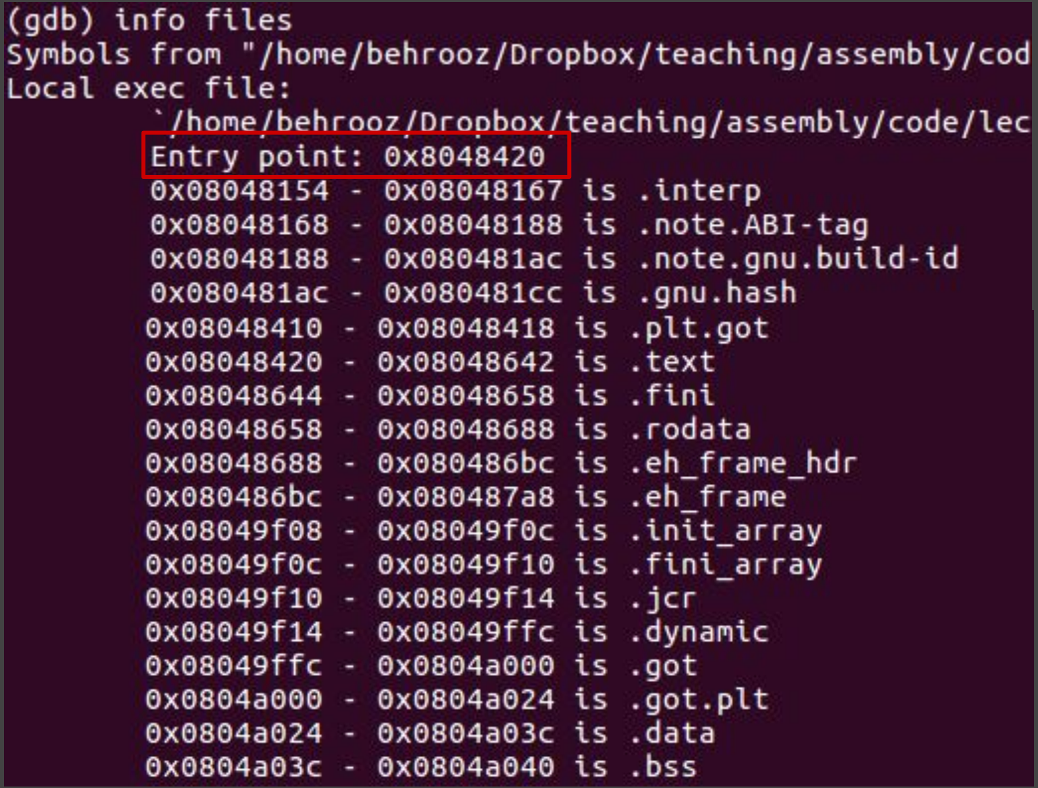
```
(gdb) info files
Symbols from "/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpass32".
Local exec file:
        `/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpass32', file
        Entry point: 0x8048420
        0x08048154 - 0x08048167 is .interp
        0x08048168 - 0x08048188 is .note.ABI-tag
        0x08048188 - 0x080481ac is .note.gnu.build-id
        0x080481ac - 0x080481cc is .gnu.hash
        0x080481cc - 0x0804825c is .dynsym
        0x0804825c - 0x080482e8 is .dynstr
        0x080482e8 - 0x080482fa is .gnu.version
        0x080482fc - 0x0804833c is .gnu.version_r
        0x0804833c - 0x08048344 is .rel.dyn
        0x08048344 - 0x08048374 is .rel.plt
        0x08048374 - 0x08048397 is .init
        0x080483a0 - 0x08048410 is .plt
        0x08048410 - 0x08048418 is .plt.got
        0x08048420 - 0x08048642 is .text
        0x08048644 - 0x08048658 is .fini
```

Getting started with GDB: Unstripped



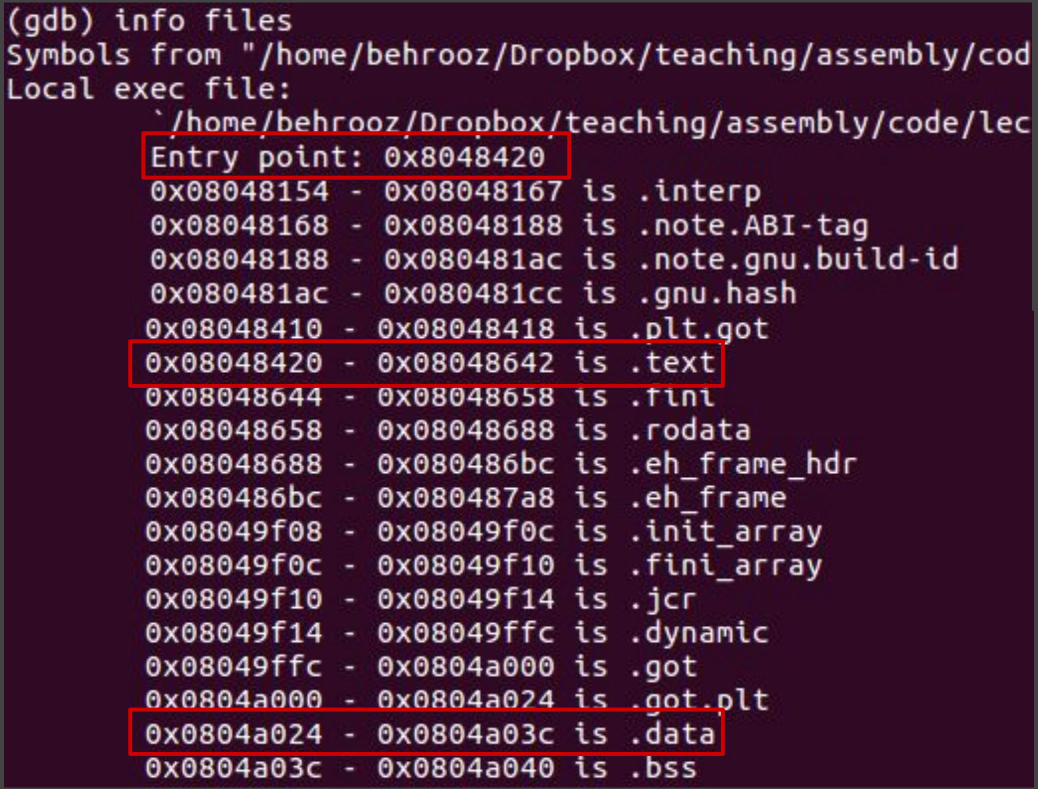
```
(gdb) info files
Symbols from "/home/behrooz/Dropbox/teaching/assembly/cod
Local exec file:
        `/home/behrooz/Dropbox/teaching/assembly/code/lec
        Entry point: 0x8048420
        0x08048154 - 0x08048167 is .interp
        0x08048168 - 0x08048188 is .note.ABI-tag
        0x08048188 - 0x080481ac is .note.gnu.build-id
        0x080481ac - 0x080481cc is .gnu.hash
        0x08048410 - 0x08048418 is .plt.got
        0x08048420 - 0x08048642 is .text
        0x08048644 - 0x08048658 is .fini
        0x08048658 - 0x08048688 is .rodata
        0x08048688 - 0x080486bc is .eh_frame_hdr
        0x080486bc - 0x080487a8 is .eh_frame
        0x08049f08 - 0x08049f0c is .init_array
        0x08049f0c - 0x08049f10 is .fini_array
        0x08049f10 - 0x08049f14 is .jcr
        0x08049f14 - 0x08049ffc is .dynamic
        0x08049ffc - 0x0804a000 is .got
        0x0804a000 - 0x0804a024 is .got.plt
        0x0804a024 - 0x0804a03c is .data
        0x0804a03c - 0x0804a040 is .bss
```

Getting started with GDB: Unstripped



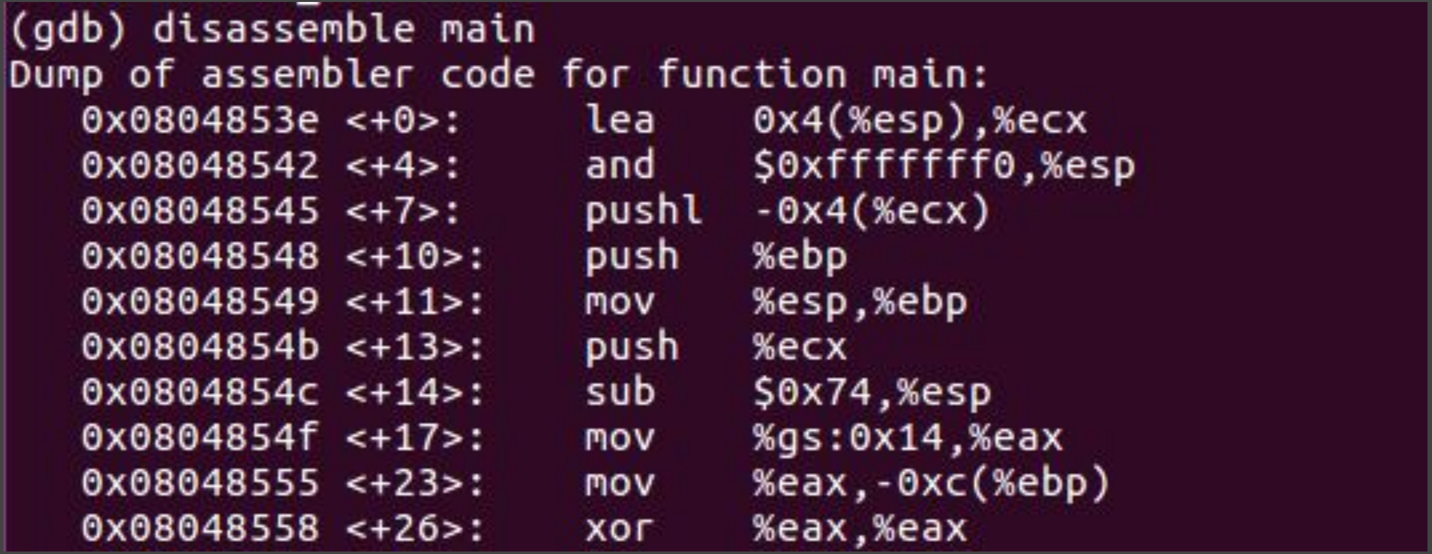
```
(gdb) info files
Symbols from "/home/behrooz/Dropbox/teaching/assembly/cod
Local exec file:
        `/home/behrooz/Dropbox/teaching/assembly/code/lec
        Entry point: 0x8048420
        0x08048154 - 0x08048167 is .interp
        0x08048168 - 0x08048188 is .note.ABI-tag
        0x08048188 - 0x080481ac is .note.gnu.build-id
        0x080481ac - 0x080481cc is .gnu.hash
        0x08048410 - 0x08048418 is .plt.got
        0x08048420 - 0x08048642 is .text
        0x08048644 - 0x08048658 is .fini
        0x08048658 - 0x08048688 is .rodata
        0x08048688 - 0x080486bc is .eh_frame_hdr
        0x080486bc - 0x080487a8 is .eh_frame
        0x08049f08 - 0x08049f0c is .init_array
        0x08049f0c - 0x08049f10 is .fini_array
        0x08049f10 - 0x08049f14 is .jcr
        0x08049f14 - 0x08049ffc is .dynamic
        0x08049ffc - 0x0804a000 is .got
        0x0804a000 - 0x0804a024 is .got.plt
        0x0804a024 - 0x0804a03c is .data
        0x0804a03c - 0x0804a040 is .bss
```

Getting started with GDB: Unstripped



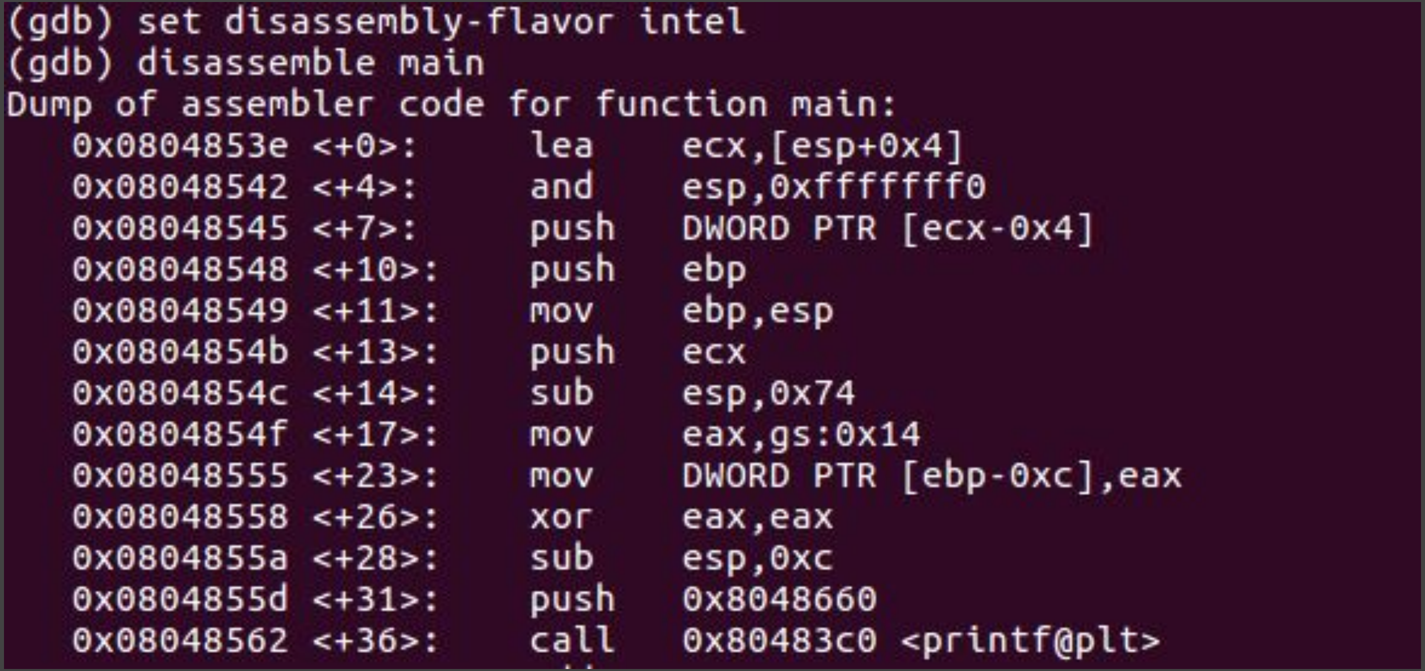
```
(gdb) info files
Symbols from "/home/behrooz/Dropbox/teaching/assembly/cod
Local exec file:
        `/home/behrooz/Dropbox/teaching/assembly/code/lec
        Entry point: 0x8048420
        0x08048154 - 0x08048167 is .interp
        0x08048168 - 0x08048188 is .note.ABI-tag
        0x08048188 - 0x080481ac is .note.gnu.build-id
        0x080481ac - 0x080481cc is .gnu.hash
        0x08048410 - 0x08048418 is .plt.got
        0x08048420 - 0x08048642 is .text
        0x08048644 - 0x08048658 is .fini
        0x08048658 - 0x08048688 is .rodata
        0x08048688 - 0x080486bc is .eh_frame_hdr
        0x080486bc - 0x080487a8 is .eh_frame
        0x08049f08 - 0x08049f0c is .init_array
        0x08049f0c - 0x08049f10 is .fini_array
        0x08049f10 - 0x08049f14 is .jcr
        0x08049f14 - 0x08049ffc is .dynamic
        0x08049ffc - 0x0804a000 is .got
        0x0804a000 - 0x0804a024 is .got.plt
        0x0804a024 - 0x0804a03c is .data
        0x0804a03c - 0x0804a040 is .bss
```

Getting started with GDB: Unstripped



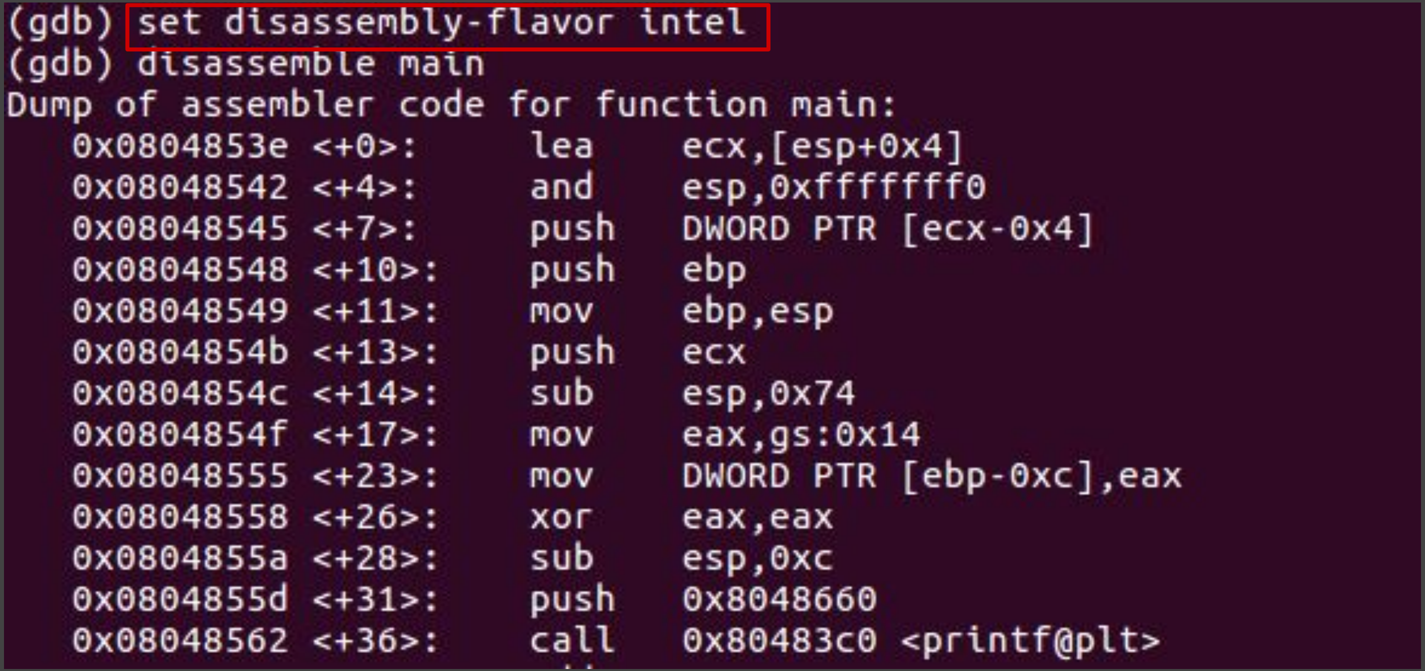
```
(gdb) disassemble main
Dump of assembler code for function main:
   0x0804853e <+0>:     lea    0x4(%esp),%ecx
   0x08048542 <+4>:     and    $0xfffffff0,%esp
   0x08048545 <+7>:     pushl  -0x4(%ecx)
   0x08048548 <+10>:    push   %ebp
   0x08048549 <+11>:    mov    %esp,%ebp
   0x0804854b <+13>:    push   %ecx
   0x0804854c <+14>:    sub    $0x74,%esp
   0x0804854f <+17>:    mov    %gs:0x14,%eax
   0x08048555 <+23>:    mov    %eax,-0xc(%ebp)
   0x08048558 <+26>:    xor    %eax,%eax
```

Getting started with GDB: Unstripped



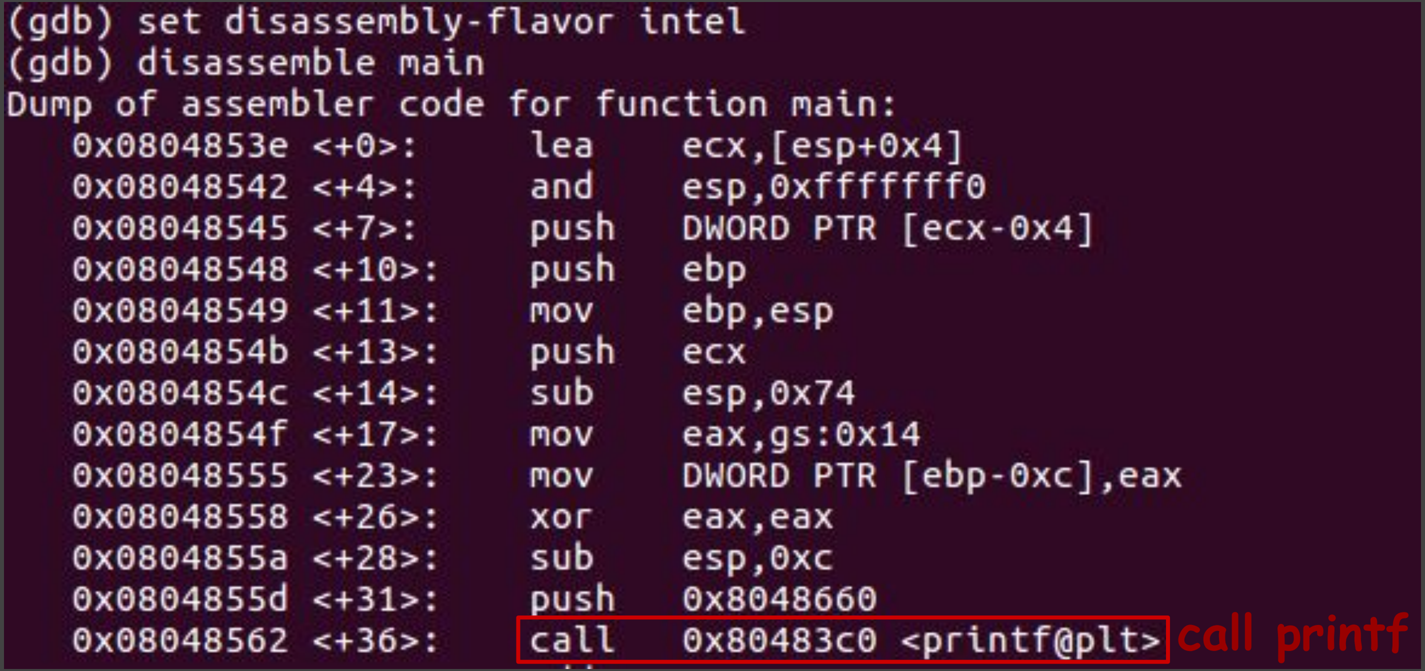
```
(gdb) set disassembly-flavor intel
(gdb) disassemble main
Dump of assembler code for function main:
   0x0804853e <+0>:     lea    ecx,[esp+0x4]
   0x08048542 <+4>:     and    esp,0xfffffff0
   0x08048545 <+7>:     push   DWORD PTR [ecx-0x4]
   0x08048548 <+10>:    push   ebp
   0x08048549 <+11>:    mov    ebp,esp
   0x0804854b <+13>:    push   ecx
   0x0804854c <+14>:    sub    esp,0x74
   0x0804854f <+17>:    mov    eax,gs:0x14
   0x08048555 <+23>:    mov    DWORD PTR [ebp-0xc],eax
   0x08048558 <+26>:    xor    eax,eax
   0x0804855a <+28>:    sub    esp,0xc
   0x0804855d <+31>:    push   0x8048660
   0x08048562 <+36>:    call   0x80483c0 <printf@plt>
```

Getting started with GDB: Unstripped



```
(gdb) set disassembly-flavor intel
(gdb) disassemble main
Dump of assembler code for function main:
   0x0804853e <+0>:     lea    ecx,[esp+0x4]
   0x08048542 <+4>:     and    esp,0xfffffff0
   0x08048545 <+7>:     push   DWORD PTR [ecx-0x4]
   0x08048548 <+10>:    push   ebp
   0x08048549 <+11>:    mov    ebp,esp
   0x0804854b <+13>:    push   ecx
   0x0804854c <+14>:    sub    esp,0x74
   0x0804854f <+17>:    mov    eax,gs:0x14
   0x08048555 <+23>:    mov    DWORD PTR [ebp-0xc],eax
   0x08048558 <+26>:    xor    eax,eax
   0x0804855a <+28>:    sub    esp,0xc
   0x0804855d <+31>:    push   0x8048660
   0x08048562 <+36>:    call   0x80483c0 <printf@plt>
```

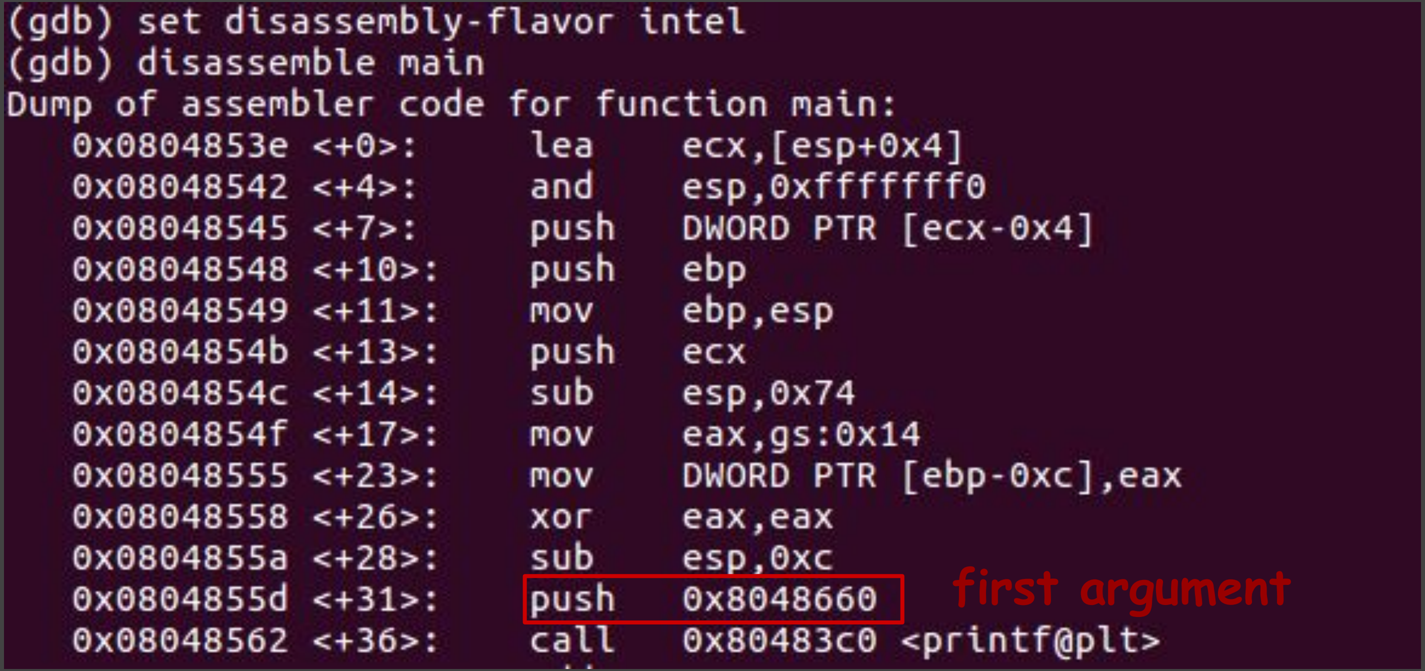
Getting started with GDB: Unstripped



```
(gdb) set disassembly-flavor intel
(gdb) disassemble main
Dump of assembler code for function main:
   0x0804853e <+0>:     lea    ecx,[esp+0x4]
   0x08048542 <+4>:     and    esp,0xfffffff0
   0x08048545 <+7>:     push   DWORD PTR [ecx-0x4]
   0x08048548 <+10>:    push   ebp
   0x08048549 <+11>:    mov    ebp,esp
   0x0804854b <+13>:    push   ecx
   0x0804854c <+14>:    sub    esp,0x74
   0x0804854f <+17>:    mov    eax,gs:0x14
   0x08048555 <+23>:    mov    DWORD PTR [ebp-0xc],eax
   0x08048558 <+26>:    xor    eax,eax
   0x0804855a <+28>:    sub    esp,0xc
   0x0804855d <+31>:    push   0x8048660
   0x08048562 <+36>:    call   0x80483c0 <printf@plt>
```

call printf

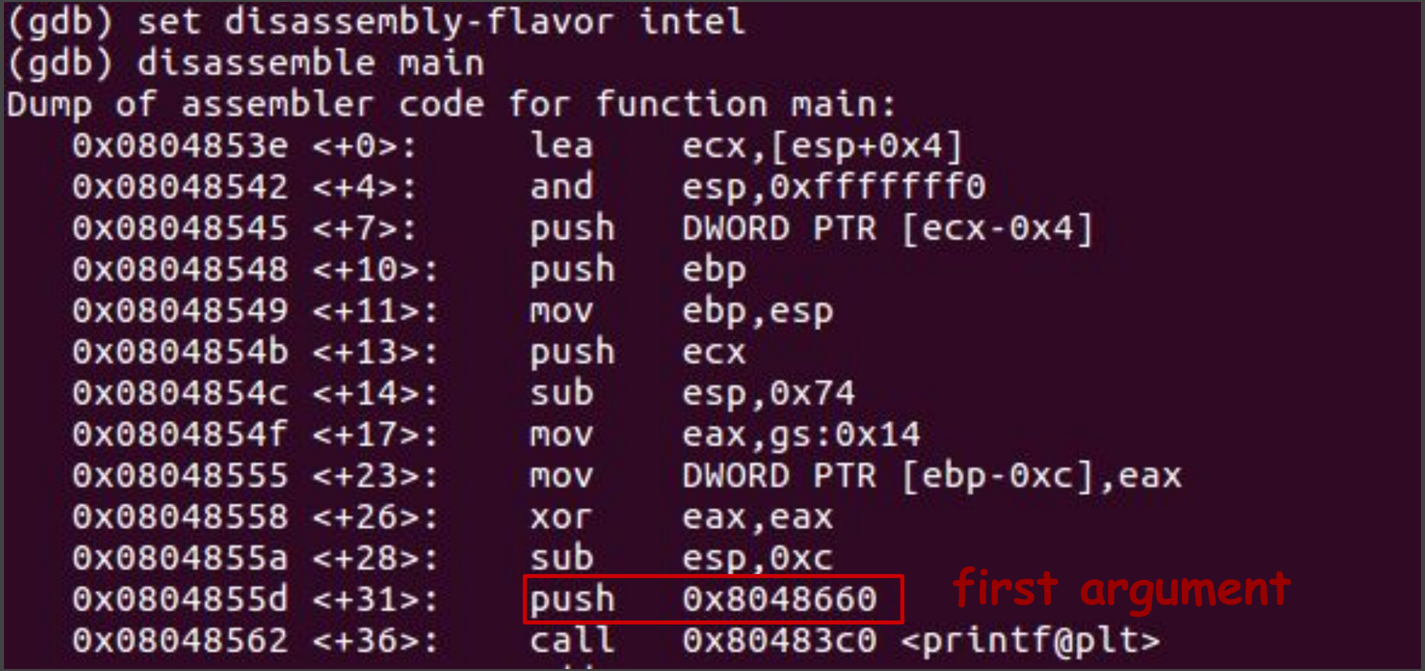
Getting started with GDB: Unstripped



```
(gdb) set disassembly-flavor intel
(gdb) disassemble main
Dump of assembler code for function main:
   0x0804853e <+0>:     lea    ecx,[esp+0x4]
   0x08048542 <+4>:     and    esp,0xfffffff0
   0x08048545 <+7>:     push   DWORD PTR [ecx-0x4]
   0x08048548 <+10>:    push   ebp
   0x08048549 <+11>:    mov    ebp,esp
   0x0804854b <+13>:    push   ecx
   0x0804854c <+14>:    sub    esp,0x74
   0x0804854f <+17>:    mov    eax,gs:0x14
   0x08048555 <+23>:    mov    DWORD PTR [ebp-0xc],eax
   0x08048558 <+26>:    xor    eax,eax
   0x0804855a <+28>:    sub    esp,0xc
   0x0804855d <+31>:    push   0x8048660
   0x08048562 <+36>:    call   0x80483c0 <printf@plt>
```

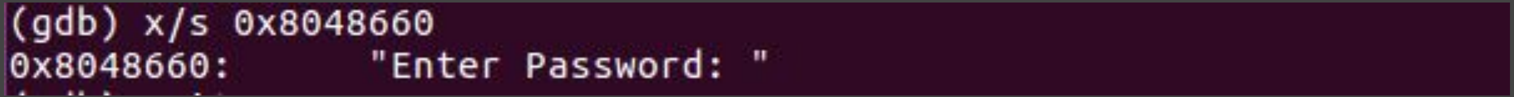
first argument

Getting started with GDB: Unstripped



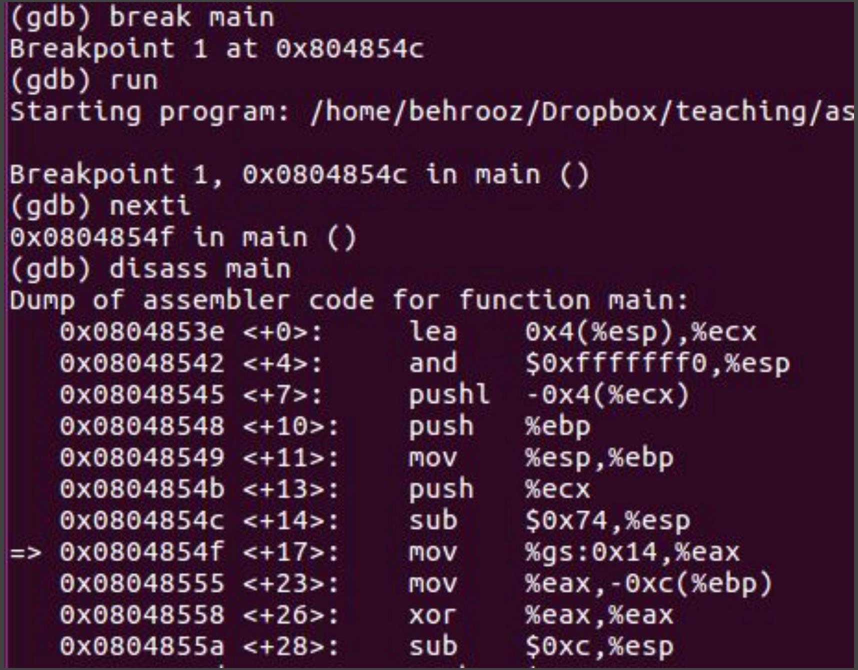
```
(gdb) set disassembly-flavor intel
(gdb) disassemble main
Dump of assembler code for function main:
   0x0804853e <+0>:     lea    ecx,[esp+0x4]
   0x08048542 <+4>:     and    esp,0xfffffff0
   0x08048545 <+7>:     push   DWORD PTR [ecx-0x4]
   0x08048548 <+10>:    push   ebp
   0x08048549 <+11>:    mov    ebp,esp
   0x0804854b <+13>:    push   ecx
   0x0804854c <+14>:    sub    esp,0x74
   0x0804854f <+17>:    mov    eax,gs:0x14
   0x08048555 <+23>:    mov    DWORD PTR [ebp-0xc],eax
   0x08048558 <+26>:    xor    eax,eax
   0x0804855a <+28>:    sub    esp,0xc
   0x0804855d <+31>:    push   0x8048660
   0x08048562 <+36>:    call   0x80483c0 <printf@plt>
```

first argument

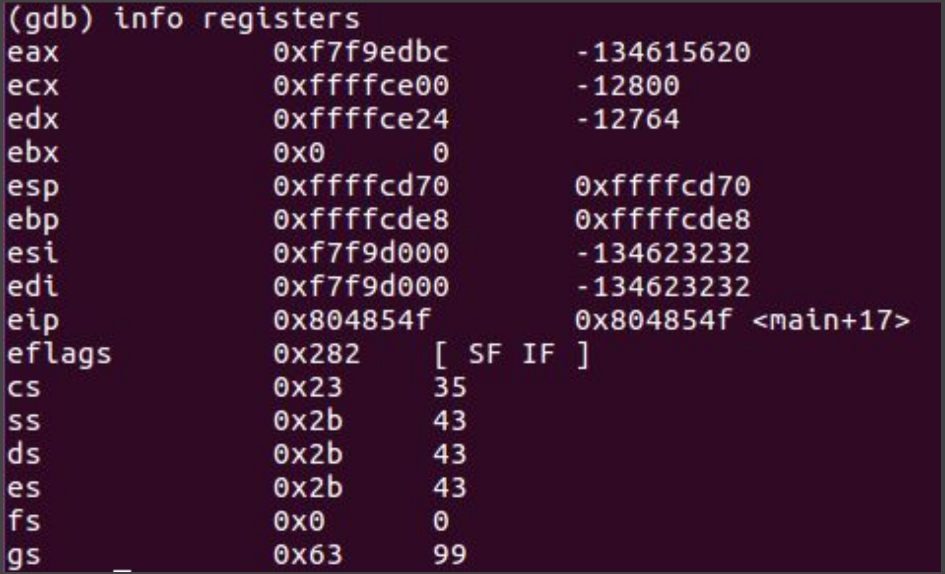


```
(gdb) x/s 0x8048660
0x8048660:      "Enter Password: "
```

Getting started with GDB: Unstripped

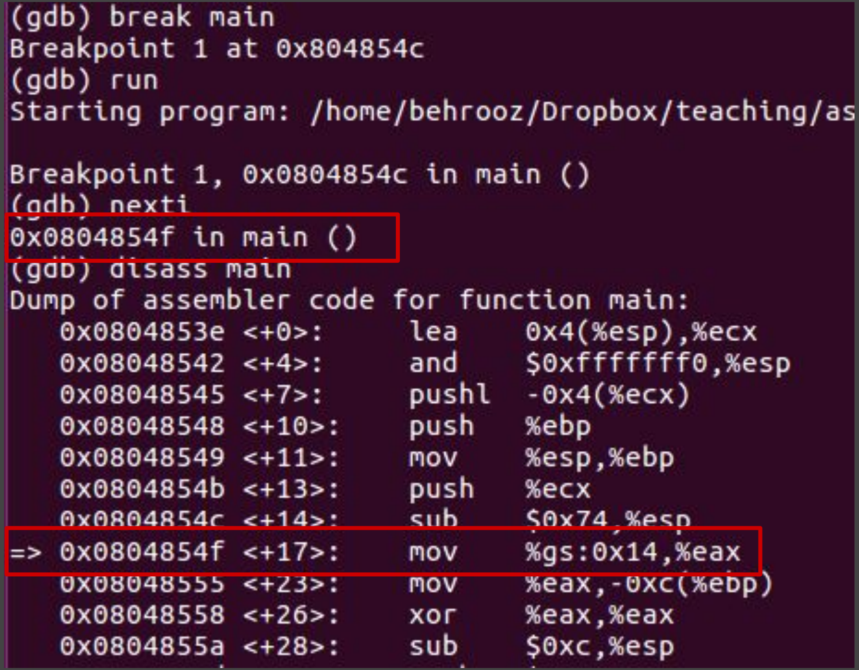


```
(gdb) break main
Breakpoint 1 at 0x804854c
(gdb) run
Starting program: /home/behrooz/Dropbox/teaching/as

Breakpoint 1, 0x0804854c in main ()
(gdb) nexti
0x0804854f in main ()
(gdb) disass main
Dump of assembler code for function main:
   0x0804853e <+0>:     lea    0x4(%esp),%ecx
   0x08048542 <+4>:     and    $0xfffffff0,%esp
   0x08048545 <+7>:     pushl  -0x4(%ecx)
   0x08048548 <+10>:    push   %ebp
   0x08048549 <+11>:    mov    %esp,%ebp
   0x0804854b <+13>:    push   %ecx
   0x0804854c <+14>:    sub    $0x74,%esp
=> 0x0804854f <+17>:    mov    %gs:0x14,%eax
   0x08048555 <+23>:    mov    %eax,-0xc(%ebp)
   0x08048558 <+26>:    xor    %eax,%eax
   0x0804855a <+28>:    sub    $0xc,%esp
```

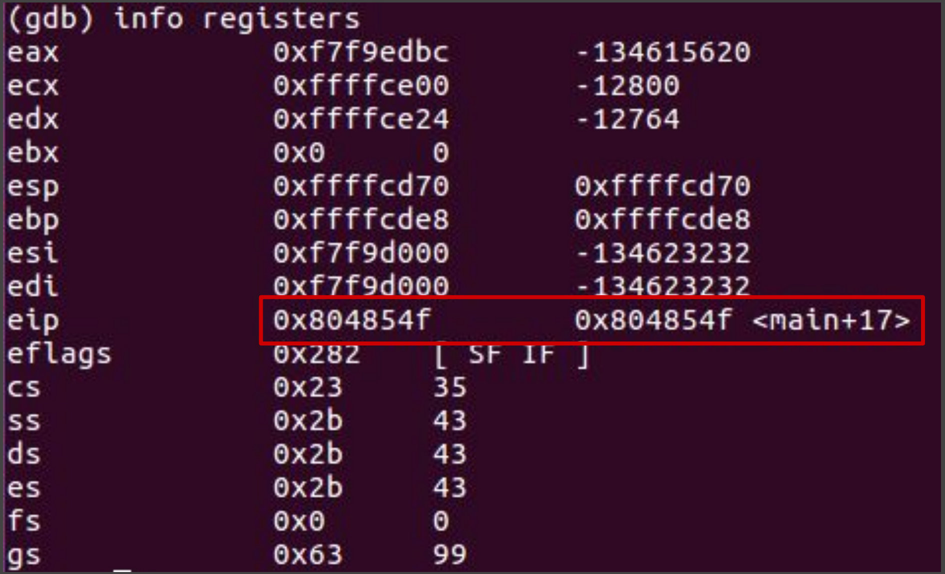


```
(gdb) info registers
eax            0xf7f9edbc       -134615620
ecx            0xffffce00       -12800
edx            0xffffce24       -12764
ebx            0x0      0
esp            0xffffcd70       0xffffcd70
ebp            0xffffcde8       0xffffcde8
esi            0xf7f9d000       -134623232
edi            0xf7f9d000       -134623232
eip            0x804854f        0x804854f <main+17>
eflags         0x282    [ SF IF ]
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x63     99
```

Getting started with GDB: Unstripped

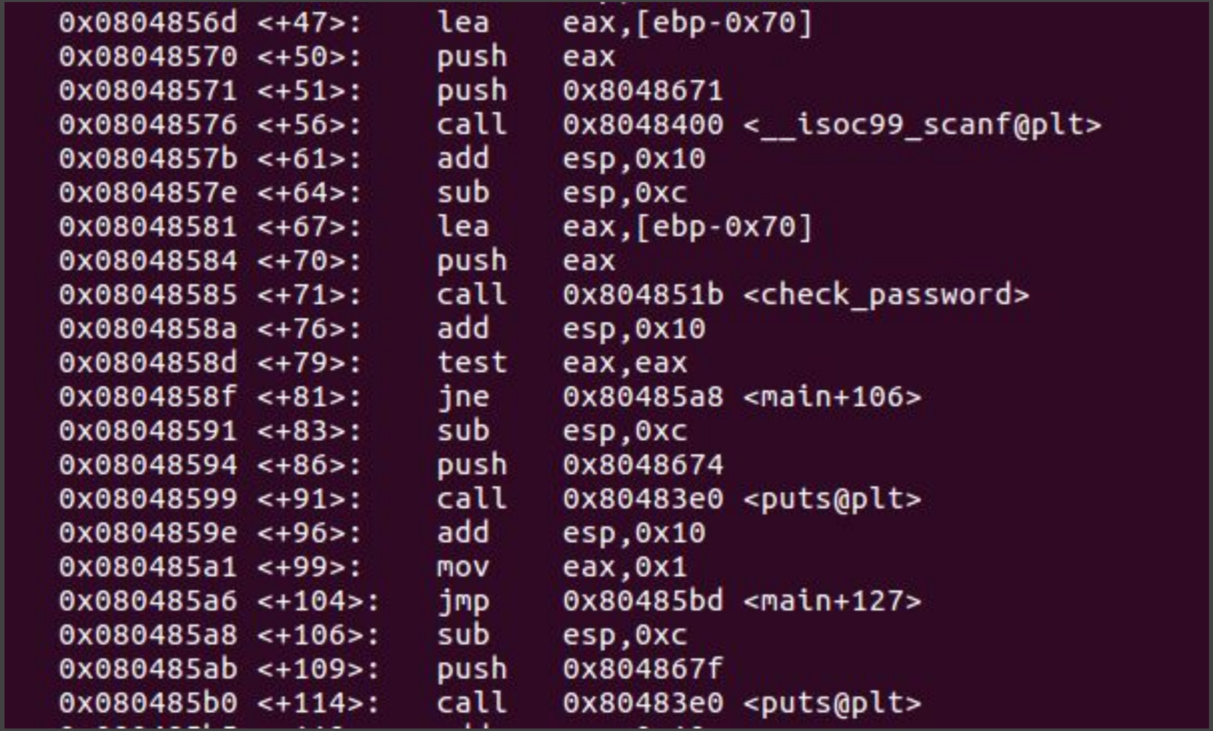


```
(gdb) break main
Breakpoint 1 at 0x804854c
(gdb) run
Starting program: /home/behrooz/Dropbox/teaching/as

Breakpoint 1, 0x0804854c in main ()
(gdb) nexti
0x0804854f in main ()
(gdb) disass main
Dump of assembler code for function main:
   0x0804853e <+0>:     lea    0x4(%esp),%ecx
   0x08048542 <+4>:     and    $0xfffffff0,%esp
   0x08048545 <+7>:     pushl  -0x4(%ecx)
   0x08048548 <+10>:    push   %ebp
   0x08048549 <+11>:    mov    %esp,%ebp
   0x0804854b <+13>:    push   %ecx
   0x0804854c <+14>:    sub    $0x74,%esp
=> 0x0804854f <+17>:    mov    %gs:0x14,%eax
   0x08048555 <+23>:    mov    %eax,-0xc(%ebp)
   0x08048558 <+26>:    xor    %eax,%eax
   0x0804855a <+28>:    sub    $0xc,%esp
```



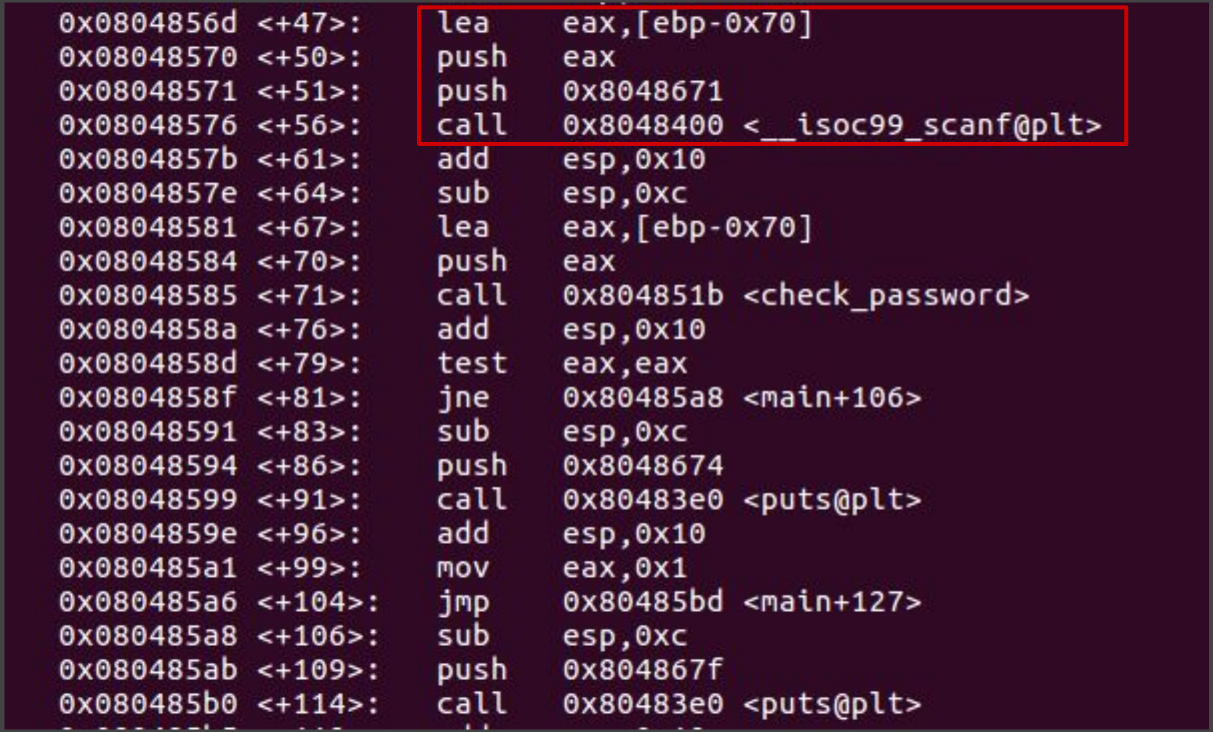
```
(gdb) info registers
eax            0xf7f9edbc       -134615620
ecx            0xffffce00       -12800
edx            0xffffce24       -12764
ebx            0x0      0
esp            0xffffcd70       0xffffcd70
ebp            0xffffcde8       0xffffcde8
esi            0xf7f9d000       -134623232
edi            0xf7f9d000       -134623232
eip            0x804854f        0x804854f <main+17>
eflags         0x282    [ SF IF ]
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x63     99
```

Getting started with GDB: Unstripped



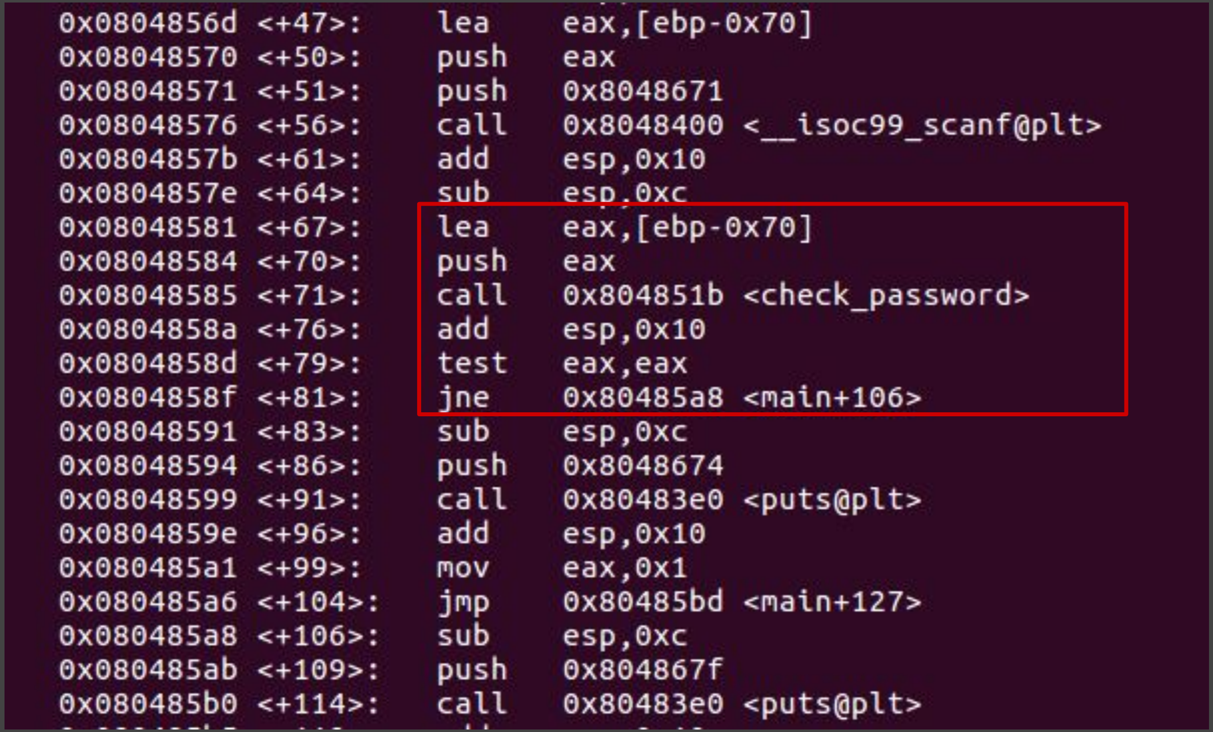
```
   0x0804856d <+47>:    lea    eax,[ebp-0x70]
   0x08048570 <+50>:    push   eax
   0x08048571 <+51>:    push   0x8048671
   0x08048576 <+56>:    call   0x8048400 <__isoc99_scanf@plt>
   0x0804857b <+61>:    add    esp,0x10
   0x0804857e <+64>:    sub    esp,0xc
   0x08048581 <+67>:    lea    eax,[ebp-0x70]
   0x08048584 <+70>:    push   eax
   0x08048585 <+71>:    call   0x804851b <check_password>
   0x0804858a <+76>:    add    esp,0x10
   0x0804858d <+79>:    test   eax,eax
   0x0804858f <+81>:    jne    0x80485a8 <main+106>
   0x08048591 <+83>:    sub    esp,0xc
   0x08048594 <+86>:    push   0x8048674
   0x08048599 <+91>:    call   0x80483e0 <puts@plt>
   0x0804859e <+96>:    add    esp,0x10
   0x080485a1 <+99>:    mov    eax,0x1
   0x080485a6 <+104>:   jmp    0x80485bd <main+127>
   0x080485a8 <+106>:   sub    esp,0xc
   0x080485ab <+109>:   push   0x804867f
   0x080485b0 <+114>:   call   0x80483e0 <puts@plt>
```

Getting started with GDB: Unstripped



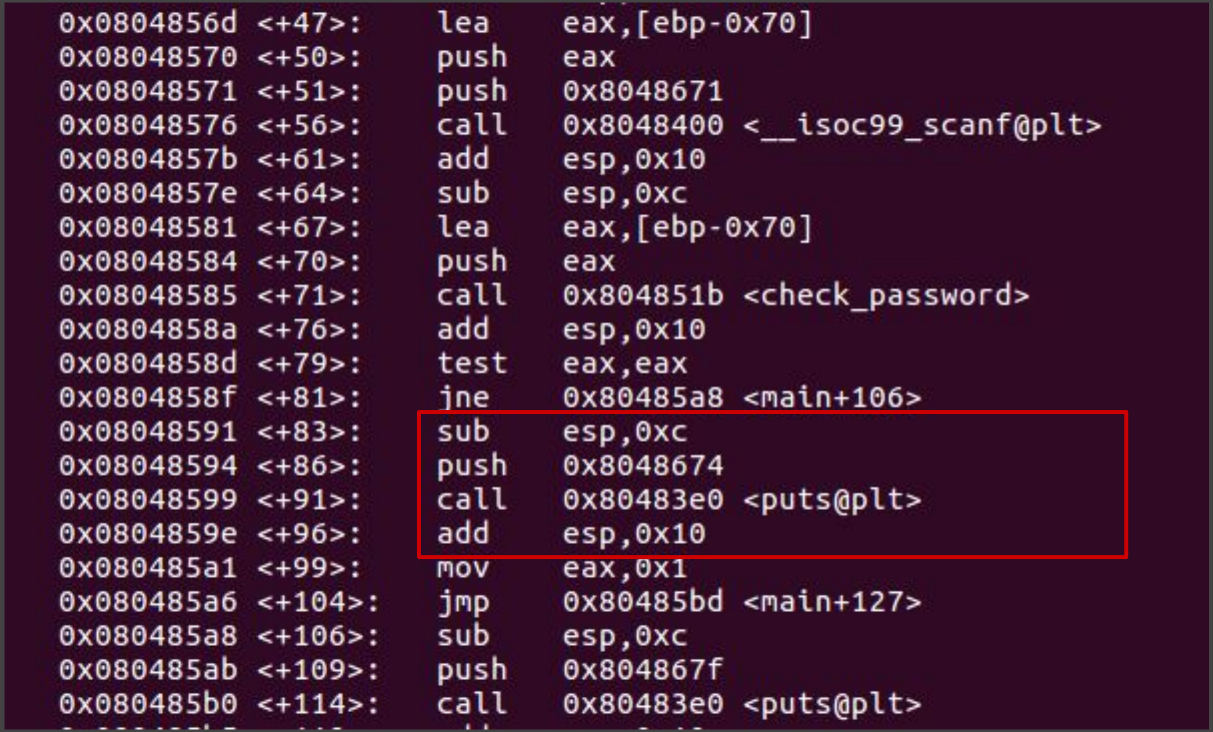
```
   0x0804856d <+47>:    lea    eax,[ebp-0x70]
   0x08048570 <+50>:    push   eax
   0x08048571 <+51>:    push   0x8048671
   0x08048576 <+56>:    call   0x8048400 <__isoc99_scanf@plt>
   0x0804857b <+61>:    add    esp,0x10
   0x0804857e <+64>:    sub    esp,0xc
   0x08048581 <+67>:    lea    eax,[ebp-0x70]
   0x08048584 <+70>:    push   eax
   0x08048585 <+71>:    call   0x804851b <check_password>
   0x0804858a <+76>:    add    esp,0x10
   0x0804858d <+79>:    test   eax,eax
   0x0804858f <+81>:    jne    0x80485a8 <main+106>
   0x08048591 <+83>:    sub    esp,0xc
   0x08048594 <+86>:    push   0x8048674
   0x08048599 <+91>:    call   0x80483e0 <puts@plt>
   0x0804859e <+96>:    add    esp,0x10
   0x080485a1 <+99>:    mov    eax,0x1
   0x080485a6 <+104>:   jmp    0x80485bd <main+127>
   0x080485a8 <+106>:   sub    esp,0xc
   0x080485ab <+109>:   push   0x804867f
   0x080485b0 <+114>:   call   0x80483e0 <puts@plt>
```

Getting started with GDB: Unstripped



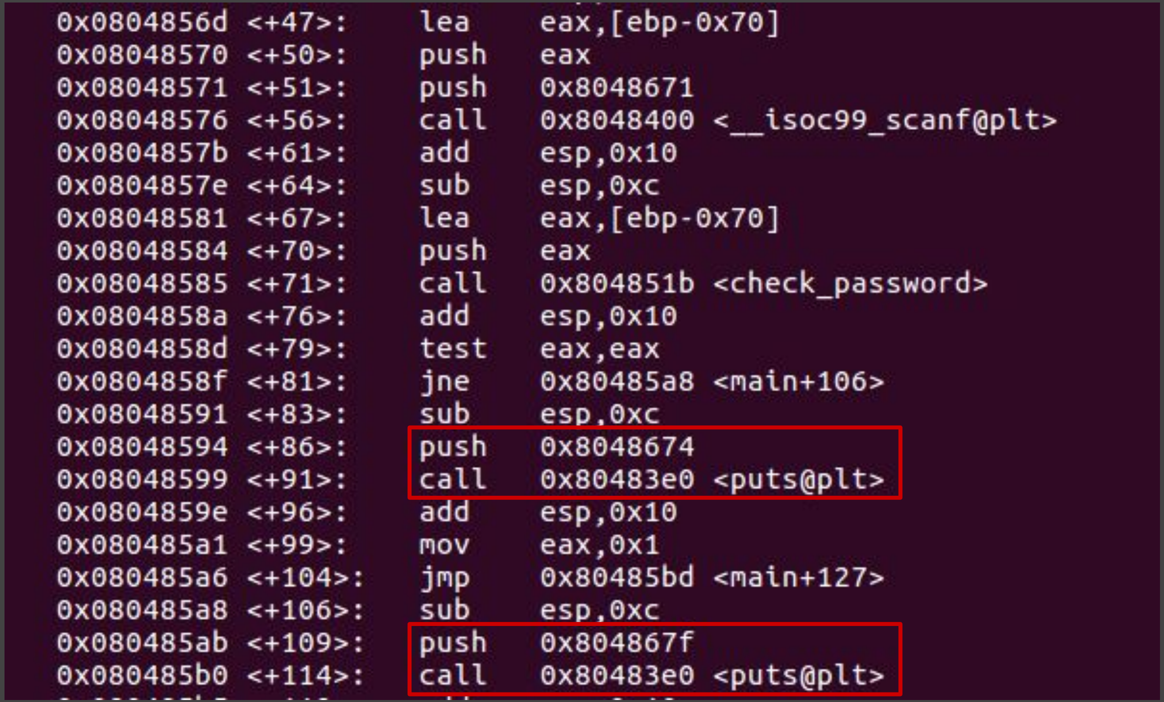
```
   0x0804856d <+47>:    lea    eax,[ebp-0x70]
   0x08048570 <+50>:    push   eax
   0x08048571 <+51>:    push   0x8048671
   0x08048576 <+56>:    call   0x8048400 <__isoc99_scanf@plt>
   0x0804857b <+61>:    add    esp,0x10
   0x0804857e <+64>:    sub    esp,0xc
   0x08048581 <+67>:    lea    eax,[ebp-0x70]
   0x08048584 <+70>:    push   eax
   0x08048585 <+71>:    call   0x804851b <check_password>
   0x0804858a <+76>:    add    esp,0x10
   0x0804858d <+79>:    test   eax,eax
   0x0804858f <+81>:    jne    0x80485a8 <main+106>
   0x08048591 <+83>:    sub    esp,0xc
   0x08048594 <+86>:    push   0x8048674
   0x08048599 <+91>:    call   0x80483e0 <puts@plt>
   0x0804859e <+96>:    add    esp,0x10
   0x080485a1 <+99>:    mov    eax,0x1
   0x080485a6 <+104>:   jmp    0x80485bd <main+127>
   0x080485a8 <+106>:   sub    esp,0xc
   0x080485ab <+109>:   push   0x804867f
   0x080485b0 <+114>:   call   0x80483e0 <puts@plt>
```

Getting started with GDB: Unstripped



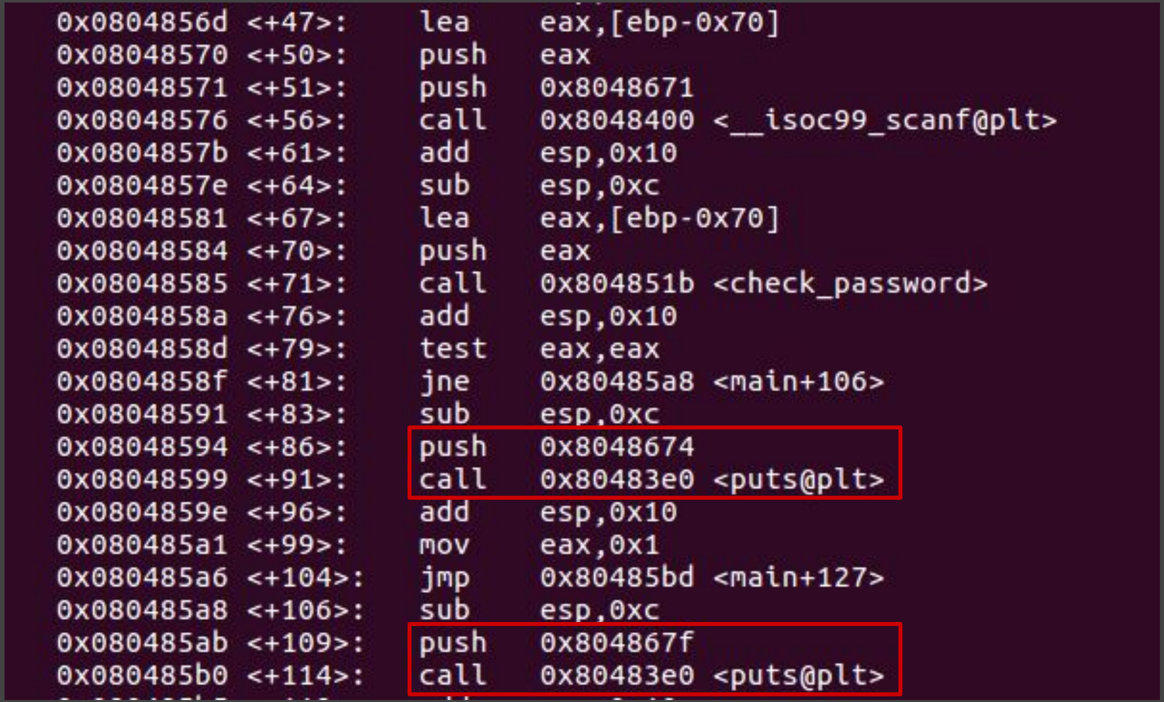
```
   0x0804856d <+47>:    lea    eax,[ebp-0x70]
   0x08048570 <+50>:    push   eax
   0x08048571 <+51>:    push   0x8048671
   0x08048576 <+56>:    call   0x8048400 <__isoc99_scanf@plt>
   0x0804857b <+61>:    add    esp,0x10
   0x0804857e <+64>:    sub    esp,0xc
   0x08048581 <+67>:    lea    eax,[ebp-0x70]
   0x08048584 <+70>:    push   eax
   0x08048585 <+71>:    call   0x804851b <check_password>
   0x0804858a <+76>:    add    esp,0x10
   0x0804858d <+79>:    test   eax,eax
   0x0804858f <+81>:    jne    0x80485a8 <main+106>
   0x08048591 <+83>:    sub    esp,0xc
   0x08048594 <+86>:    push   0x8048674
   0x08048599 <+91>:    call   0x80483e0 <puts@plt>
   0x0804859e <+96>:    add    esp,0x10
   0x080485a1 <+99>:    mov    eax,0x1
   0x080485a6 <+104>:   jmp    0x80485bd <main+127>
   0x080485a8 <+106>:   sub    esp,0xc
   0x080485ab <+109>:   push   0x804867f
   0x080485b0 <+114>:   call   0x80483e0 <puts@plt>
```

Getting started with GDB: Unstripped

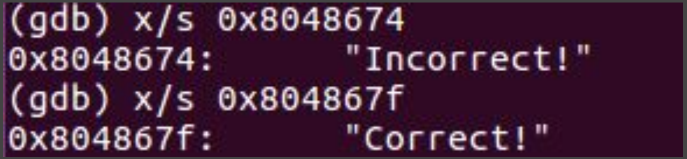


```
0x0804856d <+47>:    lea    eax,[ebp-0x70]
0x08048570 <+50>:    push   eax
0x08048571 <+51>:    push   0x8048671
0x08048576 <+56>:    call   0x8048400 <__isoc99_scanf@plt>
0x0804857b <+61>:    add    esp,0x10
0x0804857e <+64>:    sub    esp,0xc
0x08048581 <+67>:    lea    eax,[ebp-0x70]
0x08048584 <+70>:    push   eax
0x08048585 <+71>:    call   0x804851b <check_password>
0x0804858a <+76>:    add    esp,0x10
0x0804858d <+79>:    test   eax,eax
0x0804858f <+81>:    jne    0x80485a8 <main+106>
0x08048591 <+83>:    sub    esp,0xc
0x08048594 <+86>:    push   0x8048674
0x08048599 <+91>:    call   0x80483e0 <puts@plt>
0x0804859e <+96>:    add    esp,0x10
0x080485a1 <+99>:    mov    eax,0x1
0x080485a6 <+104>:   jmp    0x80485bd <main+127>
0x080485a8 <+106>:   sub    esp,0xc
0x080485ab <+109>:   push   0x804867f
0x080485b0 <+114>:   call   0x80483e0 <puts@plt>
```

Getting started with GDB: Unstripped



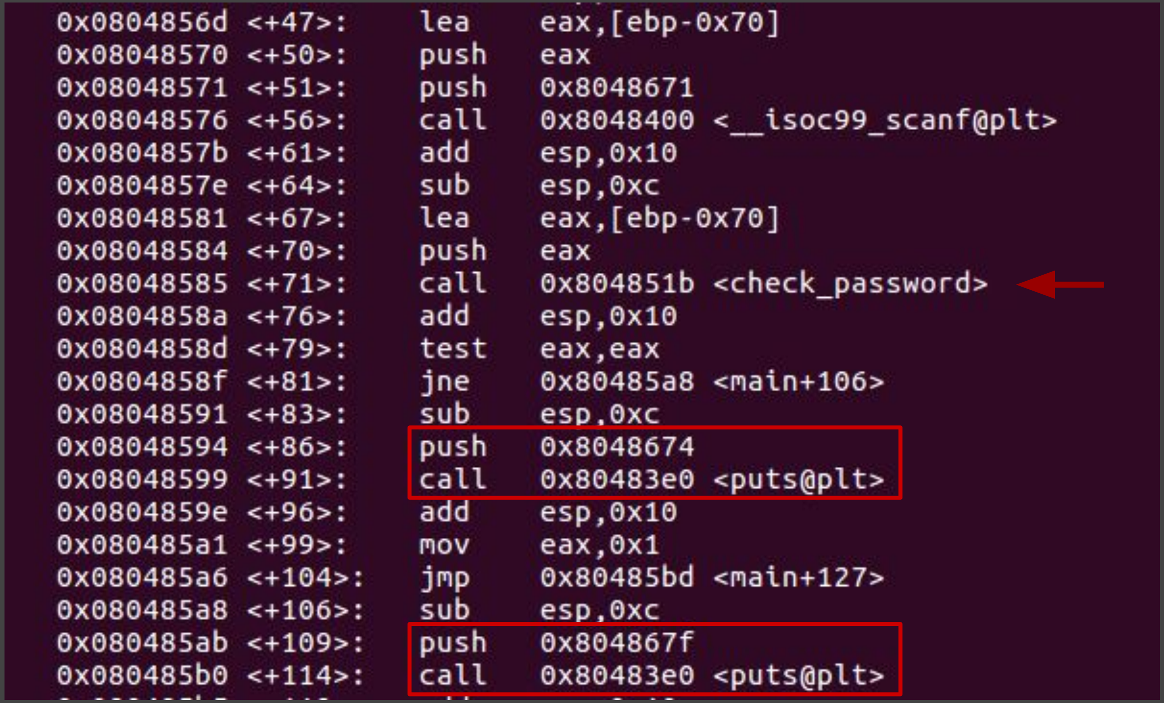
```
   0x0804856d <+47>:    lea    eax,[ebp-0x70]
   0x08048570 <+50>:    push   eax
   0x08048571 <+51>:    push   0x8048671
   0x08048576 <+56>:    call   0x8048400 <__isoc99_scanf@plt>
   0x0804857b <+61>:    add    esp,0x10
   0x0804857e <+64>:    sub    esp,0xc
   0x08048581 <+67>:    lea    eax,[ebp-0x70]
   0x08048584 <+70>:    push   eax
   0x08048585 <+71>:    call   0x804851b <check_password>
   0x0804858a <+76>:    add    esp,0x10
   0x0804858d <+79>:    test   eax,eax
   0x0804858f <+81>:    jne    0x80485a8 <main+106>
   0x08048591 <+83>:    sub    esp,0xc
   0x08048594 <+86>:    push   0x8048674
   0x08048599 <+91>:    call   0x80483e0 <puts@plt>
   0x0804859e <+96>:    add    esp,0x10
   0x080485a1 <+99>:    mov    eax,0x1
   0x080485a6 <+104>:   jmp    0x80485bd <main+127>
   0x080485a8 <+106>:   sub    esp,0xc
   0x080485ab <+109>:   push   0x804867f
   0x080485b0 <+114>:   call   0x80483e0 <puts@plt>
```



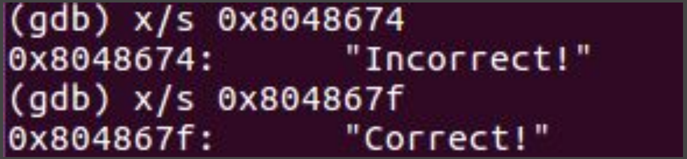
```
(gdb) x/s 0x8048674
0x8048674:      "Incorrect!"
(gdb) x/s 0x804867f
0x804867f:      "Correct!"
```

Getting started with GDB: Unstripped

K. N. Toosi
University of Technology

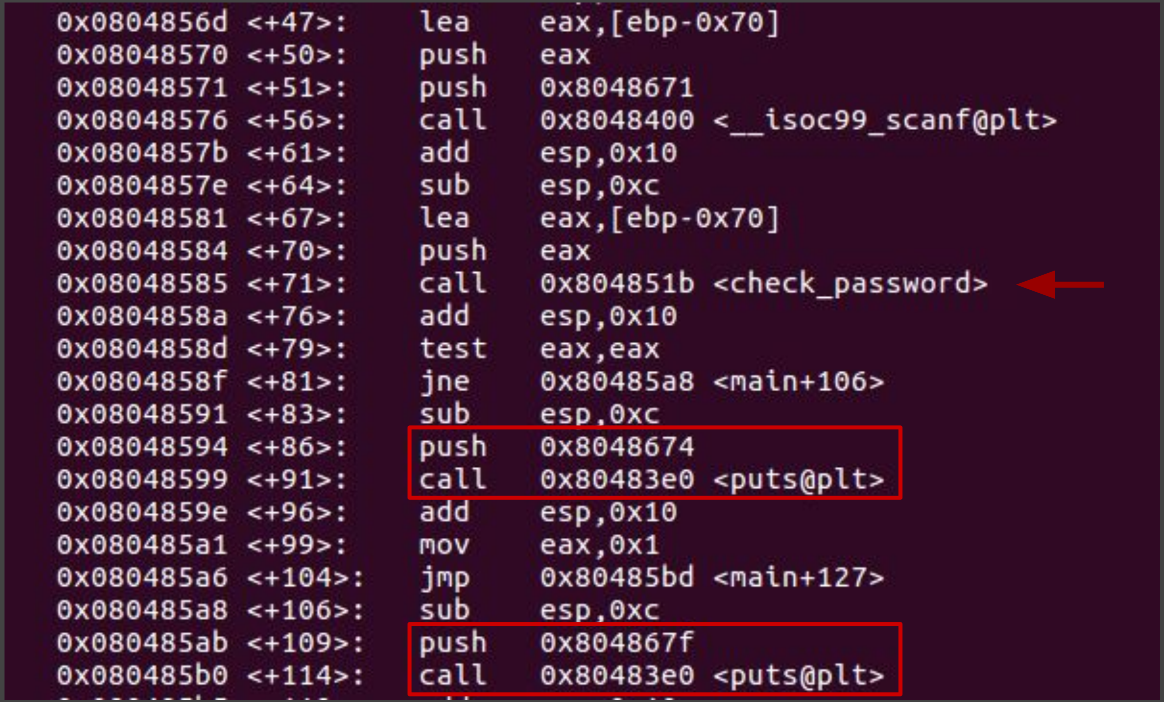


```
   0x0804856d <+47>:    lea    eax,[ebp-0x70]
   0x08048570 <+50>:    push   eax
   0x08048571 <+51>:    push   0x8048671
   0x08048576 <+56>:    call   0x8048400 <__isoc99_scanf@plt>
   0x0804857b <+61>:    add    esp,0x10
   0x0804857e <+64>:    sub    esp,0xc
   0x08048581 <+67>:    lea    eax,[ebp-0x70]
   0x08048584 <+70>:    push   eax
   0x08048585 <+71>:    call   0x804851b <check_password>
   0x0804858a <+76>:    add    esp,0x10
   0x0804858d <+79>:    test   eax,eax
   0x0804858f <+81>:    jne    0x80485a8 <main+106>
   0x08048591 <+83>:    sub    esp,0xc
   0x08048594 <+86>:    push   0x8048674
   0x08048599 <+91>:    call   0x80483e0 <puts@plt>
   0x0804859e <+96>:    add    esp,0x10
   0x080485a1 <+99>:    mov    eax,0x1
   0x080485a6 <+104>:   jmp    0x80485bd <main+127>
   0x080485a8 <+106>:   sub    esp,0xc
   0x080485ab <+109>:   push   0x804867f
   0x080485b0 <+114>:   call   0x80483e0 <puts@plt>
```

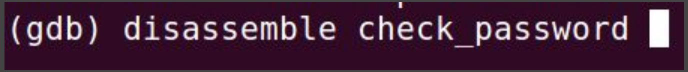


```
(gdb) x/s 0x8048674
0x8048674:       "Incorrect!"
(gdb) x/s 0x804867f
0x804867f:       "Correct!"
```

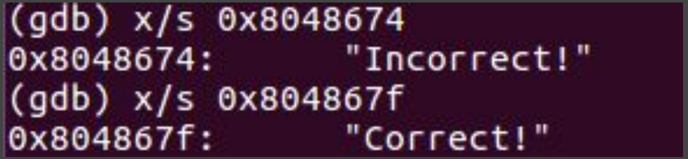
Getting started with GDB: Unstripped



```
   0x0804856d <+47>:    lea    eax,[ebp-0x70]
   0x08048570 <+50>:    push   eax
   0x08048571 <+51>:    push   0x8048671
   0x08048576 <+56>:    call   0x8048400 <__isoc99_scanf@plt>
   0x0804857b <+61>:    add    esp,0x10
   0x0804857e <+64>:    sub    esp,0xc
   0x08048581 <+67>:    lea    eax,[ebp-0x70]
   0x08048584 <+70>:    push   eax
   0x08048585 <+71>:    call   0x804851b <check_password>
   0x0804858a <+76>:    add    esp,0x10
   0x0804858d <+79>:    test   eax,eax
   0x0804858f <+81>:    jne    0x80485a8 <main+106>
   0x08048591 <+83>:    sub    esp,0xc
   0x08048594 <+86>:    push   0x8048674
   0x08048599 <+91>:    call   0x80483e0 <puts@plt>
   0x0804859e <+96>:    add    esp,0x10
   0x080485a1 <+99>:    mov    eax,0x1
   0x080485a6 <+104>:   jmp    0x80485bd <main+127>
   0x080485a8 <+106>:   sub    esp,0xc
   0x080485ab <+109>:   push   0x804867f
   0x080485b0 <+114>:   call   0x80483e0 <puts@plt>
```

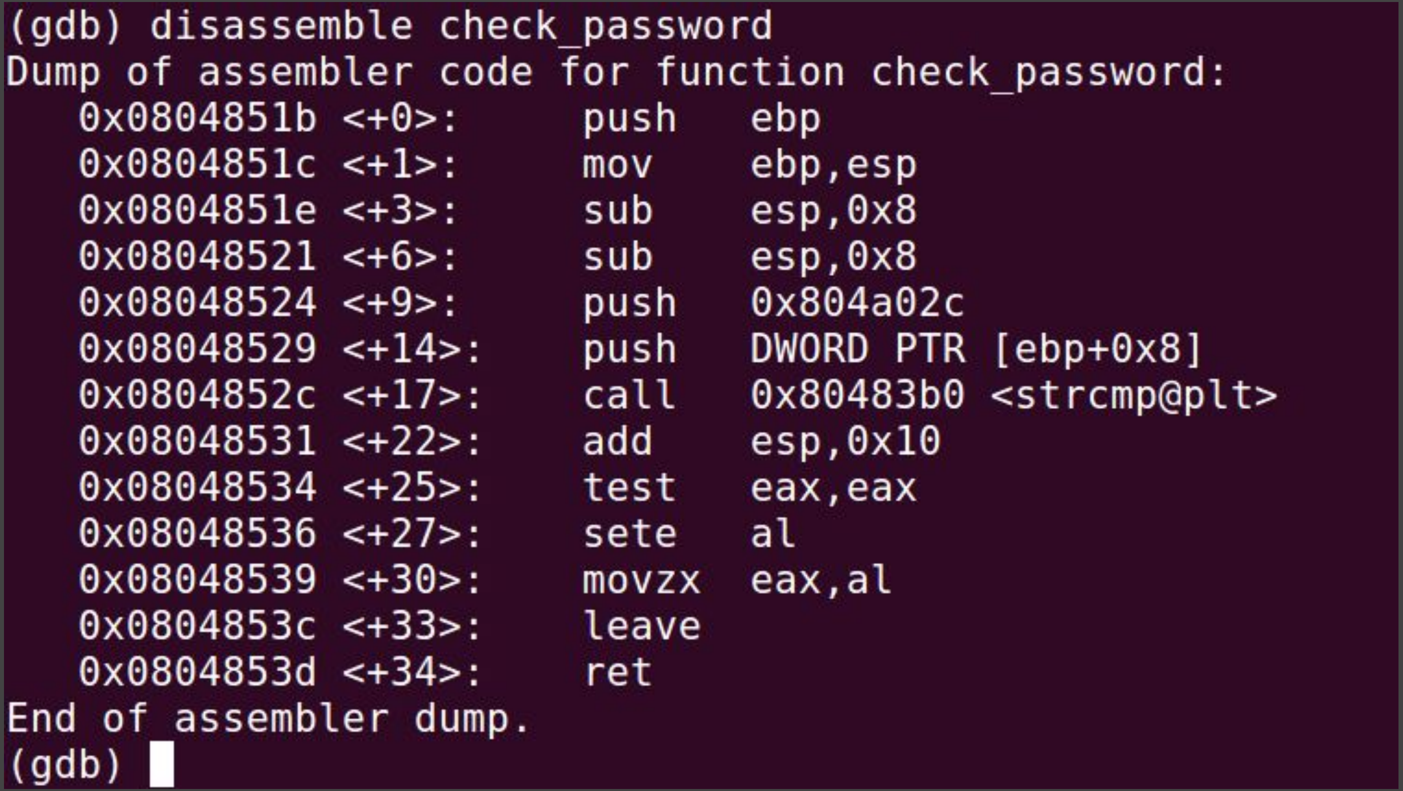


```
(gdb) disassemble check_password
```



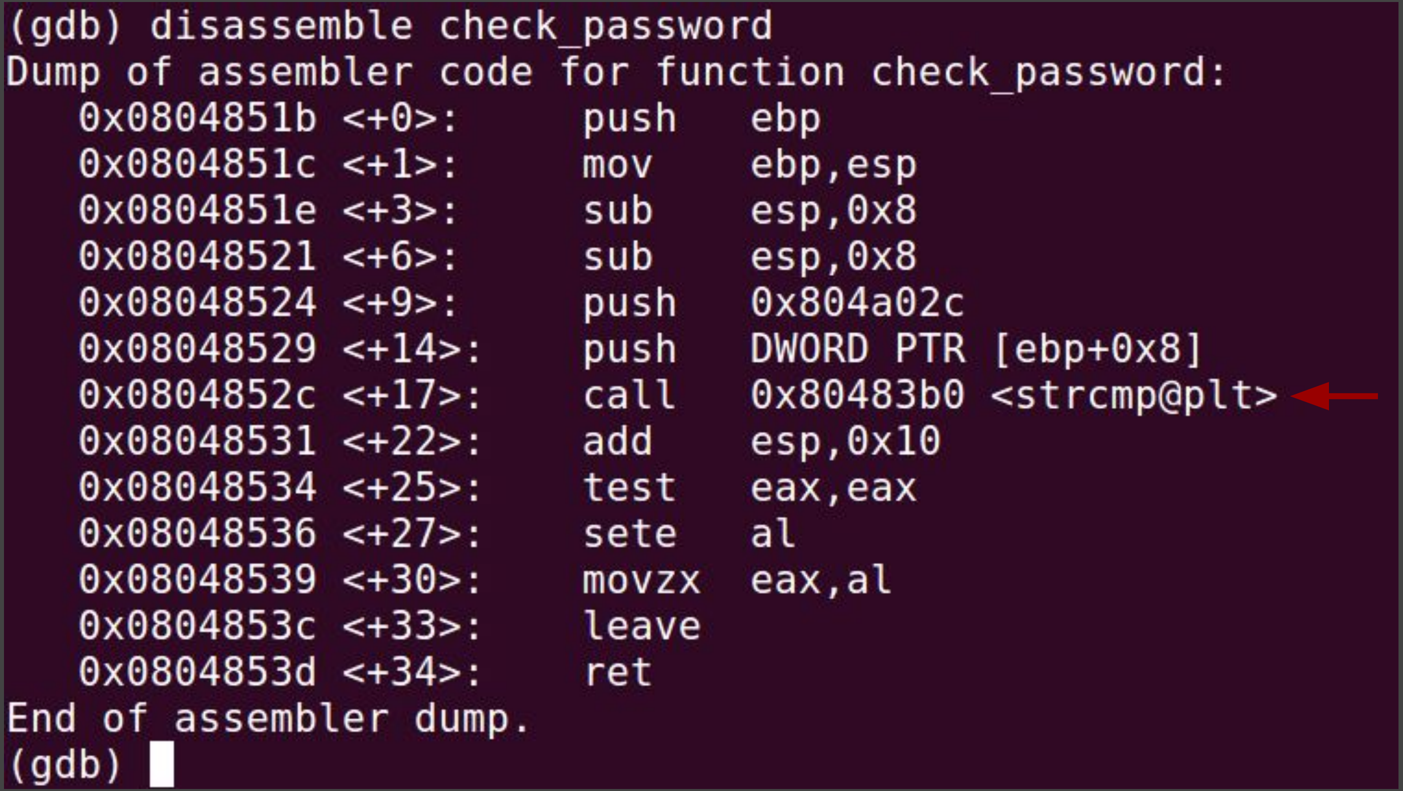
```
(gdb) x/s 0x8048674
0x8048674:      "Incorrect!"
(gdb) x/s 0x804867f
0x804867f:      "Correct!"
```

Getting started with GDB: Unstripped



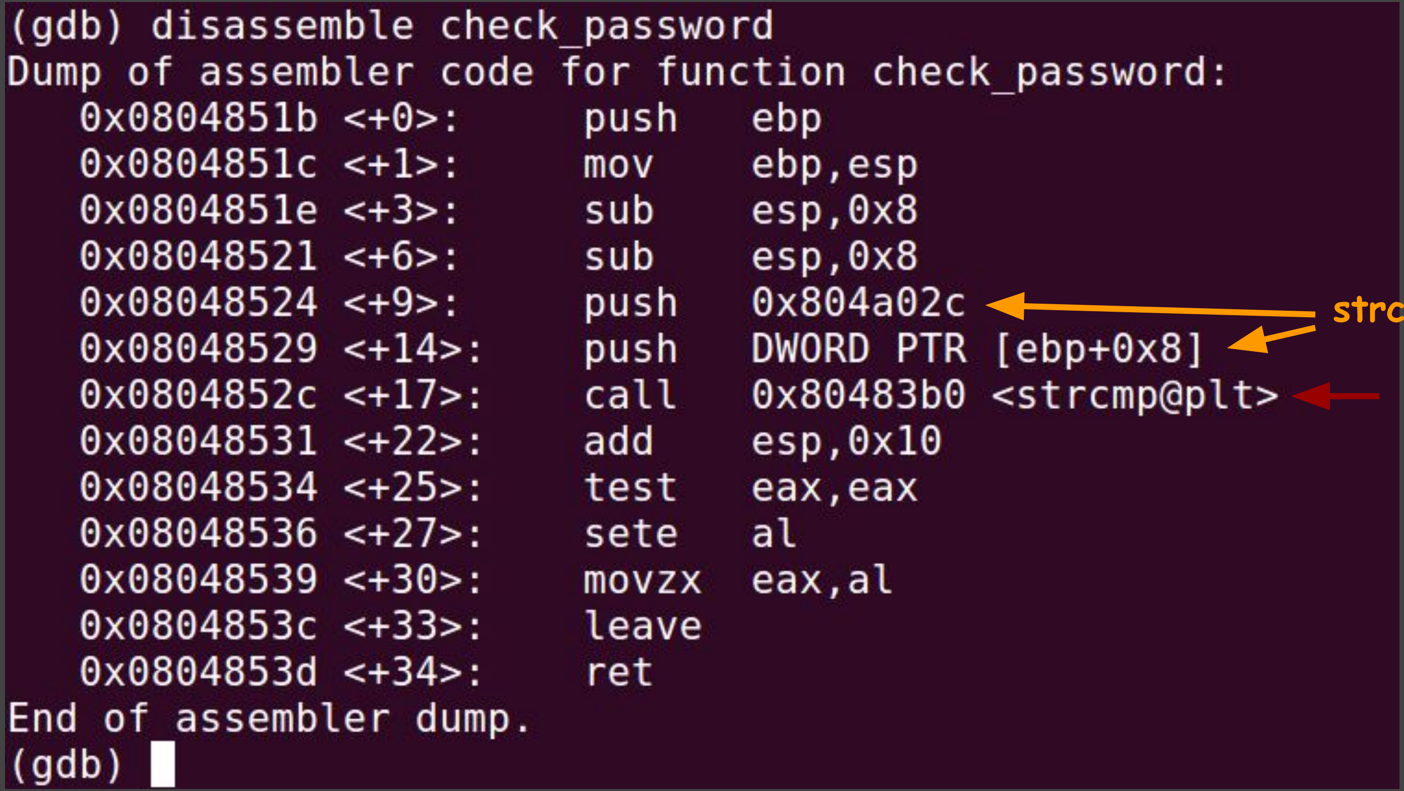
```
(gdb) disassemble check_password
Dump of assembler code for function check_password:
   0x0804851b <+0>:     push   ebp
   0x0804851c <+1>:     mov    ebp,esp
   0x0804851e <+3>:     sub    esp,0x8
   0x08048521 <+6>:     sub    esp,0x8
   0x08048524 <+9>:     push   0x804a02c
   0x08048529 <+14>:    push   DWORD PTR [ebp+0x8]
   0x0804852c <+17>:    call   0x80483b0 <strcmp@plt>
   0x08048531 <+22>:    add    esp,0x10
   0x08048534 <+25>:    test   eax,eax
   0x08048536 <+27>:    sete   al
   0x08048539 <+30>:    movzx  eax,al
   0x0804853c <+33>:    leave
   0x0804853d <+34>:    ret
End of assembler dump.
(gdb)
```

Getting started with GDB: Unstripped



```
(gdb) disassemble check_password
Dump of assembler code for function check_password:
   0x0804851b <+0>:     push   ebp
   0x0804851c <+1>:     mov    ebp,esp
   0x0804851e <+3>:     sub    esp,0x8
   0x08048521 <+6>:     sub    esp,0x8
   0x08048524 <+9>:     push   0x804a02c
   0x08048529 <+14>:    push   DWORD PTR [ebp+0x8]
   0x0804852c <+17>:    call   0x80483b0 <strcmp@plt>
   0x08048531 <+22>:    add    esp,0x10
   0x08048534 <+25>:    test   eax,eax
   0x08048536 <+27>:    sete   al
   0x08048539 <+30>:    movzx  eax,al
   0x0804853c <+33>:    leave
   0x0804853d <+34>:    ret
End of assembler dump.
(gdb)
```

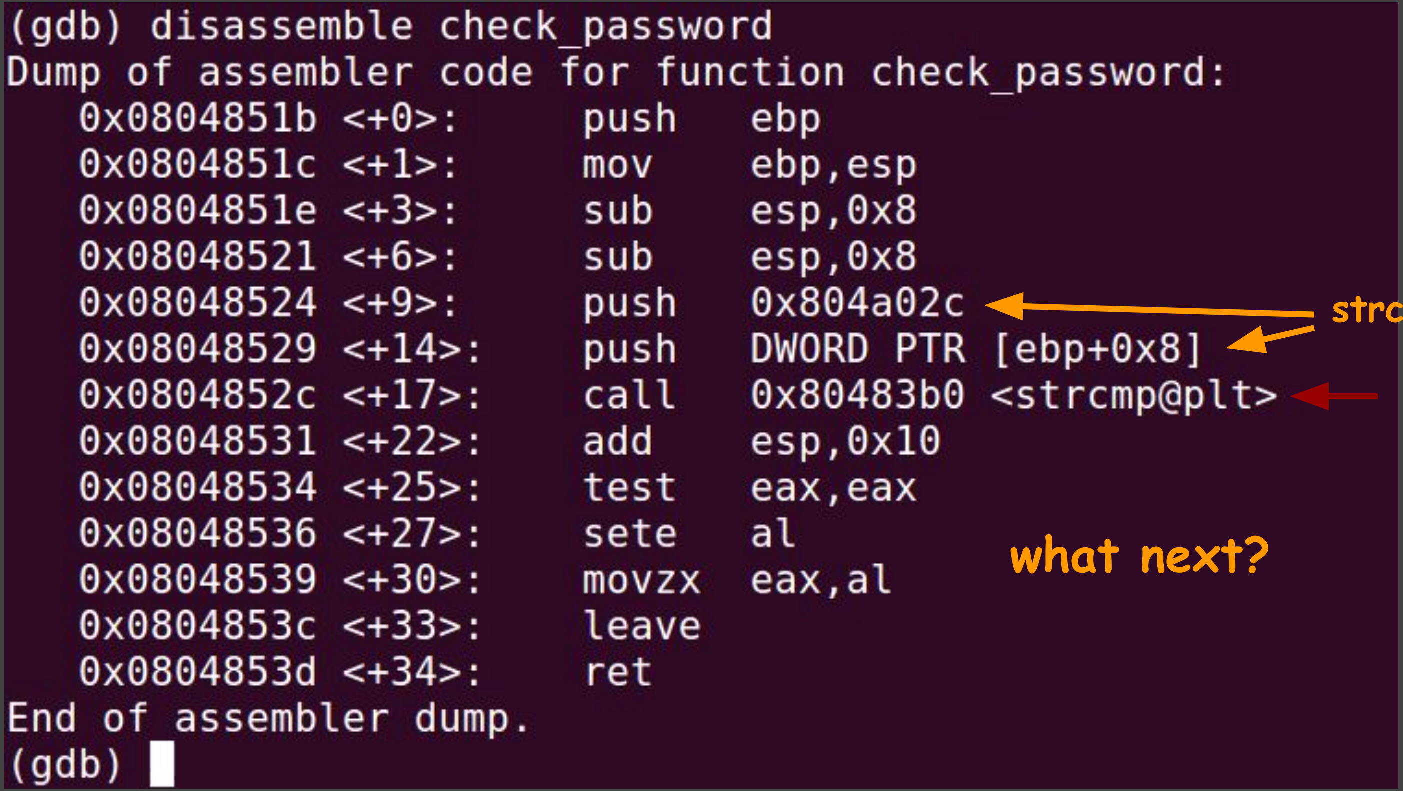
Getting started with GDB: Unstripped



```
(gdb) disassemble check_password
Dump of assembler code for function check_password:
   0x0804851b <+0>:     push   ebp
   0x0804851c <+1>:     mov    ebp,esp
   0x0804851e <+3>:     sub    esp,0x8
   0x08048521 <+6>:     sub    esp,0x8
   0x08048524 <+9>:     push   0x804a02c
   0x08048529 <+14>:    push   DWORD PTR [ebp+0x8]
   0x0804852c <+17>:    call   0x80483b0 <strcmp@plt>
   0x08048531 <+22>:    add    esp,0x10
   0x08048534 <+25>:    test   eax,eax
   0x08048536 <+27>:    sete   al
   0x08048539 <+30>:    movzx  eax,al
   0x0804853c <+33>:    leave
   0x0804853d <+34>:    ret
End of assembler dump.
(gdb)
```

strcmp arguments

Getting started with GDB: Unstripped

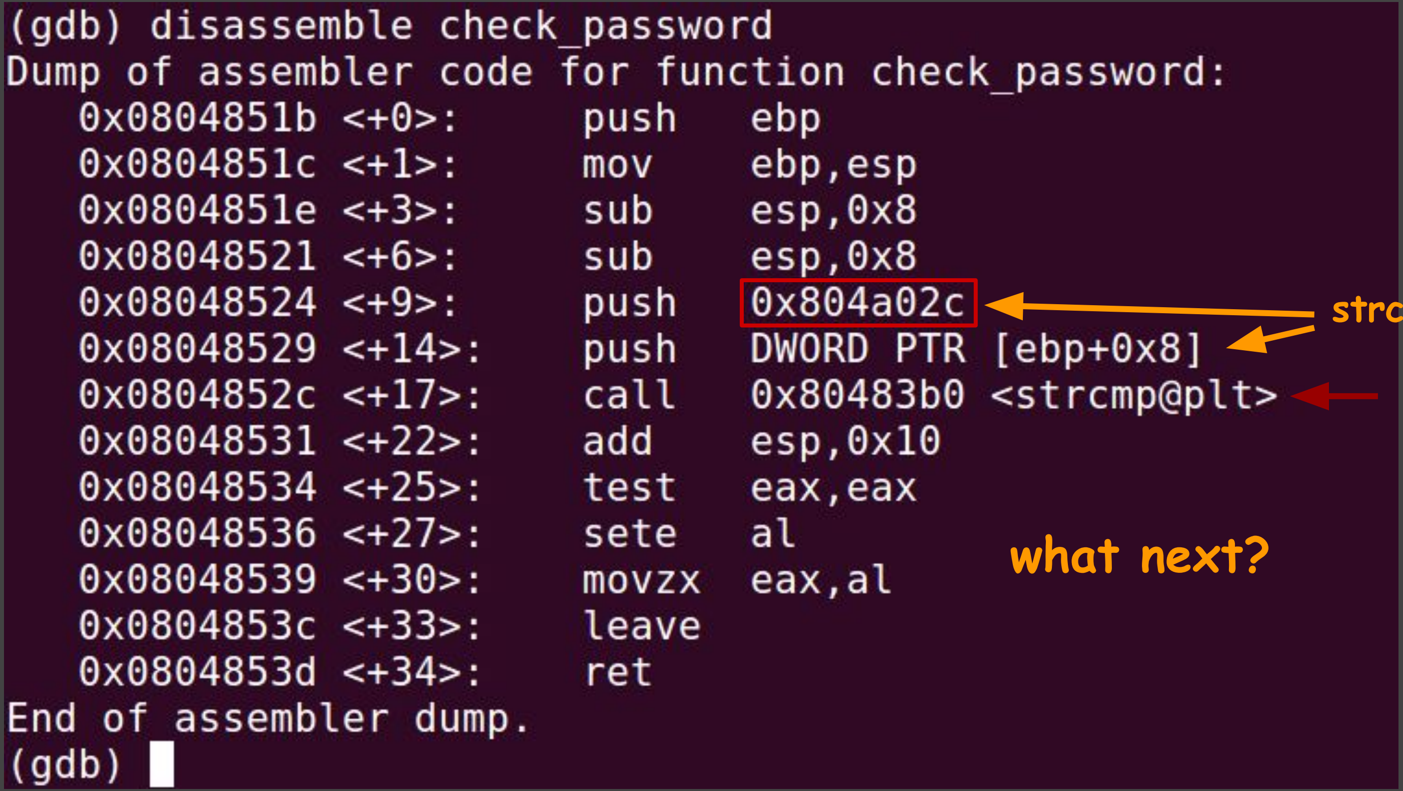


```
(gdb) disassemble check_password
Dump of assembler code for function check_password:
   0x0804851b <+0>:     push   ebp
   0x0804851c <+1>:     mov    ebp,esp
   0x0804851e <+3>:     sub    esp,0x8
   0x08048521 <+6>:     sub    esp,0x8
   0x08048524 <+9>:     push   0x804a02c
   0x08048529 <+14>:    push   DWORD PTR [ebp+0x8]
   0x0804852c <+17>:    call   0x80483b0 <strcmp@plt>
   0x08048531 <+22>:    add    esp,0x10
   0x08048534 <+25>:    test   eax,eax
   0x08048536 <+27>:    sete   al
   0x08048539 <+30>:    movzx  eax,al
   0x0804853c <+33>:    leave
   0x0804853d <+34>:    ret
End of assembler dump.
(gdb)
```

strcmp arguments

what next?

Getting started with GDB: Unstripped



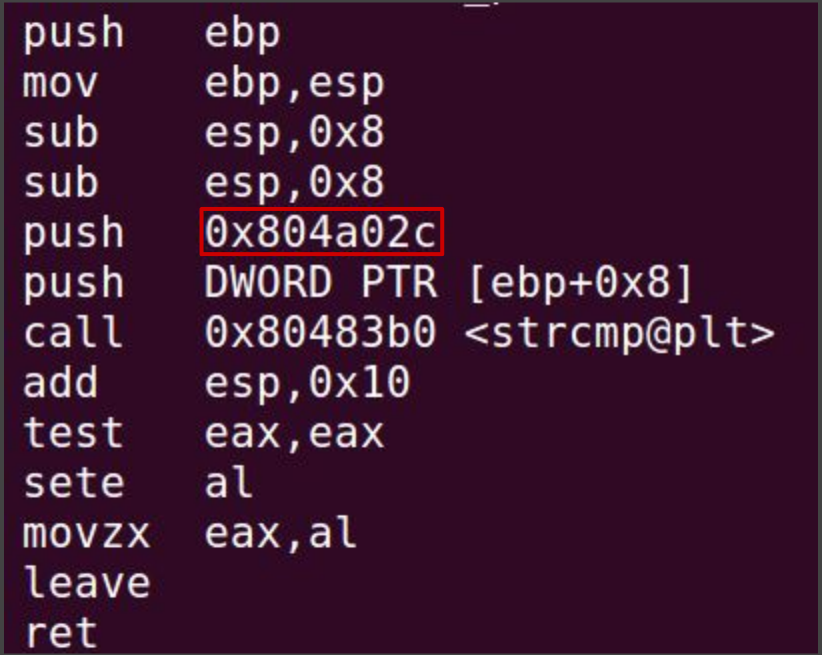
```
(gdb) disassemble check_password
Dump of assembler code for function check_password:
   0x0804851b <+0>:     push   ebp
   0x0804851c <+1>:     mov    ebp,esp
   0x0804851e <+3>:     sub    esp,0x8
   0x08048521 <+6>:     sub    esp,0x8
   0x08048524 <+9>:     push   0x804a02c
   0x08048529 <+14>:    push   DWORD PTR [ebp+0x8]
   0x0804852c <+17>:    call   0x80483b0 <strcmp@plt>
   0x08048531 <+22>:    add    esp,0x10
   0x08048534 <+25>:    test   eax,eax
   0x08048536 <+27>:    sete   al
   0x08048539 <+30>:    movzx  eax,al
   0x0804853c <+33>:    leave
   0x0804853d <+34>:    ret
End of assembler dump.
(gdb)
```

strcmp arguments

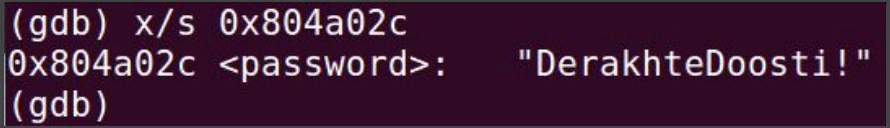
what next?

Getting started with GDB: Unstripped

K. N. Toosi
University of Technology

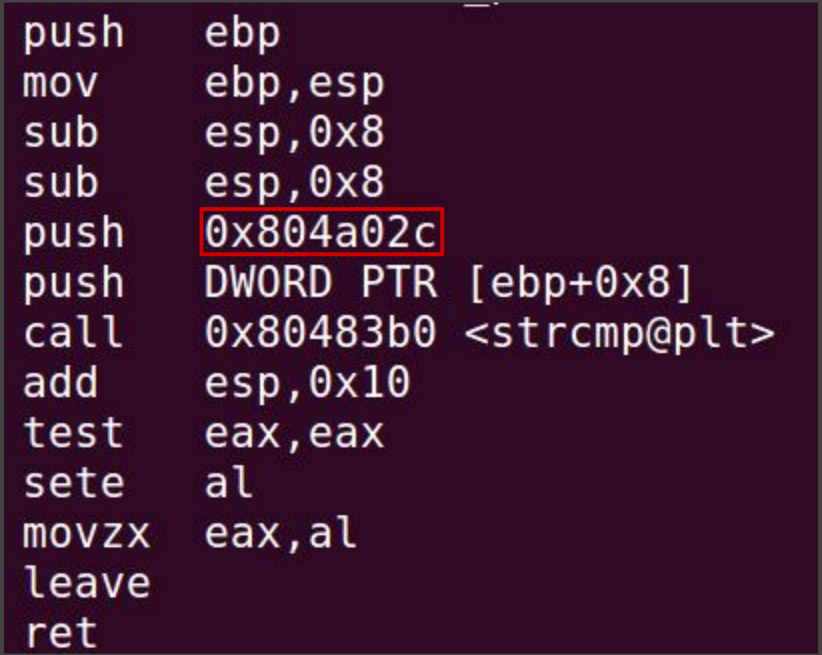


```
push    ebp
mov     ebp,esp
sub     esp,0x8
sub     esp,0x8
push    0x804a02c
push    DWORD PTR [ebp+0x8]
call    0x80483b0 <strcmp@plt>
add     esp,0x10
test    eax,eax
sete    al
movzx   eax,al
leave
ret
```

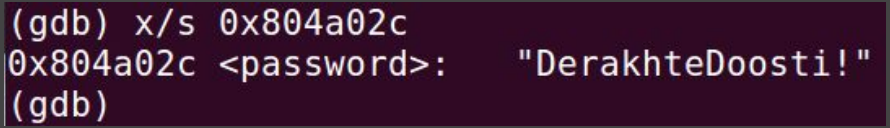


```
(gdb) x/s 0x804a02c
0x804a02c <password>:   "DerakhteDoosti!"
(gdb)
```

Getting started with GDB: Unstripped



```
push    ebp
mov     ebp,esp
sub     esp,0x8
sub     esp,0x8
push    0x804a02c
push    DWORD PTR [ebp+0x8]
call    0x80483b0 <strcmp@plt>
add     esp,0x10
test    eax,eax
sete    al
movzx   eax,al
leave
ret
```



```
(gdb) x/s 0x804a02c
0x804a02c <password>:   "DerakhteDoosti!"
(gdb)
```

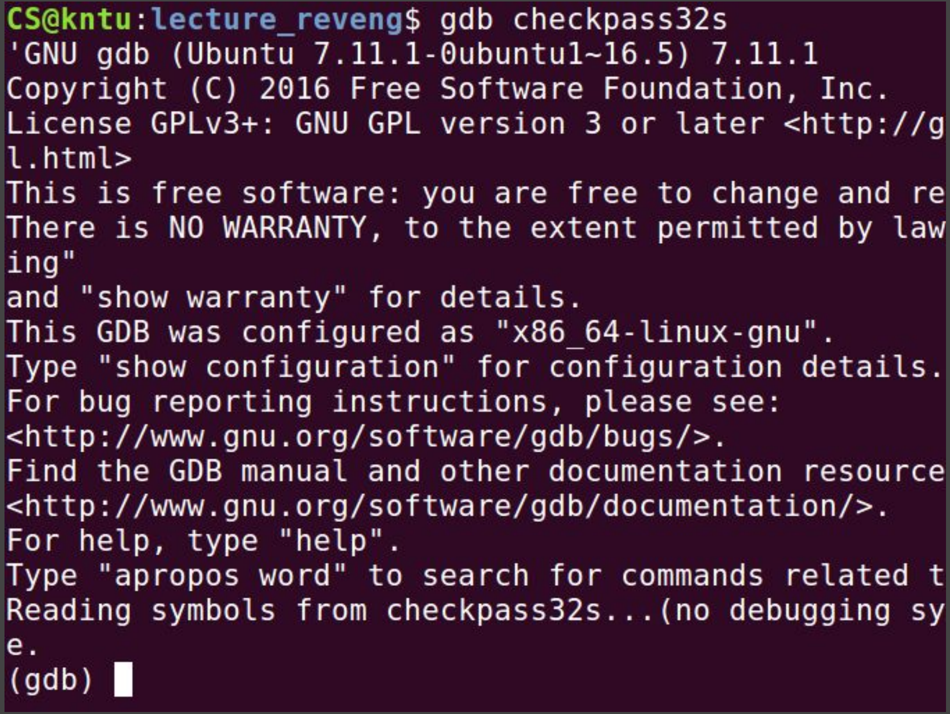
what about
stripped programs?

Stripped programs

- Where to start?
- Where to look?

Stripped programs

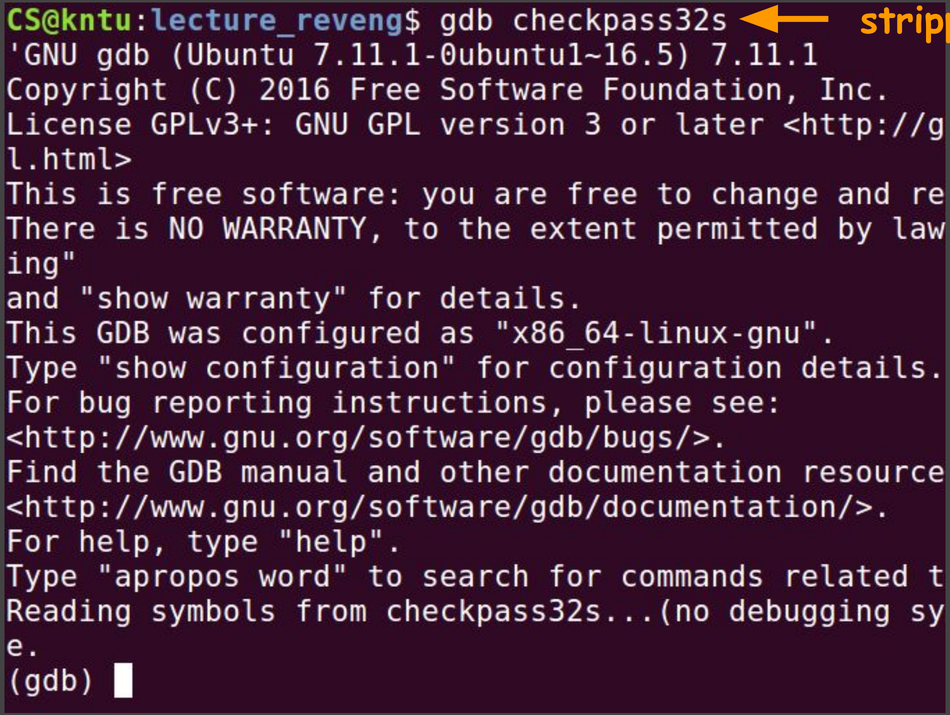
- Where to start?
- Where to look?



```
CS@kntu:lecture_reveng$ gdb checkpass32s
'GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://g
l.html>
This is free software: you are free to change and re
There is NO WARRANTY, to the extent permitted by law
ing"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resource
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related t
Reading symbols from checkpass32s...(no debugging sy
e.
(gdb) █
```

Stripped programs

- Where to start?
- Where to look?

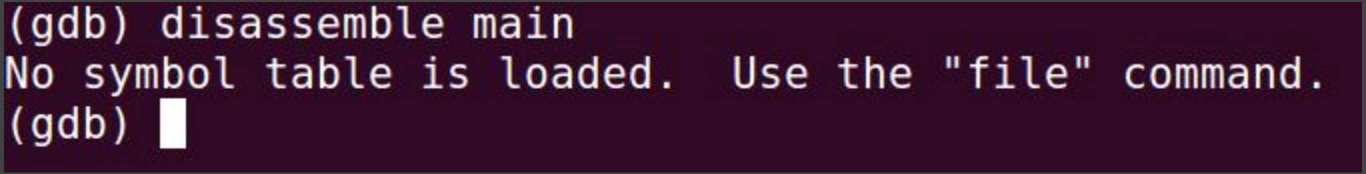


```
CS@kntu:lecture_reveng$ gdb checkpass32s
'GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://g
l.html>
This is free software: you are free to change and re
There is NO WARRANTY, to the extent permitted by law
ing"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resource
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related t
Reading symbols from checkpass32s...(no debugging sy
e.
(gdb) █
```

stripped

Stripped programs

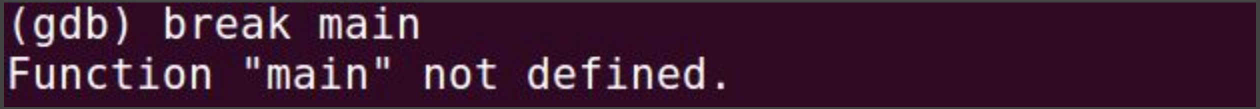
- Where to start?
- Where to look?



```
(gdb) disassemble main
No symbol table is loaded.  Use the "file" command.
(gdb) 
```

Stripped programs

- Where to start?
- Where to look?
- Where to set the breakpoint?



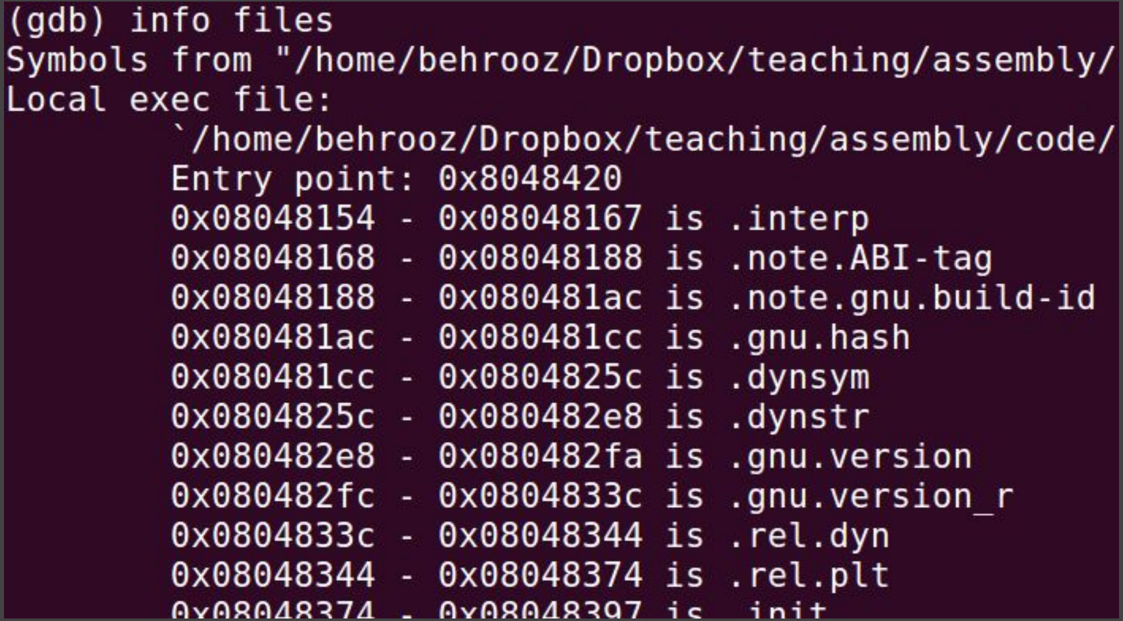
```
(gdb) break main
Function "main" not defined.
```

Stripped programs

- Solution 1: start from the very beginning

Stripped programs

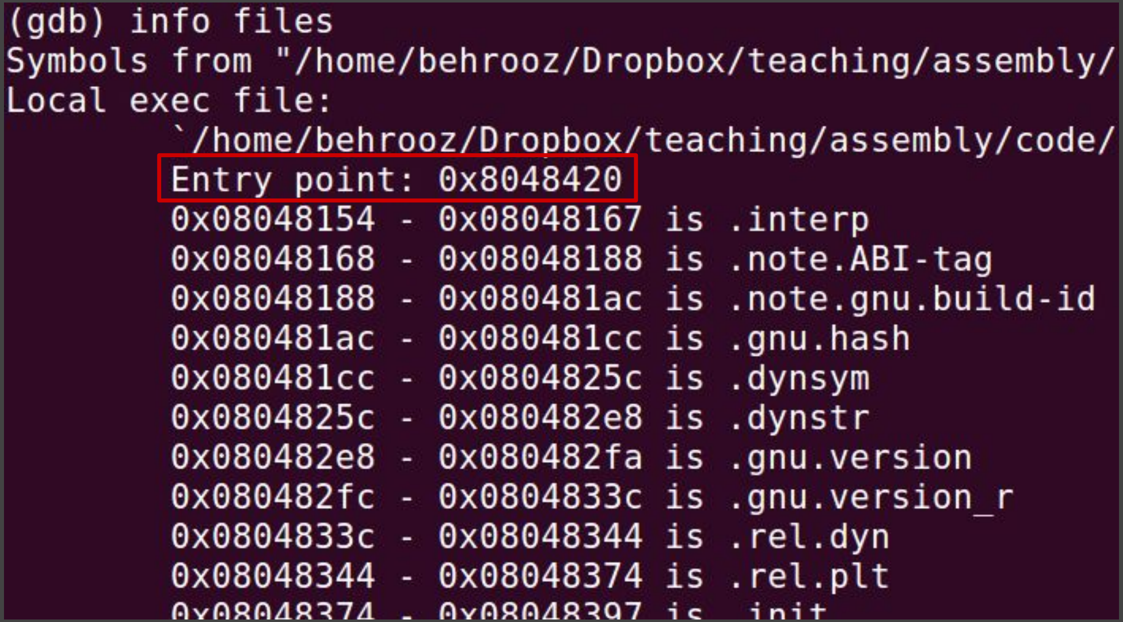
- Solution 1: start from the very beginning



```
(gdb) info files
Symbols from "/home/behrooz/Dropbox/teaching/assembly/
Local exec file:
        `/home/behrooz/Dropbox/teaching/assembly/code/
        Entry point: 0x8048420
        0x08048154 - 0x08048167 is .interp
        0x08048168 - 0x08048188 is .note.ABI-tag
        0x08048188 - 0x080481ac is .note.gnu.build-id
        0x080481ac - 0x080481cc is .gnu.hash
        0x080481cc - 0x0804825c is .dynsym
        0x0804825c - 0x080482e8 is .dynstr
        0x080482e8 - 0x080482fa is .gnu.version
        0x080482fc - 0x0804833c is .gnu.version_r
        0x0804833c - 0x08048344 is .rel.dyn
        0x08048344 - 0x08048374 is .rel.plt
        0x08048374 - 0x08048397 is .init
```

Stripped programs

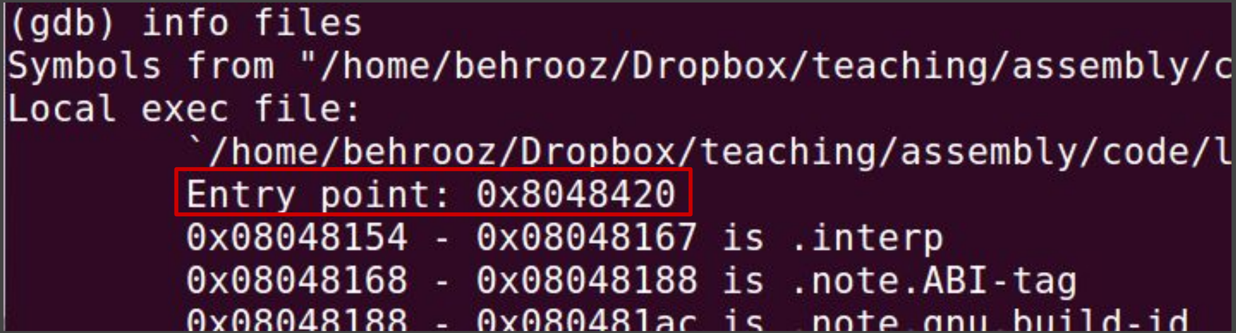
- Solution 1: start from the very beginning



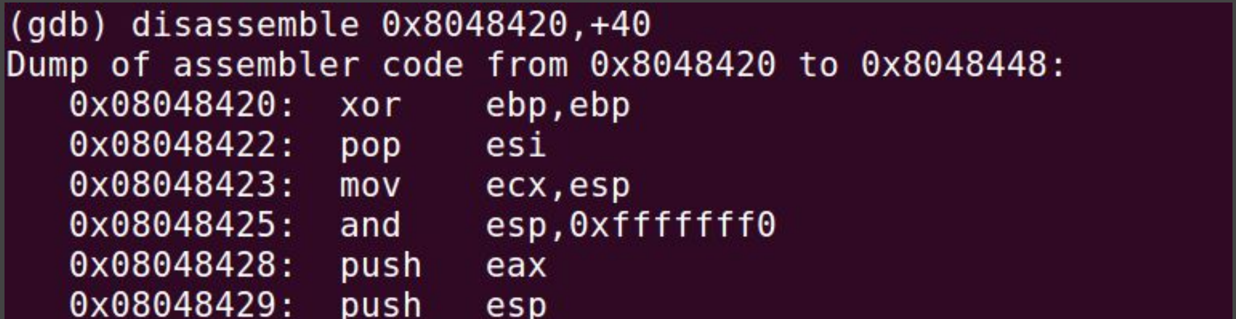
```
(gdb) info files
Symbols from "/home/behrooz/Dropbox/teaching/assembly/
Local exec file:
        `/home/behrooz/Dropbox/teaching/assembly/code/
        Entry point: 0x8048420
        0x08048154 - 0x08048167 is .interp
        0x08048168 - 0x08048188 is .note.ABI-tag
        0x08048188 - 0x080481ac is .note.gnu.build-id
        0x080481ac - 0x080481cc is .gnu.hash
        0x080481cc - 0x0804825c is .dynsym
        0x0804825c - 0x080482e8 is .dynstr
        0x080482e8 - 0x080482fa is .gnu.version
        0x080482fc - 0x0804833c is .gnu.version_r
        0x0804833c - 0x08048344 is .rel.dyn
        0x08048344 - 0x08048374 is .rel.plt
        0x08048374 - 0x08048397 is .init
```

Stripped programs

- Solution 1: start from the very beginning



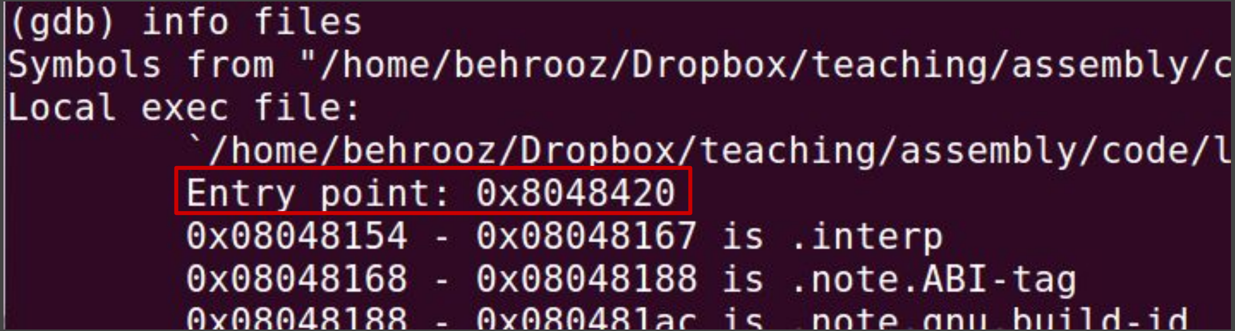
```
(gdb) info files
Symbols from "/home/behrooz/Dropbox/teaching/assembly/c
Local exec file:
        `/home/behrooz/Dropbox/teaching/assembly/code/l
        Entry point: 0x8048420
        0x08048154 - 0x08048167 is .interp
        0x08048168 - 0x08048188 is .note.ABI-tag
        0x08048188 - 0x080481ac is .note.gnu.build-id
```



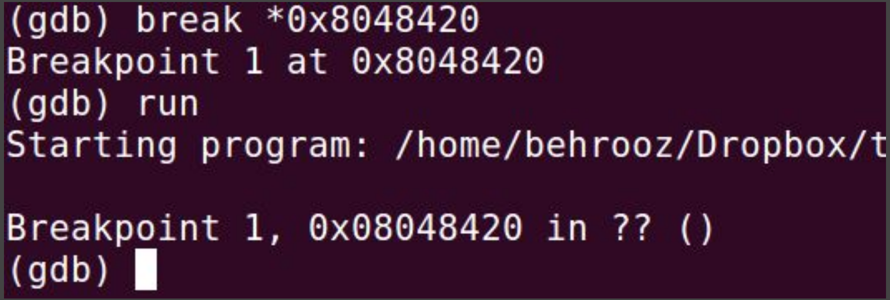
```
(gdb) disassemble 0x8048420,+40
Dump of assembler code from 0x8048420 to 0x8048448:
   0x08048420:  xor    ebp,ebp
   0x08048422:  pop    esi
   0x08048423:  mov    ecx,esp
   0x08048425:  and    esp,0xfffffff0
   0x08048428:  push   eax
   0x08048429:  push   esp
```

Stripped programs

- Solution 1: start from the very beginning

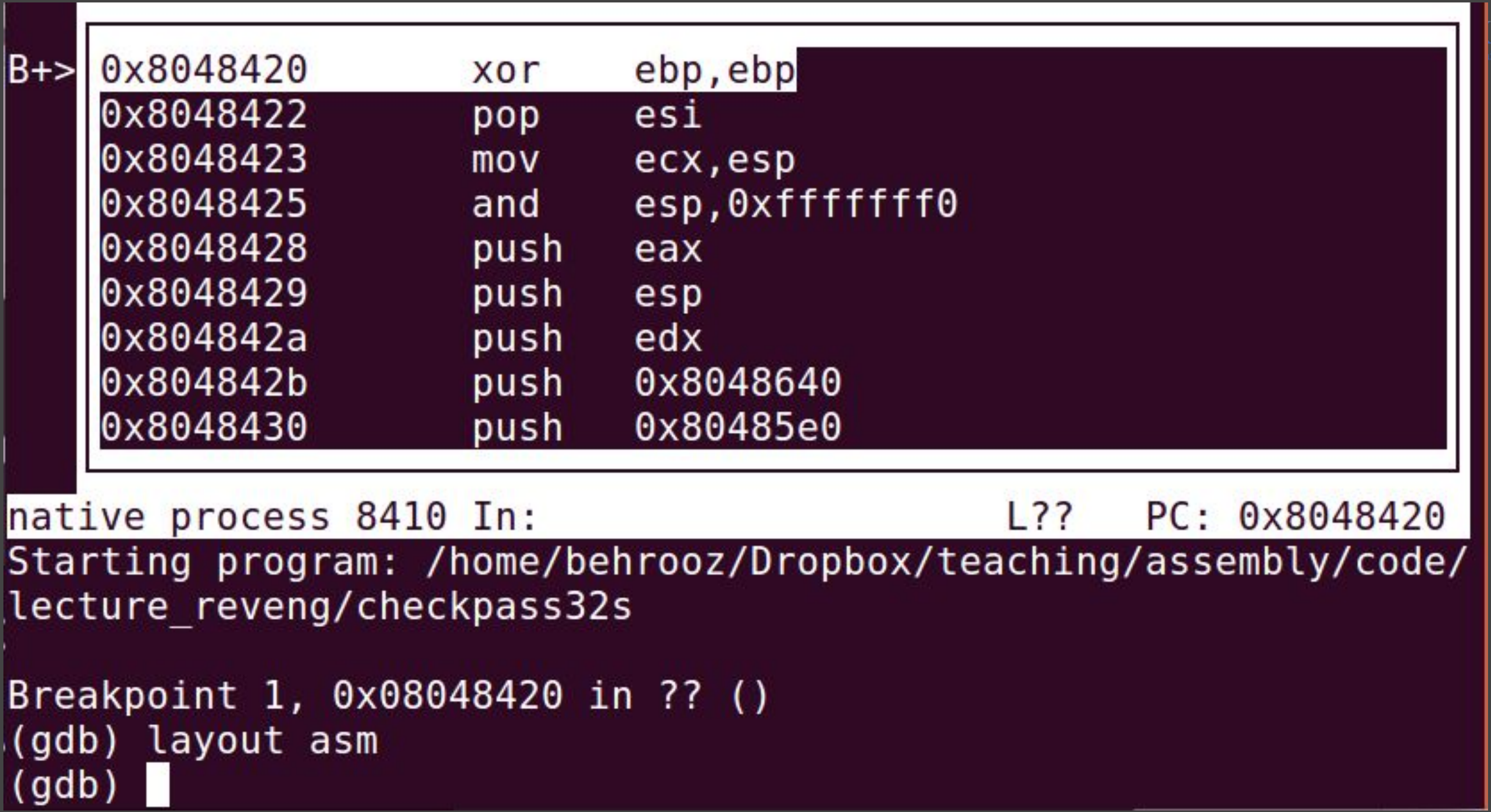


```
(gdb) info files
Symbols from "/home/behrooz/Dropbox/teaching/assembly/c
Local exec file:
        `/home/behrooz/Dropbox/teaching/assembly/code/l
        Entry point: 0x8048420
        0x08048154 - 0x08048167 is .interp
        0x08048168 - 0x08048188 is .note.ABI-tag
        0x08048188 - 0x080481ac is .note.gnu.build-id
```



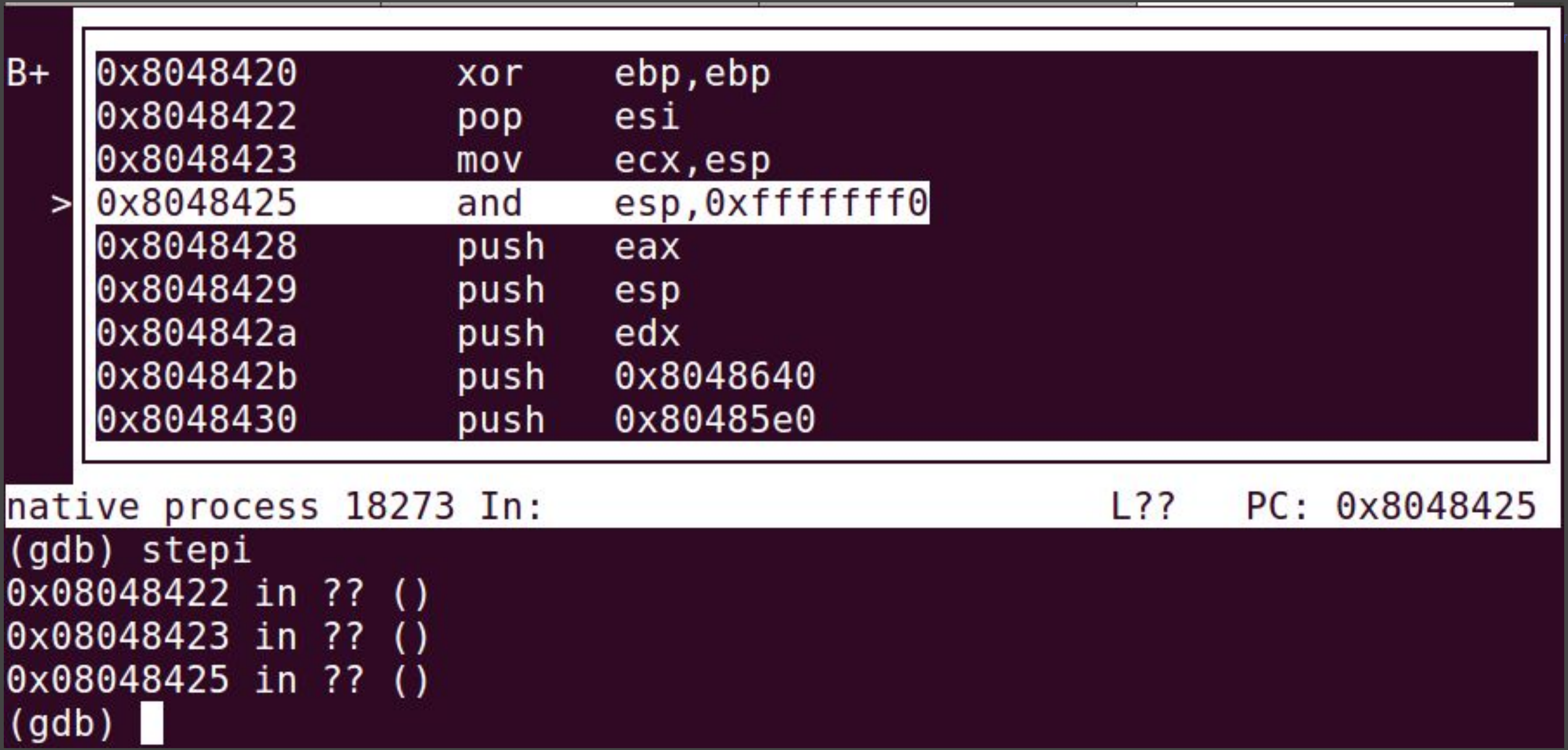
```
(gdb) break *0x8048420
Breakpoint 1 at 0x8048420
(gdb) run
Starting program: /home/behrooz/Dropbox/t

Breakpoint 1, 0x08048420 in ?? ()
(gdb) 
```

Stripped programs: start from entry point



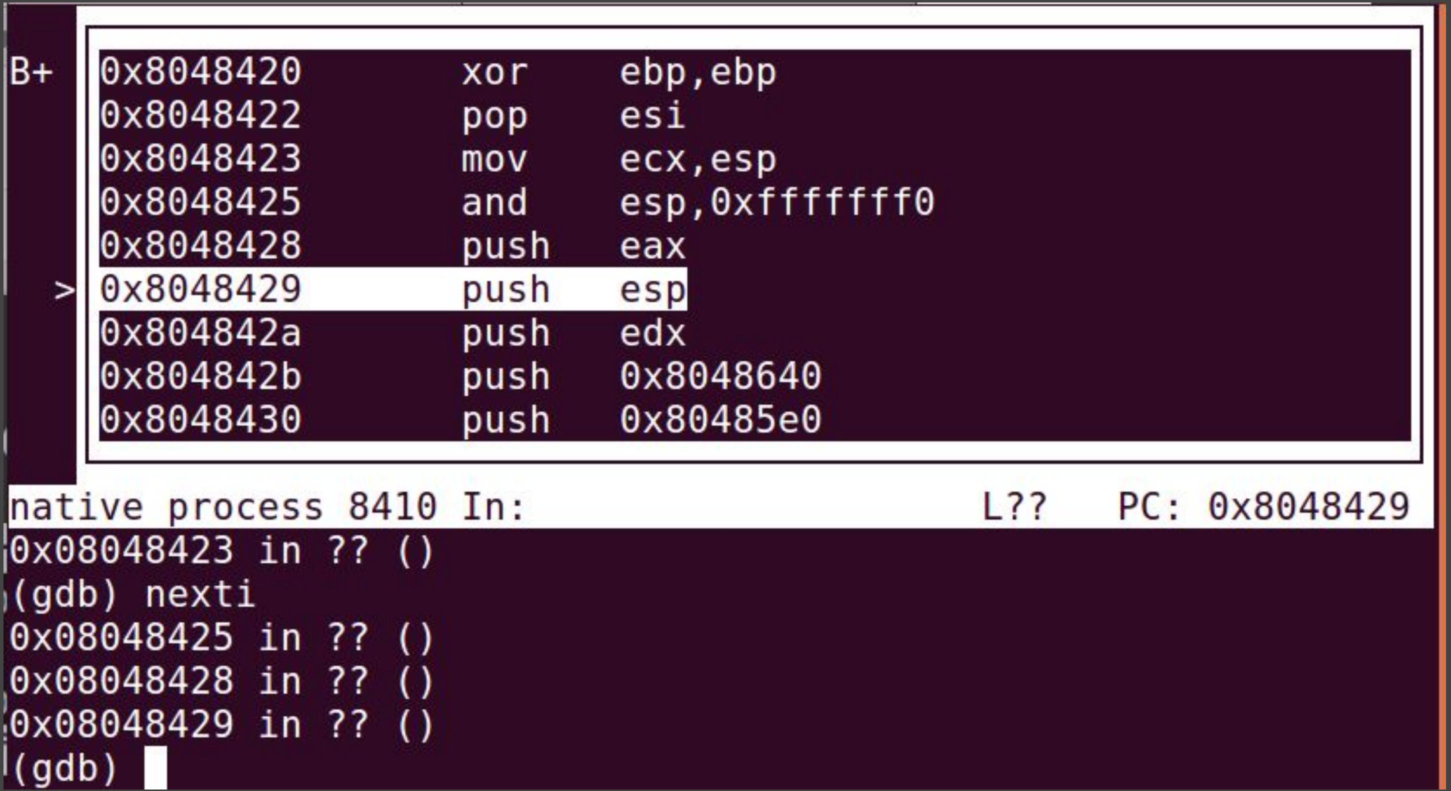
```
B+> 0x8048420        xor    ebp,ebp
    0x8048422        pop    esi
    0x8048423        mov    ecx,esp
    0x8048425        and    esp,0xfffffff0
    0x8048428        push   eax
    0x8048429        push   esp
    0x804842a        push   edx
    0x804842b        push   0x8048640
    0x8048430        push   0x80485e0

native process 8410 In:                        L??   PC: 0x8048420
Starting program: /home/behrooz/Dropbox/teaching/assembly/code/
lecture_reveng/checkpass32s

Breakpoint 1, 0x08048420 in ?? ()
(gdb) layout asm
(gdb) 
```

Stripped programs: start from entry point



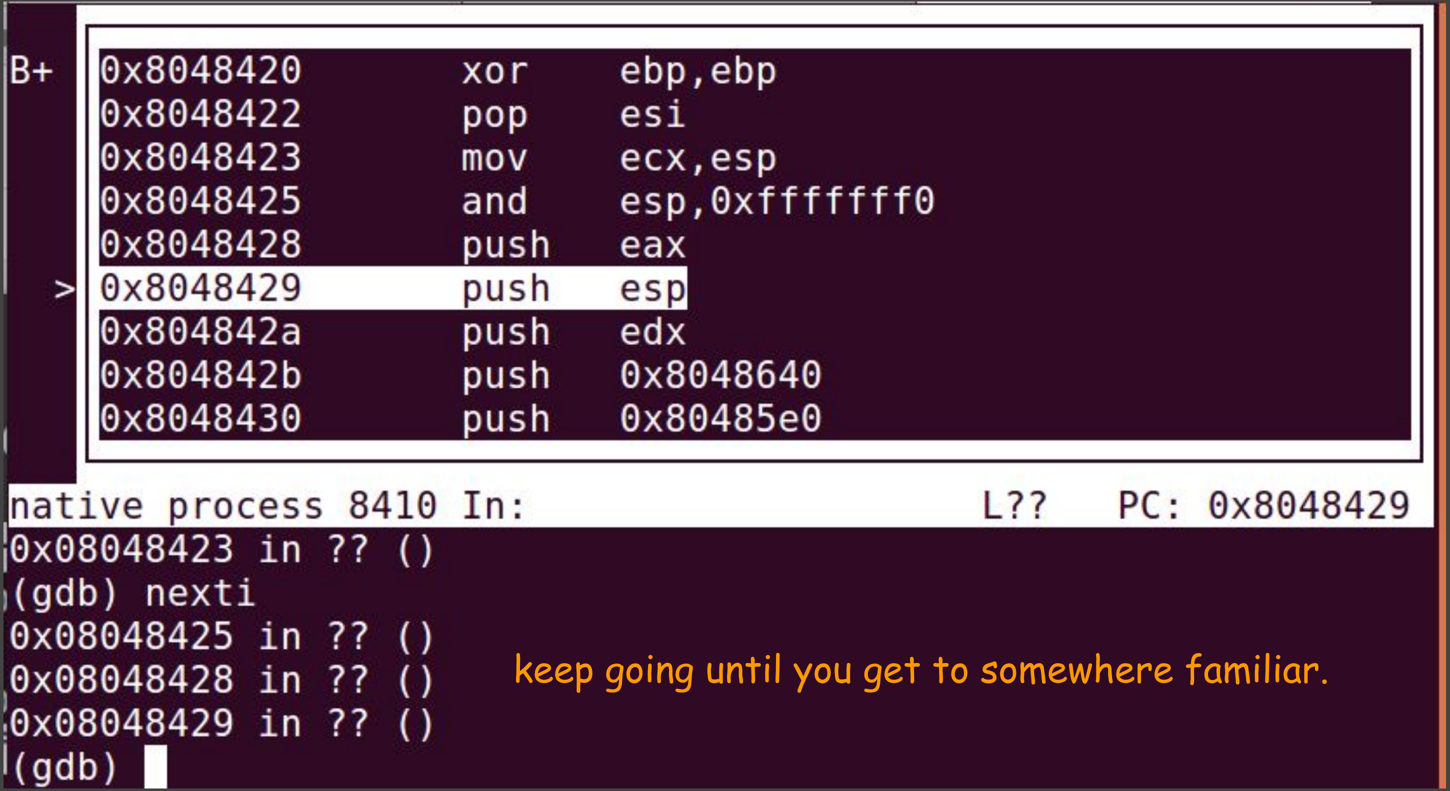
```
B+  0x8048420       xor    ebp,ebp
    0x8048422       pop    esi
    0x8048423       mov    ecx,esp
  > 0x8048425       and    esp,0xfffffff0
    0x8048428       push   eax
    0x8048429       push   esp
    0x804842a       push   edx
    0x804842b       push   0x8048640
    0x8048430       push   0x80485e0

native process 18273 In:                          L??   PC: 0x8048425
(gdb) stepi
0x08048422 in ?? ()
0x08048423 in ?? ()
0x08048425 in ?? ()
(gdb)
```

Stripped programs: start from entry point



```
B+  0x8048420        xor    ebp,ebp
    0x8048422        pop    esi
    0x8048423        mov    ecx,esp
    0x8048425        and    esp,0xfffffff0
    0x8048428        push   eax
  > 0x8048429        push   esp
    0x804842a        push   edx
    0x804842b        push   0x8048640
    0x8048430        push   0x80485e0

native process 8410 In:                        L??   PC: 0x8048429
0x08048423 in ?? ()
(gdb) nexti
0x08048425 in ?? ()
0x08048428 in ?? ()
0x08048429 in ?? ()
(gdb)
```

Stripped programs: start from entry point



```
B+  0x8048420          xor     ebp,ebp
    0x8048422          pop     esi
    0x8048423          mov     ecx,esp
    0x8048425          and     esp,0xfffffff0
    0x8048428          push    eax
  > 0x8048429          push    esp
    0x804842a          push    edx
    0x804842b          push    0x8048640
    0x8048430          push    0x80485e0

native process 8410 In:                     L??   PC: 0x8048429
0x08048423 in ?? ()
(gdb) nexti
0x08048425 in ?? ()
0x08048428 in ?? ()
0x08048429 in ?? ()
(gdb)
```

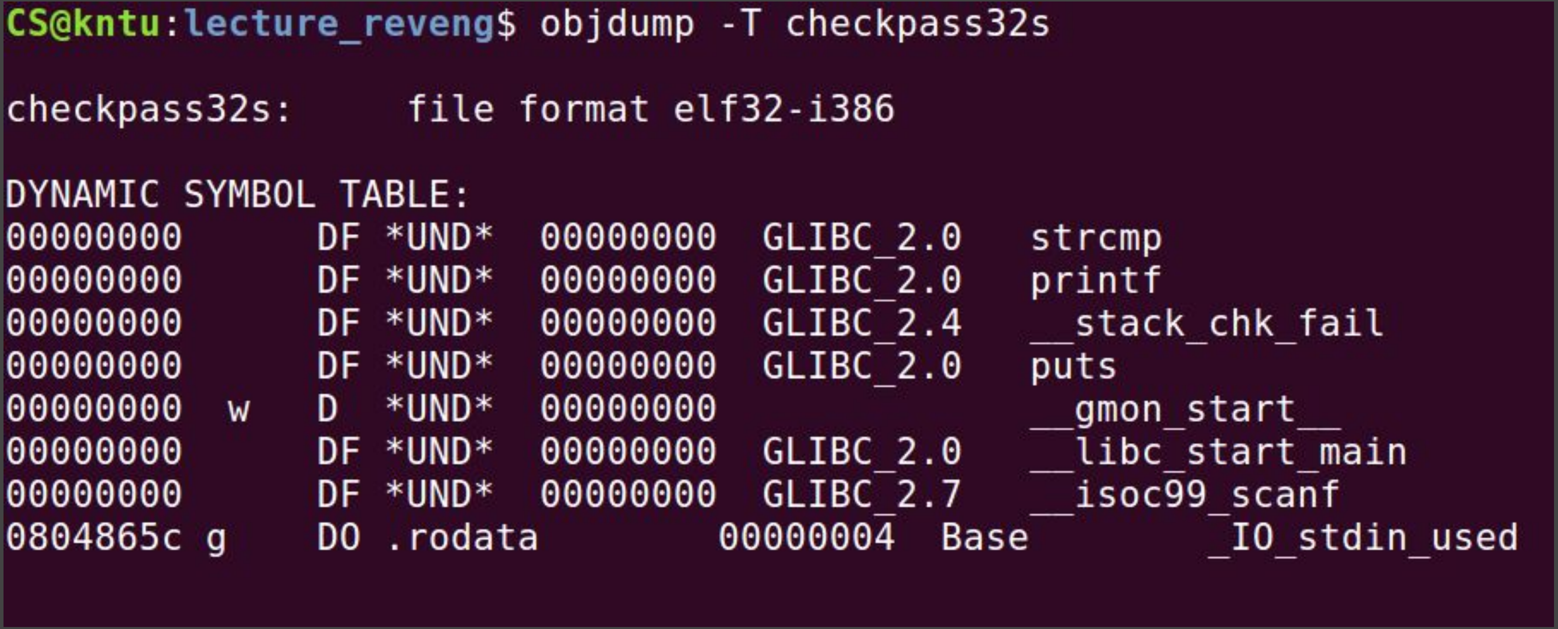
keep going until you get to somewhere familiar.

Stripped programs

- Solution 2: look at interesting library calls

Stripped programs

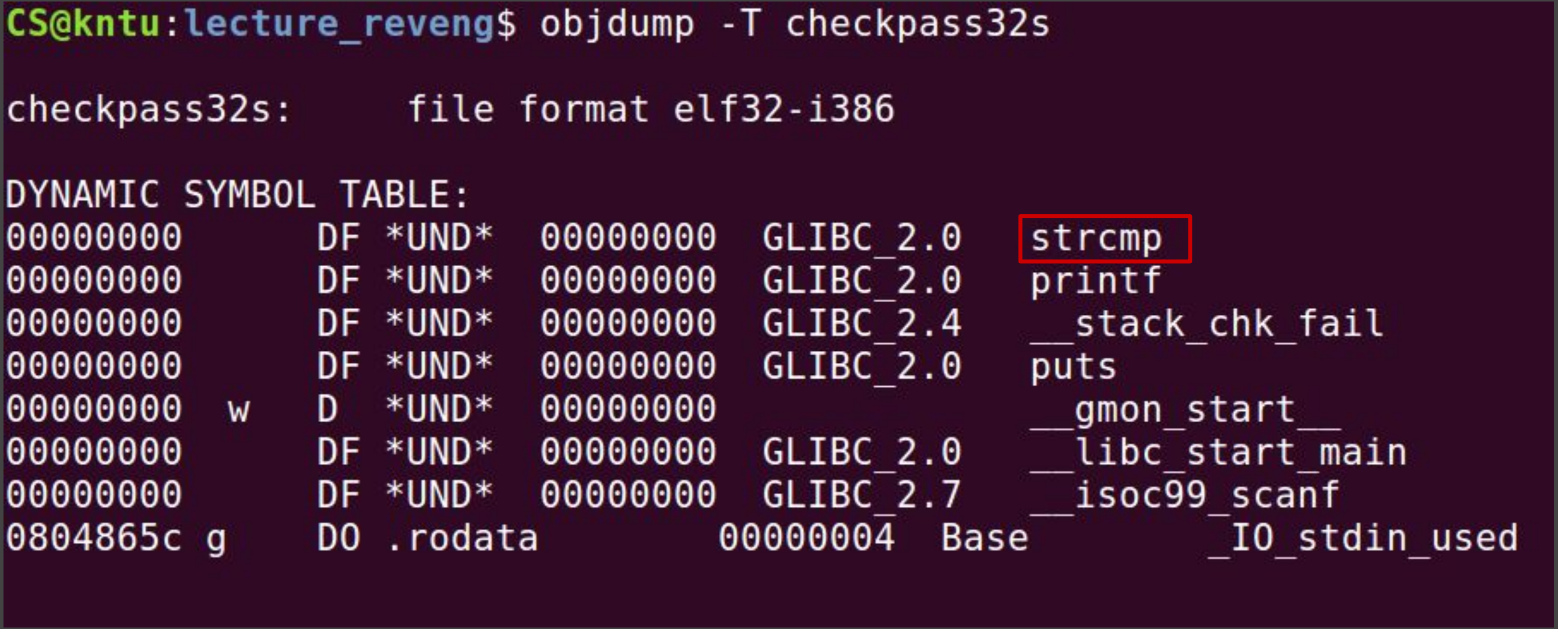
- Solution 2: look at interesting library calls



```
CS@kntu:lecture_reveng$ objdump -T checkpass32s

checkpass32s:     file format elf32-i386

DYNAMIC SYMBOL TABLE:
00000000      DF *UND*  00000000  GLIBC_2.0   strcmp
00000000      DF *UND*  00000000  GLIBC_2.0   printf
00000000      DF *UND*  00000000  GLIBC_2.4   __stack_chk_fail
00000000      DF *UND*  00000000  GLIBC_2.0   puts
00000000  w   D  *UND*  00000000              __gmon_start__
00000000      DF *UND*  00000000  GLIBC_2.0   __libc_start_main
00000000      DF *UND*  00000000  GLIBC_2.7   __isoc99_scanf
0804865c g    DO .rodata        00000004  Base        _IO_stdin_used
```

Stripped programs

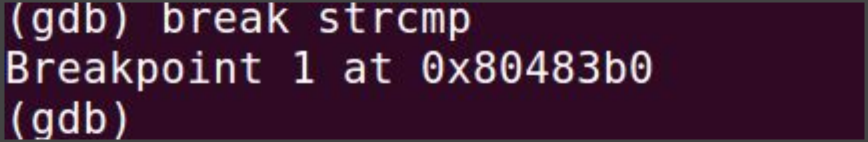
- Solution 2: look at interesting library calls



```
CS@kntu:lecture_reveng$ objdump -T checkpass32s

checkpass32s:     file format elf32-i386

DYNAMIC SYMBOL TABLE:
00000000      DF *UND*  00000000  GLIBC_2.0   strcmp
00000000      DF *UND*  00000000  GLIBC_2.0   printf
00000000      DF *UND*  00000000  GLIBC_2.4   __stack_chk_fail
00000000      DF *UND*  00000000  GLIBC_2.0   puts
00000000  w   D  *UND*  00000000              __gmon_start__
00000000      DF *UND*  00000000  GLIBC_2.0   __libc_start_main
00000000      DF *UND*  00000000  GLIBC_2.7   __isoc99_scanf
0804865c g    DO .rodata        00000004  Base        _IO_stdin_used
```

Stripped programs

- Solution 2: look at interesting library calls
  a. set a breakpoint at the function of interest



```
(gdb) break strcmp
Breakpoint 1 at 0x80483b0
(gdb)
```

Stripped programs

- Solution 2: look at interesting library calls
  a. set a breakpoint at the function of interest
  b. run

```
(gdb) break strcmp
Breakpoint 1 at 0x80483b0
(gdb) run
Starting program:
/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpas
s32s
Enter Password: 1234

Breakpoint 1, 0xf7f25040 in ?? () from /lib/i386-linux-gnu/libc.so.6
(gdb)
```

Stripped programs

- Solution 2: look at interesting library calls
  a. set a breakpoint at the function of interest
  b. run
  c. exit function

```
Enter Password: 1234

Breakpoint 1, 0xf7f25040 in ?? () from /lib/i386-linux-gnu/libc.so.6
(gdb) finish
Run till exit from #0  0xf7f25040 in ?? ()
   from /lib/i386-linux-gnu/libc.so.6
0x08048531 in ?? ()
(gdb)
```

Stripped programs

- Solution 2: look at interesting library calls
  - set a breakpoint at the function of interest
  - run
  - exit function

```
(gdb) disassemble $eip-20, $eip+4
Dump of assembler code from 0x804851d to 0x8048535:
   0x0804851d:    in      eax,0x83
   0x0804851f:    in      al,dx
   0x08048520:    or      BYTE PTR [ebx+0x2c6808ec],al
   0x08048526:    mov     al,ds:0x75ff0804
   0x0804852b:    or      al,ch
   0x0804852d:    jg      0x804852d
   0x0804852f:    (bad)
   0x08048530:    inc     DWORD PTR [ebx-0x3f7aef3c]
End of assembler dump.
```

Oops!

Stripped programs

- Solution 2: look at interesting library calls
    - set a breakpoint at the function of interest
    - run
    - exit function

```
(gdb) disassemble $eip-21, $eip+4
Dump of assembler code from 0x804851c to 0x8048535:
   0x0804851c:    mov    ebp,esp
   0x0804851e:    sub    esp,0x8
   0x08048521:    sub    esp,0x8
   0x08048524:    push   0x804a02c
   0x08048529:    push   DWORD PTR [ebp+0x8]
   0x0804852c:    call   0x80483b0 <strcmp@plt>
=> 0x08048531:    add    esp,0x10
   0x08048534:    test   eax,eax
End of assembler dump.
```

Stripped programs

- Solution 2: look at interesting library calls
  - set a breakpoint at the function of interest
  - run
  - exit function

```
(gdb) disassemble $eip-21, $eip+4
Dump of assembler code from 0x804851c to 0x8048535:
   0x0804851c:    mov     ebp,esp
   0x0804851e:    sub     esp,0x8
   0x08048521:    sub     esp,0x8
   0x08048524:    push    0x804a02c
   0x08048529:    push    DWORD PTR [ebp+0x8]
   0x0804852c:    call    0x80483b0 <strcmp@plt>
=> 0x08048531:    add     esp,0x10
   0x08048534:    test    eax,eax
End of assembler dump.
```

Stripped programs

- Solution 2: look at interesting library calls
  - set a breakpoint at the function of interest
  - run
  - exit function

```
   0x08048521:    sub     esp,0x8
   0x08048524:    push    0x804a02c
   0x08048529:    push    DWORD PTR [ebp+0x8]
   0x0804852c:    call    0x80483b0 <strcmp@plt>
=> 0x08048531:    add     esp,0x10
   0x08048534:    test    eax,eax
End of assembler dump.
(gdb) x/s 0x804a02c
0x804a02c:    "DerakhteDoosti!"
(gdb)
```

Stripped programs

- Solution 2: look at interesting library calls
  - set a breakpoint at the function of interest
  - run
  - exit function
  - see where you are

```
(gdb)info registers
eax            0xffffffff -1
ecx            0x44   68
edx            0xffffccb8 -13128
ebx            0x0    0
esp            0xffffcc80 0xffffcc80
ebp            0xffffcc98 0xffffcc98
esi            0xf7f9b000 -134631424
edi            0xf7f9b000 -134631424
eip            0x8048531  0x8048531
```

Stripped programs

- Solution 2: look at interesting library calls
  - set a breakpoint at the function of interest
  - run
  - exit function
  - see where you are

```
(gdb)info registers
eax            0xffffffff -1
ecx            0x44   68
edx            0xffffccb8 -13128
ebx            0x0    0
esp            0xffffcc80 0xffffcc80
ebp            0xffffcc98 0xffffcc98
esi            0xf7f9b000 -134631424
edi            0xf7f9b000 -134631424
eip            0x8048531  0x8048531
```

GDB: useful tips

- set assembly syntax Intel/AT&T

  -
    ```
    set disassembly-flavor att
    set disassembly-flavor intel
    show disassembly-flavor
    ```

GDB: useful tips

- set assembly syntax Intel/AT&T

  -
    ```
    set disassembly-flavor att
    set disassembly-flavor intel
    show disassembly-flavor
    ```
- pressing Enter repeats previous instruction
  - no need to type nexti/stepi over and over
- display assembly while running
  - layout asm

Patching

```c
char password[] = "DerakhteDoosti!";
int check_password(char *input) {
  return strcmp(input,password)== 0;
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

Patching

```c
char password[] = "DerakhteDoosti!";
int check_password(char *input) {
  return strcmp(input,password)== 0
}

int main() {
  char input[100];

  printf("Enter Password: ");
  scanf("%s", input);

  if (! check_password(input)) {
    printf("Incorrect!\n");
    return 1;
  }

  printf("Correct!\n");
  return 0;
}
```

checkpass1.c

```
$ objdump -d -M intel -j .text checkpass32s

checkpass32s:     file format elf32-i386


Disassembly of section .text:

08048420 <.text>:
 8048420: 31 ed                          xor    ebp,ebp
 8048422: 5e                             pop    esi
 8048423: 89 e1                          mov    ecx,esp
 8048425: 83 e4 f0                       and    esp,0xfffffff0
 8048428: 50                             push   eax
 8048429: 54                             push   esp
 804842a: 52                             push   edx
```

Patching

```
8048576: e8 85 fe ff ff          call   8048400 <__isoc99_scanf@plt>
804857b: 83 c4 10                add    esp,0x10
804857e: 83 ec 0c                sub    esp,0xc
8048581: 8d 45 90                lea    eax,[ebp-0x70]
8048584: 50                      push   eax
8048585: e8 91 ff ff ff          call   804851b
804858a: 83 c4 10                add    esp,0x10
804858d: 85 c0                   test   eax,eax
804858f: 75 17                   jne    80485a8
8048591: 83 ec 0c                sub    esp,0xc
8048594: 68 74 86 04 08          push   0x8048674
8048599: e8 42 fe ff ff          call   80483e0 <puts@plt>
804859e: 83 c4 10                add    esp,0x10
80485a1: b8 01 00 00 00          mov    eax,0x1
80485a6: eb 15                   jmp    80485bd
80485a8: 83 ec 0c                sub    esp,0xc
80485ab: 68 7f 86 04 08          push   0x804867f
80485b0: e8 2b fe ff ff          call   80483e0 <puts@plt>
```

Patching

```
8048576: e8 85 fe ff ff          call   8048400 <__isoc99_scanf@plt>
804857b: 83 c4 10                add    esp,0x10
804857e: 83 ec 0c                sub    esp,0xc
8048581: 8d 45 90                lea    eax,[ebp-0x70]
8048584: 50                      push   eax
8048585: e8 91 ff ff ff          call   804851b
804858a: 83 c4 10                add    esp,0x10
804858d: 85 c0                   test   eax,eax
804858f: 75 17                   jne    80485a8
8048591: 83 ec 0c                sub    esp,0xc
8048594: 68 74 86 04 08          push   0x8048674
8048599: e8 42 fe ff ff          call   80483e0 <puts@plt>
804859e: 83 c4 10                add    esp,0x10
80485a1: b8 01 00 00 00          mov    eax,0x1
80485a6: eb 15                   jmp    80485bd
80485a8: 83 ec 0c                sub    esp,0xc
80485ab: 68 7f 86 04 08          push   0x804867f
80485b0: e8 2b fe ff ff          call   80483e0 <puts@plt>
```

Patching

```
8048576: e8 85 fe ff ff          call   8048400 <__isoc99_scanf@plt>
804857b: 83 c4 10                add    esp,0x10
804857e: 83 ec 0c                sub    esp,0xc
8048581: 8d 45 90                lea    eax,[ebp-0x70]
8048584: 50                      push   eax
8048585: e8 91 ff ff ff          call   804851b
804858a: 83 c4 10                add    esp,0x10
804858d: 85 c0                   test   eax,eax
804858f: 75 17                   jne    80485a8
8048591: 83 ec 0c                sub    esp,0xc
8048594: 68 74 86 04 08          push   0x8048674
8048599: e8 42 fe ff ff          call   80483e0 <puts@plt>
804859e: 83 c4 10                add    esp,0x10
80485a1: b8 01 00 00 00          mov    eax,0x1
80485a6: eb 15                   jmp    80485bd
80485a8: 83 ec 0c                sub    esp,0xc
80485ab: 68 7f 86 04 08          push   0x804867f
80485b0: e8 2b fe ff ff          call   80483e0 <puts@plt>
```

Patching

```
8048576: e8 85 fe ff ff          call   8048400 <__isoc99_scanf@plt>
804857b: 83 c4 10                add    esp,0x10
804857e: 83 ec 0c                sub    esp,0xc
8048581: 8d 45 90                lea    eax,[ebp-0x70]
8048584: 50                      push   eax
8048585: e8 91 ff ff ff          call   804851b
804858a: 83 c4 10                add    esp,0x10
804858d: 85 c0                   test   eax,eax
804858f: 75 17                   jne    80485a8
8048591: 83 ec 0c                sub    esp,0xc
8048594: 68 74 86 04 08          push   0x8048674                incorrect
8048599: e8 42 fe ff ff          call   80483e0 <puts@plt>       password
804859e: 83 c4 10                add    esp,0x10
80485a1: b8 01 00 00 00          mov    eax,0x1
80485a6: eb 15                   jmp    80485bd
80485a8: 83 ec 0c                sub    esp,0xc
80485ab: 68 7f 86 04 08          push   0x804867f                 correct
80485b0: e8 2b fe ff ff          call   80483e0 <puts@plt>       password
```

Patching

```
8048576: e8 85 fe ff ff        call   8048400 <__isoc99_scanf@plt>
804857b: 83 c4 10              add    esp,0x10
804857e: 83 ec 0c              sub    esp,0xc
8048581: 8d 45 90              lea    eax,[ebp-0x70]
8048584: 50                    push   eax
8048585: e8 91 ff ff ff        call   804851b
804858a: 83 c4 10              add    esp,0x10
804858d: 85 c0                 test   eax,eax
804858f: 75 17                 jne    80485a8
8048591: 83 ec 0c              sub    esp,0xc
8048594: 68 74 86 04 08        push   0x8048674
8048599: e8 42 fe ff ff        call   80483e0 <puts@plt>
804859e: 83 c4 10              add    esp,0x10
80485a1: b8 01 00 00 00        mov    eax,0x1
80485a6: eb 15                 jmp    80485bd
80485a8: 83 ec 0c              sub    esp,0xc
80485ab: 68 7f 86 04 08        push   0x804867f
80485b0: e8 2b fe ff ff        call   80483e0 <puts@plt>
```

bypass condition checking and jump directly to 80485ab

* convert jne to jmp (or je)

correct password

Patching

```
8048576: e8 85 fe ff ff          call   8048400 <__isoc99_scanf@plt>
804857b: 83 c4 10                add    esp,0x10
804857e: 83 ec 0c                sub    esp,0xc
8048581: 8d 45 90                lea    eax,[ebp-0x70]
8048584: 50                      push   eax
8048585: e8 91 ff ff ff          call   804851b
804858a: 83 c4 10                add    esp,0x10
804858d: 85 c0                   test   eax,eax
804858f: 75 17   75 -> EB        jne    80485a8
8048591: 83 ec 0c                sub    esp,0xc
8048594: 68 74 86 04 08          push   0x8048674
8048599: e8 42 fe ff ff          call   80483e0 <puts@plt>
804859e: 83 c4 10                add    esp,0x10
80485a1: b8 01 00 00 00          mov    eax,0x1
80485a6: eb 15                   jmp    80485bd
80485a8: 83 ec 0c                sub    esp,0xc
80485ab: 68 7f 86 04 08          push   0x804867f
80485b0: e8 2b fe ff ff          call   80483e0 <puts@plt>
```

jne rel8 ↪ opcode: 75

jmp rel8 ↪ opcode: EB

correct password

Patching: Use a hex editor

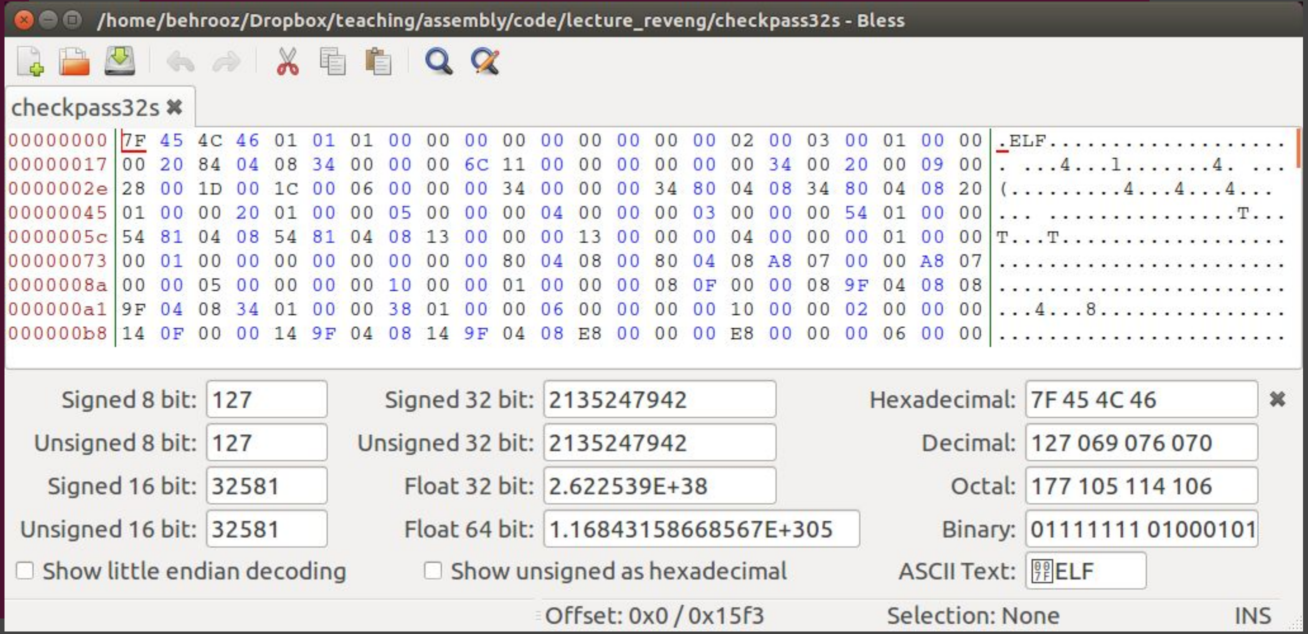
```
8048576: e8 85 fe ff ff
804857b: 83 c4 10
804857e: 83 ec 0c
8048581: 8d 45 90
8048584: 50
8048585: e8 91 ff ff ff
804858a: 83 c4 10
804858d: 85 c0
804858f: 75 17     75 -> EB
8048591: 83 ec 0c
8048594: 68 74 86 04 08
8048599: e8 42 fe ff ff
804859e: 83 c4 10
80485a1: b8 01 00 00 00
80485a6: eb 15
80485a8: 83 ec 0c
80485ab: 68 7f 86 04 08
80485b0: e8 2b fe ff ff
```

```
$ bless checkpass32s
```

Patching: Use a hex editor

```
8048576: e8 85 fe ff ff
804857b: 83 c4 10
804857e: 83 ec 0c
8048581: 8d 45 90
8048584: 50
8048585: e8 91 ff ff ff
804858a: 83 c4 10
804858d: 85 c0
804858f: 75 17        75 -> EB
8048591: 83 ec 0c
8048594: 68 74 86 04 08
8048599: e8 42 fe ff ff
804859e: 83 c4 10
80485a1: b8 01 00 00 00
80485a6: eb 15
80485a8: 83 ec 0c
80485ab: 68 7f 86 04 08
80485b0: e8 2b fe ff ff
```

```
$ bless checkpass32s
```



/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpass32s - Bless

checkpass32s

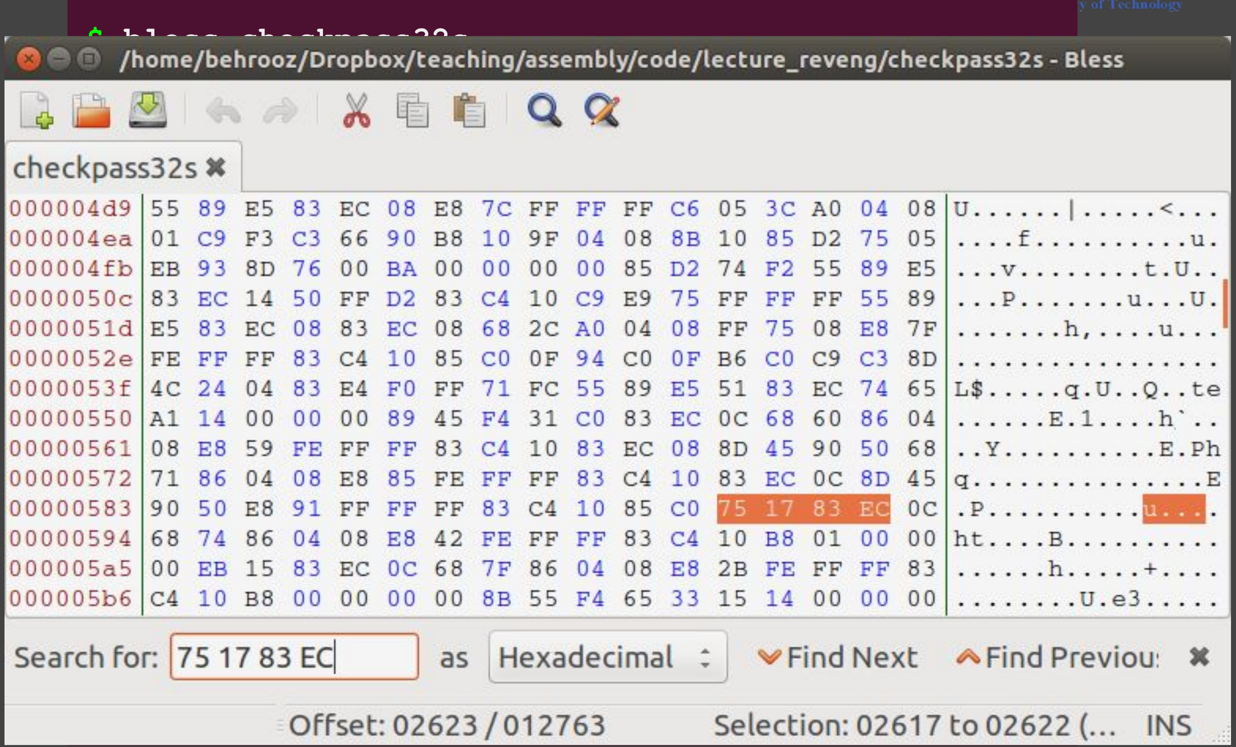
```
00000000 7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00 02 00 03 00 01 00 00 .ELF.....................
00000017 00 20 84 04 08 34 00 00 00 6C 11 00 00 00 00 00 00 34 00 20 00 09 00 . ...4...l.......4. ...
0000002e 28 00 1D 00 1C 00 06 00 00 00 34 00 00 00 34 80 04 08 34 80 04 08 20 (.........4...4...4...
00000045 01 00 00 20 01 00 00 05 00 00 00 04 00 00 00 03 00 00 00 54 01 00 00 ... ...............T...
0000005c 54 81 04 08 54 81 04 08 13 00 00 00 13 00 00 00 04 00 00 00 01 00 00 T...T..................
00000073 00 01 00 00 00 00 00 00 00 80 04 08 00 80 04 08 A8 07 00 00 A8 07 .......................
0000008a 00 00 05 00 00 00 00 10 00 00 01 00 00 00 08 0F 00 00 08 9F 04 08 08 .......................
000000a1 9F 04 08 34 01 00 00 38 01 00 00 06 00 00 00 00 10 00 00 02 00 00 00 ...4...8...............
000000b8 14 0F 00 00 14 9F 04 08 14 9F 04 08 E8 00 00 00 E8 00 00 00 06 00 00 .......................
```

Signed 8 bit:	127	Signed 32 bit:	2135247942	Hexadecimal:	7F 45 4C 46
Unsigned 8 bit:	127	Unsigned 32 bit:	2135247942	Decimal:	127 069 076 070
Signed 16 bit:	32581	Float 32 bit:	2.622539E+38	Octal:	177 105 114 106
Unsigned 16 bit:	32581	Float 64 bit:	1.16843158668567E+305	Binary:	01111111 01000101
☐ Show little endian decoding		☐ Show unsigned as hexadecimal		ASCII Text:	ELF

Offset: 0x0 / 0x15f3 Selection: None INS

Patching: Use a hex editor

```
8048576: e8 85 fe ff ff
804857b: 83 c4 10
804857e: 83 ec 0c
8048581: 8d 45 90
8048584: 50
8048585: e8 91 ff ff ff
804858a: 83 c4 10
804858d: 85 c0
804858f: 75 17        75 -> EB
8048591: 83 ec 0c
8048594: 68 74 86 04 08
8048599: e8 42 fe ff ff
804859e: 83 c4 10
80485a1: b8 01 00 00 00
80485a6: eb 15
80485a8: 83 ec 0c
80485ab: 68 7f 86 04 08
80485b0: e8 2b fe ff ff
```


/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpass32s - Bless
checkpass32s

```
000004d9 55 89 E5 83 EC 08 E8 7C FF FF FF C6 05 3C A0 04 08 U......|.....<...
000004ea 01 C9 F3 C3 66 90 B8 10 9F 04 08 8B 10 85 D2 75 05 ....f..........u.
000004fb EB 93 8D 76 00 BA 00 00 00 00 85 D2 74 F2 55 89 E5 ...v........t.U..
0000050c 83 EC 14 50 FF D2 83 C4 10 C9 E9 75 FF FF FF 55 89 ...P.......u...U.
0000051d E5 83 EC 08 83 EC 08 68 2C A0 04 08 FF 75 08 E8 7F .......h,....u...
0000052e FE FF FF 83 C4 10 85 C0 0F 94 C0 0F B6 C0 C9 C3 8D .................
0000053f 4C 24 04 83 E4 F0 FF 71 FC 55 89 E5 51 83 EC 74 65 L$.....q.U..Q..te
00000550 A1 14 00 00 00 89 45 F4 31 C0 83 EC 0C 68 60 86 04 ......E.1....h`..
00000561 08 E8 59 FE FF FF 83 C4 10 83 EC 08 8D 45 90 50 68 ..Y..........E.Ph
00000572 71 86 04 08 E8 85 FE FF FF 83 C4 10 83 EC 0C 8D 45 q...............E
00000583 90 50 E8 91 FF FF FF 83 C4 10 85 C0 75 17 83 EC 0C .P..........u....
00000594 68 74 86 04 08 E8 42 FE FF FF 83 C4 10 B8 01 00 00 ht....B..........
000005a5 00 EB 15 83 EC 0C 68 7F 86 04 08 E8 2B FE FF FF 83 ......h.....+....
000005b6 C4 10 B8 00 00 00 00 8B 55 F4 65 33 15 14 00 00 00 ........U.e3.....
```

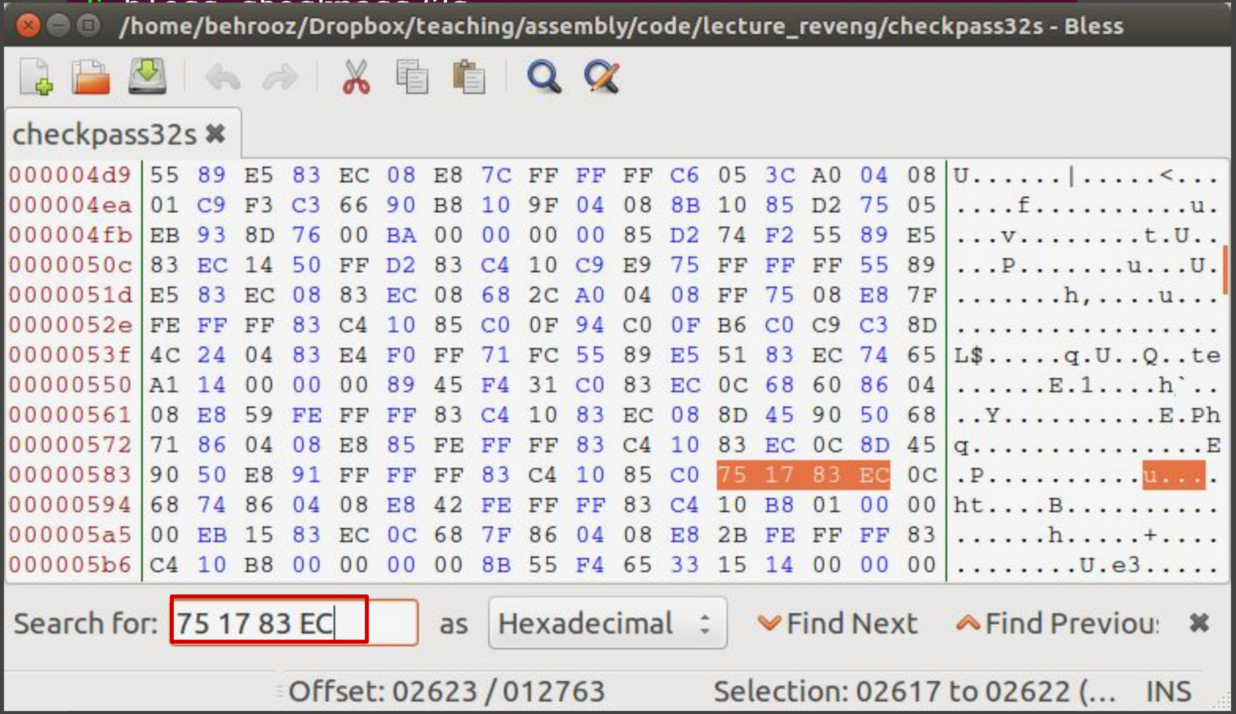
Search for: 75 17 83 EC as Hexadecimal

Offset: 02623 / 012763 Selection: 02617 to 02622 (... INS

Patching: Use a hex editor

```
8048576: e8 85 fe ff ff
804857b: 83 c4 10
804857e: 83 ec 0c
8048581: 8d 45 90
8048584: 50
8048585: e8 91 ff ff ff
804858a: 83 c4 10
804858d: 85 c0
804858f: 75 17      75 -> EB
8048591: 83 ec 0c
8048594: 68 74 86 04 08
8048599: e8 42 fe ff ff
804859e: 83 c4 10
80485a1: b8 01 00 00 00
80485a6: eb 15
80485a8: 83 ec 0c
80485ab: 68 7f 86 04 08
80485b0: e8 2b fe ff ff
```

```
$ bless checkpass32s
```



/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpass32s - Bless

checkpass32s

```
000004d9 55 89 E5 83 EC 08 E8 7C FF FF FF C6 05 3C A0 04 08 U......|.....<...
000004ea 01 C9 F3 C3 66 90 B8 10 9F 04 08 8B 10 85 D2 75 05 ....f..........u.
000004fb EB 93 8D 76 00 BA 00 00 00 00 85 D2 74 F2 55 89 E5 ...v........t.U..
0000050c 83 EC 14 50 FF D2 83 C4 10 C9 E9 75 FF FF FF 55 89 ...P.......u...U.
0000051d E5 83 EC 08 83 EC 08 68 2C A0 04 08 FF 75 08 E8 7F .......h,....u...
0000052e FE FF FF 83 C4 10 85 C0 0F 94 C0 0F B6 C0 C9 C3 8D .................
0000053f 4C 24 04 83 E4 F0 FF 71 FC 55 89 E5 51 83 EC 74 65 L$.....q.U..Q..te
00000550 A1 14 00 00 00 89 45 F4 31 C0 83 EC 0C 68 60 86 04 ......E.1....h`..
00000561 08 E8 59 FE FF FF 83 C4 10 83 EC 08 8D 45 90 50 68 ..Y..........E.Ph
00000572 71 86 04 08 E8 85 FE FF FF 83 C4 10 83 EC 0C 8D 45 q...............E
00000583 90 50 E8 91 FF FF FF 83 C4 10 85 C0 75 17 83 EC 0C .P..........u....
00000594 68 74 86 04 08 E8 42 FE FF FF 83 C4 10 B8 01 00 00 ht....B..........
000005a5 00 EB 15 83 EC 0C 68 7F 86 04 08 E8 2B FE FF FF 83 ......h.....+....
000005b6 C4 10 B8 00 00 00 00 8B 55 F4 65 33 15 14 00 00 00 ........U.e3.....
```

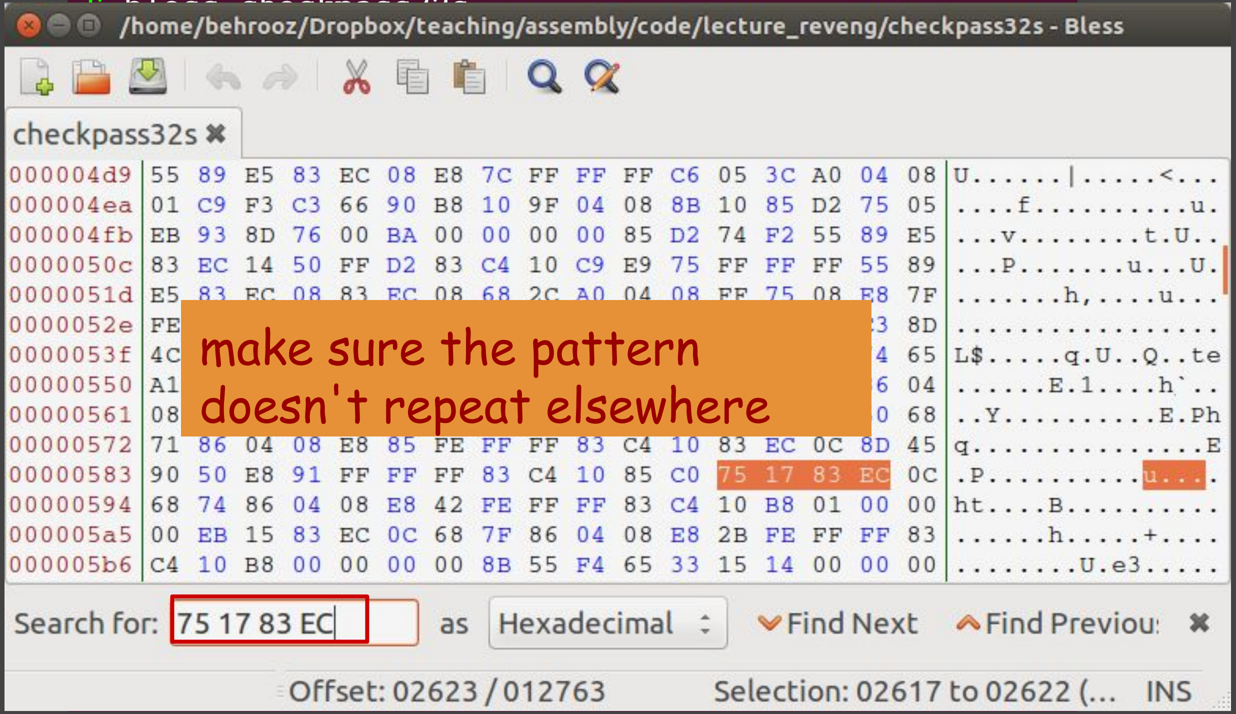
Search for: 75 17 83 EC as Hexadecimal

Offset: 02623 / 012763 Selection: 02617 to 02622 (... INS

Patching: Use a hex editor

```
8048576: e8 85 fe ff ff
804857b: 83 c4 10
804857e: 83 ec 0c
8048581: 8d 45 90
8048584: 50
8048585: e8 91 ff ff ff
804858a: 83 c4 10
804858d: 85 c0
804858f: 75 17      75 -> EB
8048591: 83 ec 0c
8048594: 68 74 86 04 08
8048599: e8 42 fe ff ff
804859e: 83 c4 10
80485a1: b8 01 00 00 00
80485a6: eb 15
80485a8: 83 ec 0c
80485ab: 68 7f 86 04 08
80485b0: e8 2b fe ff ff
```

```
$ bless checkpass32s
```



/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpass32s - Bless

checkpass32s

```
000004d9 55 89 E5 83 EC 08 E8 7C FF FF FF C6 05 3C A0 04 08 U......|.....<...
000004ea 01 C9 F3 C3 66 90 B8 10 9F 04 08 8B 10 85 D2 75 05 ....f..........u.
000004fb EB 93 8D 76 00 BA 00 00 00 00 85 D2 74 F2 55 89 E5 ...v.........t.U..
0000050c 83 EC 14 50 FF D2 83 C4 10 C9 E9 75 FF FF FF 55 89 ...P.......u...U.
0000051d E5 83 EC 08 83 EC 08 68 2C A0 04 08 FF 75 08 E8 7F .......h,....u...
0000052e FE                                            3 8D .................
0000053f 4C                                            4 65 L$.....q.U..Q..te
00000550 A1                                            6 04 ......E.1....h`..
00000561 08                                            0 68 ..Y..........E.Ph
00000572 71 86 04 08 E8 85 FE FF FF 83 C4 10 83 EC 0C 8D 45 q...............E
00000583 90 50 E8 91 FF FF FF 83 C4 10 85 C0 75 17 83 EC 0C .P...........u....
00000594 68 74 86 04 08 E8 42 FE FF FF 83 C4 10 B8 01 00 00 ht....B..........
000005a5 00 EB 15 83 EC 0C 68 7F 86 04 08 E8 2B FE FF FF 83 ......h.....+....
000005b6 C4 10 B8 00 00 00 00 8B 55 F4 65 33 15 14 00 00 00 ........U.e3.....
```

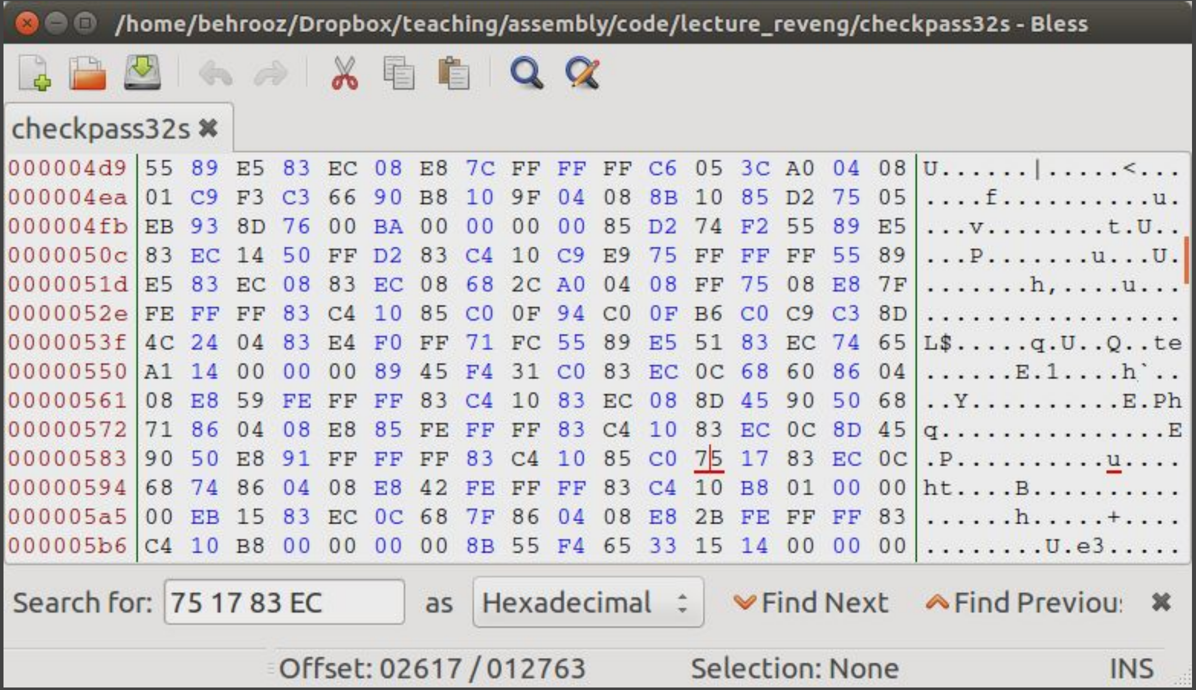
make sure the pattern doesn't repeat elsewhere

Search for: 75 17 83 EC as Hexadecimal

Offset: 02623 / 012763   Selection: 02617 to 02622 (...   INS

Patching: Use a hex editor

```
8048576: e8 85 fe ff ff
804857b: 83 c4 10
804857e: 83 ec 0c
8048581: 8d 45 90
8048584: 50
8048585: e8 91 ff ff ff
804858a: 83 c4 10
804858d: 85 c0
804858f: 75 17       75 -> EB
8048591: 83 ec 0c
8048594: 68 74 86 04 08
8048599: e8 42 fe ff ff
804859e: 83 c4 10
80485a1: b8 01 00 00 00
80485a6: eb 15
80485a8: 83 ec 0c
80485ab: 68 7f 86 04 08
80485b0: e8 2b fe ff ff
```


/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpass32s - Bless
checkpass32s

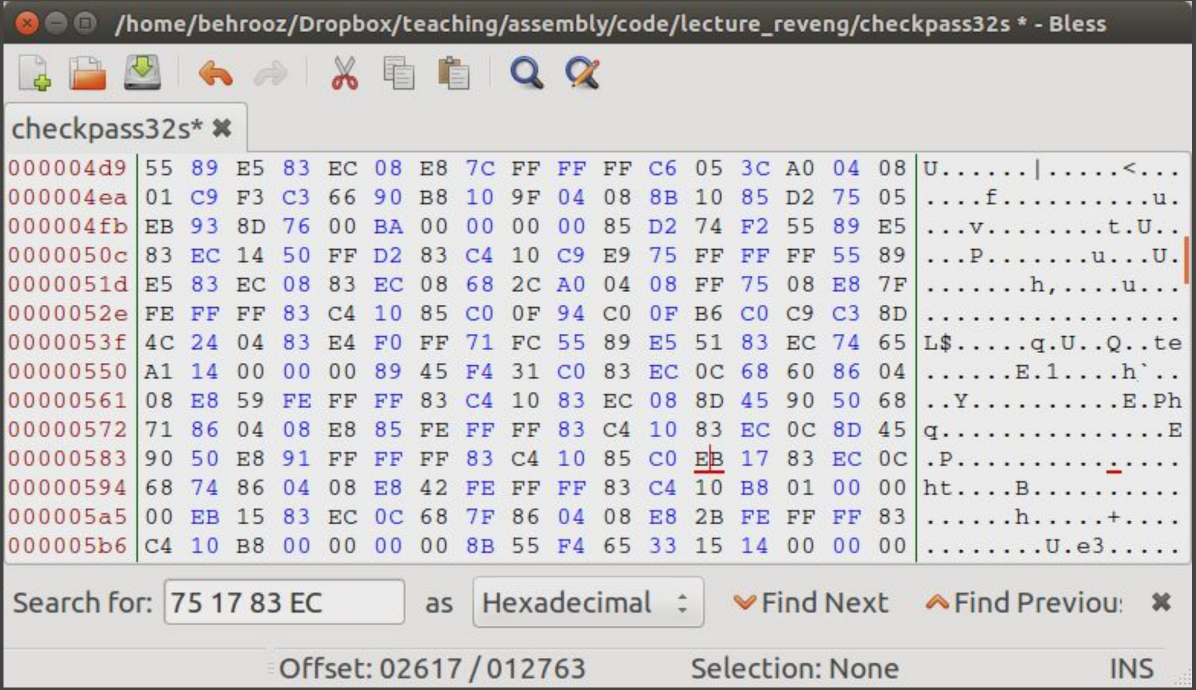
```
000004d9 55 89 E5 83 EC 08 E8 7C FF FF FF C6 05 3C A0 04 08 U......|.....<...
000004ea 01 C9 F3 C3 66 90 B8 10 9F 04 08 8B 10 85 D2 75 05 ....f..........u.
000004fb EB 93 8D 76 00 BA 00 00 00 00 85 D2 74 F2 55 89 E5 ...v........t.U..
0000050c 83 EC 14 50 FF D2 83 C4 10 C9 E9 75 FF FF FF 55 89 ...P.......u...U.
0000051d E5 83 EC 08 83 EC 08 68 2C A0 04 08 FF 75 08 E8 7F .......h,....u...
0000052e FE FF FF 83 C4 10 85 C0 0F 94 C0 0F B6 C0 C9 C3 8D .................
0000053f 4C 24 04 83 E4 F0 FF 71 FC 55 89 E5 51 83 EC 74 65 L$.....q.U..Q..te
00000550 A1 14 00 00 00 89 45 F4 31 C0 83 EC 0C 68 60 86 04 ......E.1....h`..
00000561 08 E8 59 FE FF FF 83 C4 10 83 EC 08 8D 45 90 50 68 ..Y..........E.Ph
00000572 71 86 04 08 E8 85 FE FF FF 83 C4 10 83 EC 0C 8D 45 q...............E
00000583 90 50 E8 91 FF FF FF 83 C4 10 85 C0 75 17 83 EC 0C .P..........u....
00000594 68 74 86 04 08 E8 42 FE FF FF 83 C4 10 B8 01 00 00 ht....B..........
000005a5 00 EB 15 83 EC 0C 68 7F 86 04 08 E8 2B FE FF FF 83 ......h.....+....
000005b6 C4 10 B8 00 00 00 00 8B 55 F4 65 33 15 14 00 00 00 ........U.e3.....
```

Search for: 75 17 83 EC as Hexadecimal

Offset: 02617 / 012763 Selection: None INS

Patching: Use a hex editor

```
8048576: e8 85 fe ff ff
804857b: 83 c4 10
804857e: 83 ec 0c
8048581: 8d 45 90
8048584: 50
8048585: e8 91 ff ff ff
804858a: 83 c4 10
804858d: 85 c0
804858f: 75 17       75 -> EB
8048591: 83 ec 0c
8048594: 68 74 86 04 08
8048599: e8 42 fe ff ff
804859e: 83 c4 10
80485a1: b8 01 00 00 00
80485a6: eb 15
80485a8: 83 ec 0c
80485ab: 68 7f 86 04 08
80485b0: e8 2b fe ff ff
```



/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpass32s * - Bless

checkpass32s*

```
000004d9 55 89 E5 83 EC 08 E8 7C FF FF FF C6 05 3C A0 04 08 U......|.....<...
000004ea 01 C9 F3 C3 66 90 B8 10 9F 04 08 8B 10 85 D2 75 05 ....f..........u.
000004fb EB 93 8D 76 00 BA 00 00 00 00 85 D2 74 F2 55 89 E5 ...v.........t.U..
0000050c 83 EC 14 50 FF D2 83 C4 10 C9 E9 75 FF FF FF 55 89 ...P.......u...U.
0000051d E5 83 EC 08 83 EC 08 68 2C A0 04 08 FF 75 08 E8 7F .......h,....u...
0000052e FE FF FF 83 C4 10 85 C0 0F 94 C0 0F B6 C0 C9 C3 8D .................
0000053f 4C 24 04 83 E4 F0 FF 71 FC 55 89 E5 51 83 EC 74 65 L$.....q.U..Q..te
00000550 A1 14 00 00 00 89 45 F4 31 C0 83 EC 0C 68 60 86 04 ......E.1....h`..
00000561 08 E8 59 FE FF FF 83 C4 10 83 EC 08 8D 45 90 50 68 ..Y...........E.Ph
00000572 71 86 04 08 E8 85 FE FF FF 83 C4 10 83 EC 0C 8D 45 q...............E
00000583 90 50 E8 91 FF FF FF 83 C4 10 85 C0 EB 17 83 EC 0C .P...............
00000594 68 74 86 04 08 E8 42 FE FF FF 83 C4 10 B8 01 00 00 ht....B..........
000005a5 00 EB 15 83 EC 0C 68 7F 86 04 08 E8 2B FE FF FF 83 ......h.....+....
000005b6 C4 10 B8 00 00 00 00 8B 55 F4 65 33 15 14 00 00 00 ........U.e3.....
```

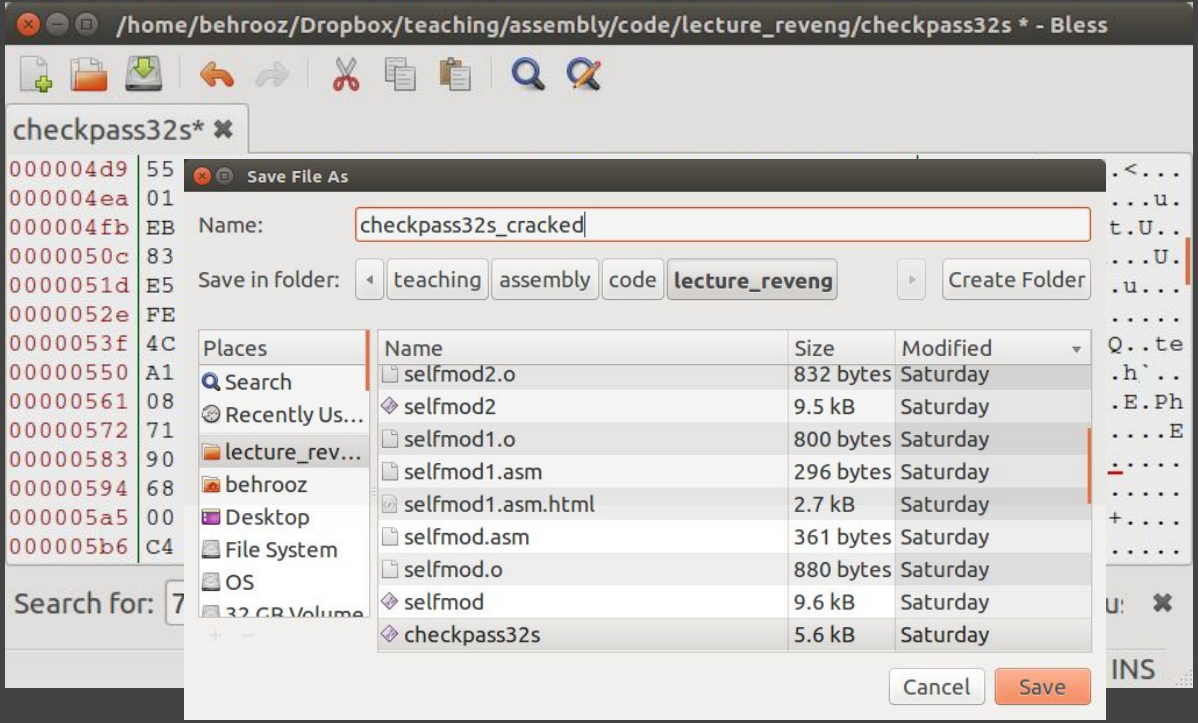
Search for: 75 17 83 EC as Hexadecimal Find Next Find Previou:

Offset: 02617 / 012763 Selection: None INS

Patching: Use a hex editor

```
8048576: e8 85 fe ff ff
804857b: 83 c4 10
804857e: 83 ec 0c
8048581: 8d 45 90
8048584: 50
8048585: e8 91 ff ff ff
804858a: 83 c4 10
804858d: 85 c0
804858f: 75 17
8048591: 83 ec 0c
8048594: 68 74 86 04 08
8048599: e8 42 fe ff ff
804859e: 83 c4 10
80485a1: b8 01 00 00 00
80485a6: eb 15
80485a8: 83 ec 0c
80485ab: 68 7f 86 04 08
80485b0: e8 2b fe ff ff
```

$\boxed{75\ 17}$ 75 -> EB



/home/behrooz/Dropbox/teaching/assembly/code/lecture_reveng/checkpass32s * - Bless

checkpass32s* ✖

```
000004d9 55   .<...
000004ea 01   ...u.
000004fb EB   t.U..
0000050c 83   ...U.
0000051d E5   .u...
0000052e FE   .....
0000053f 4C   Q..te
00000550 A1   .h`..
00000561 08   .E.Ph
00000572 71   ....E
00000583 90   .....
00000594 68   .....
000005a5 00   +....
000005b6 C4   .....
```

Search for: 7

INS

Save File As

Name: checkpass32s_cracked

Save in folder: teaching | assembly | code | **lecture_reveng** | Create Folder

Places
Search
Recently Us...
lecture_rev...
behrooz
Desktop
File System
OS
32 GB Volume

Name	Size	Modified
selfmod2.o	832 bytes	Saturday
selfmod2	9.5 kB	Saturday
selfmod1.o	800 bytes	Saturday
selfmod1.asm	296 bytes	Saturday
selfmod1.asm.html	2.7 kB	Saturday
selfmod.asm	361 bytes	Saturday
selfmod.o	880 bytes	Saturday
selfmod	9.6 kB	Saturday
checkpass32s	5.6 kB	Saturday

Cancel | Save

Patching

```
8048576: e8 85 fe ff ff
804857b: 83 c4 10
804857e: 83 ec 0c
8048581: 8d 45 90
8048584: 50
8048585: e8 91 ff ff ff
804858a: 83 c4 10
804858d: 85 c0
804858f: 75 17      75 -> EB
8048591: 83 ec 0c
8048594: 68 74 86 04 08
8048599: e8 42 fe ff ff
804859e: 83 c4 10
80485a1: b8 01 00 00 00
80485a6: eb 15
80485a8: 83 ec 0c
80485ab: 68 7f 86 04 08
80485b0: e8 2b fe ff ff
```

```
$ ./checkpass32s_cracked
bash: ./checkpass32s_cracked: Permission denied

$ chmod u+x ./checkpass32s_cracked

$ ./checkpass32s_cracked
Enter Password: 1234
Correct!
```

Patching

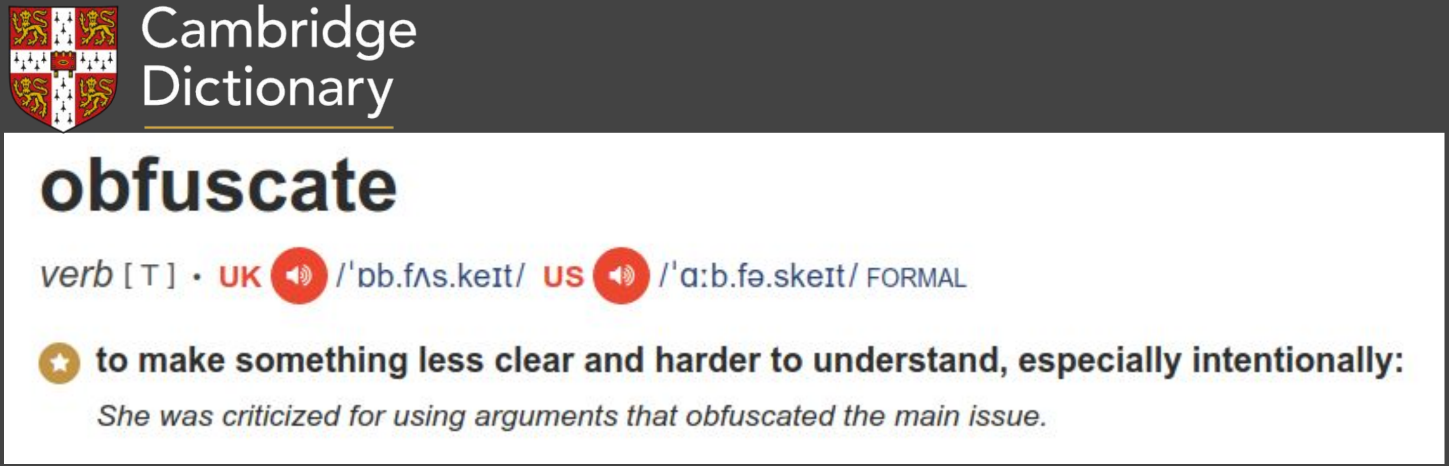
- Not specific to reverse engineering
    - Fixing bugs, vulnerabilities,
    - Updates
- Linux commands/applications
    - diff/patch
    - bsdiff/bspatch
    - rdiff

Protect your code against reversing

K. N. Toosi
University of Technology

- omit debug info
- Strip symbols
- Optimize code
- Code obfuscation
- Self-modifying code
- Debugger detector

Code Obfuscation


Cambridge Dictionary

obfuscate

verb [T] · UK /ˈɒb.fʌs.keɪt/ US /ˈɑːb.fə.skeɪt/ FORMAL

to make something less clear and harder to understand, especially intentionally:

She was criticized for using arguments that obfuscated the main issue.

Code Obfuscation

K. N. Toosi
University of Technology

- Optimize code
- non-intuitive instructions
- code obfuscators
- reorder code
- insert dummy code
- Code encryption/self modifying code
  - all at the beginning
  - on the flow

Detect debuggers

- Use OS API's
- Delayed execution
- breakpoint interrupts (int 1)
- check if a debugger is installed

Self-modifying code

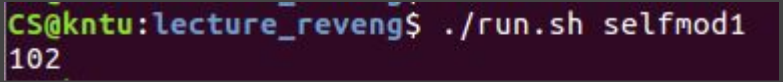
```
segment .text

global asm_main

asm_main:

        pusha
        ;; ===========================



        mov eax, 100
        mov ebx, 2

        add eax, ebx


        call print_int
        call print_nl
```

selfmod1.asm

Self-modifying code

```
segment .text

global asm_main

asm_main:

        pusha
        ;; ============================



        mov eax, 100
        mov ebx, 2

        add eax, ebx


        call print_int
        call print_nl
```

selfmod1.asm



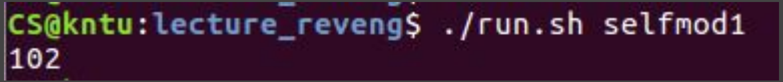
```
cs@kntu:lecture_reveng$ ./run.sh selfmod1
102
```

Self-modifying code

```
segment .text

global asm_main

asm_main:

        pusha
        ;; ==============================



        mov eax, 100
        mov ebx, 2

        add eax, ebx


        call print_int
        call print_nl
```

selfmod1.asm

`add reg32/mem32, reg32` opcode: `0x01`



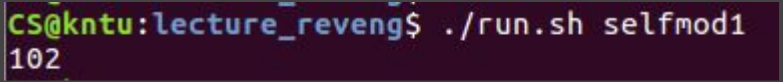
```
cs@kntu:lecture_reveng$ ./run.sh selfmod1
102
```

Self-modifying code

```
segment .text

global asm_main

asm_main:

        pusha
        ;; ============================



        mov eax, 100
        mov ebx, 2

        add eax, ebx


        call print_int
        call print_nl
```

selfmod1.asm

```
add reg32/mem32, reg32   opcode: 0x01

sub reg32/mem32, reg32   opcode: 0x29
```



```
CS@kntu:lecture_reveng$ ./run.sh selfmod1
102
```

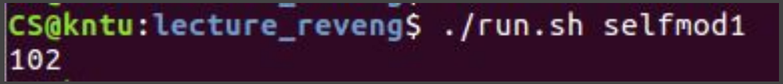
Self-modifying code

```
segment .text

global asm_main

asm_main:

        pusha
        ;; ============================



        mov eax, 100
        mov ebx, 2

        add eax, ebx


        call print_int
        call print_nl
```

selfmod1.asm

The idea: change the opcode 0x01 to 0x29

```
add reg32/mem32, reg32   opcode: 0x01

sub reg32/mem32, reg32   opcode: 0x29
```



```
cs@kntu:lecture_reveng$ ./run.sh selfmod1
102
```

Self-modifying code

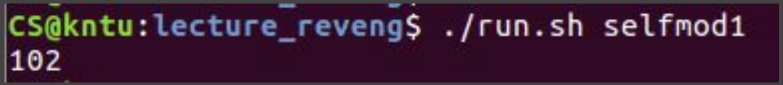
```
segment .text

global asm_main

asm_main:

        pusha
        ;; ============================



        mov eax, 100
        mov ebx, 2

        add eax, ebx


        call print_int
        call print_nl
```

selfmod1.asm

The idea: change the opcode 0x01 to 0x29

at run time!

add	reg32/mem32,	reg32	opcode: 0x01
sub	reg32/mem32,	reg32	opcode: 0x29



```
cs@kntu:lecture_reveng$ ./run.sh selfmod1
102
```

Self-modifying code

```
segment .text

global asm_main

asm_main:

        pusha
        ;; ==============================

        mov  byte [label1], 0x29

        mov eax, 100
        mov ebx, 2
label1:
        add eax, ebx


        call print_int
        call print_nl
```

selfmod2.asm

The idea: change the opcode 0x01 to 0x29 **at run time!**

```
add reg32/mem32, reg32   opcode: 0x01

sub reg32/mem32, reg32   opcode: 0x29
```

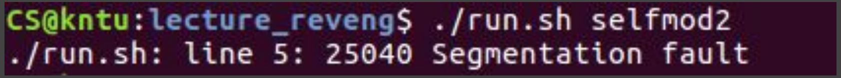
Self-modifying code

```
segment .text

global asm_main

asm_main:

        pusha
        ;; ================================

        mov  byte [label1], 0x29

        mov eax, 100
        mov ebx, 2
label1:
        add eax, ebx


        call print_int
        call print_nl
```

selfmod2.asm

The idea: change the opcode 0x01 to 0x29 **at run time!**

```
add  reg32/mem32, reg32   opcode: 0x01

sub  reg32/mem32, reg32   opcode: 0x29
```



```
CS@kntu:lecture_reveng$ ./run.sh selfmod2
./run.sh: line 5: 25040 Segmentation fault
```

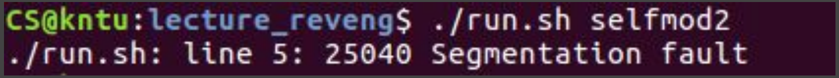
Self-modifying code

```
segment .text        ← Read Only!

global asm_main

asm_main:

        pusha
        ;; ============================

        mov  byte [label1], 0x29

        mov eax, 100
        mov ebx, 2
label1:
        add eax, ebx


        call print_int
        call print_nl
```

selfmod2.asm

The idea: change the opcode 0x01 to 0x29 at run time!

```
add reg32/mem32, reg32   opcode: 0x01

sub reg32/mem32, reg32   opcode: 0x29
```



```
CS@kntu:lecture_reveng$ ./run.sh selfmod2
./run.sh: line 5: 25040 Segmentation fault
```

Self-modifying code

```
segment .data

global asm_main

asm_main:

        pusha
        ;; ============================

        mov  byte [label1], 0x29

        mov eax, 100
        mov ebx, 2
label1:
        add eax, ebx


        call print_int
        call print_nl
```

selfmod3.asm

The idea: change the opcode 0x01 to 0x29 **at run time!**

```
add reg32/mem32, reg32   opcode: 0x01

sub reg32/mem32, reg32   opcode: 0x29
```

Self-modifying code

```
segment .data

global asm_main

asm_main:

        pusha
        ;; ============================

        mov  byte [label1], 0x29

        mov eax, 100
        mov ebx, 2
label1:
        add eax, ebx


        call print_int
        call print_nl
```

selfmod3.asm

The idea: change the opcode 0x01 to 0x29 **at run time!**

```
add reg32/mem32, reg32   opcode: 0x01

sub reg32/mem32, reg32   opcode: 0x29
```

Self-modifying code

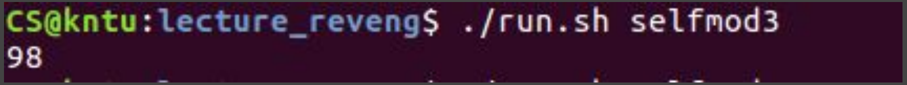
K. N. Toosi
University of Technology

```
segment .data

global asm_main

asm_main:

        pusha
        ;; ============================

        mov  byte [label1], 0x29

        mov eax, 100
        mov ebx, 2
label1:
        add eax, ebx


        call print_int
        call print_nl
```

selfmod3.asm

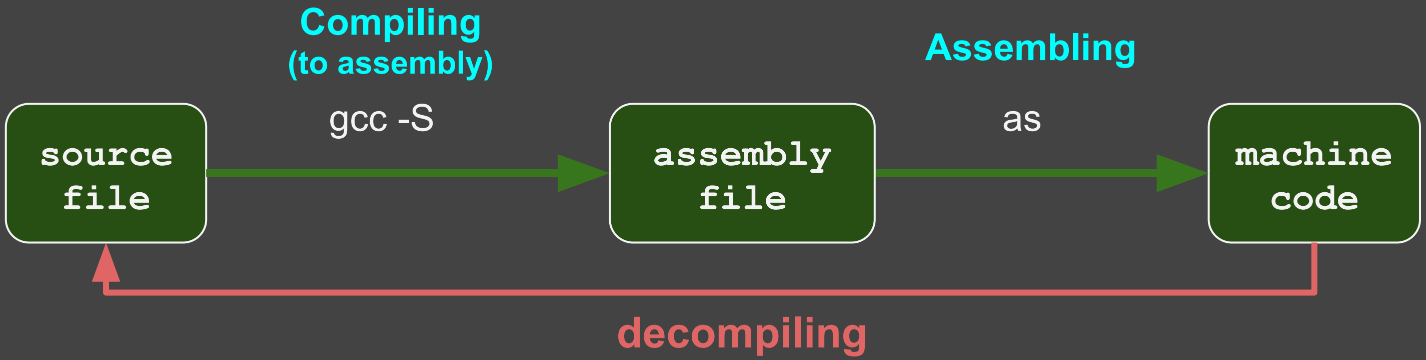
The idea: change the opcode 0x01 to 0x29 **at run time!**

```
add reg32/mem32, reg32   opcode: 0x01

sub reg32/mem32, reg32   opcode: 0x29
```

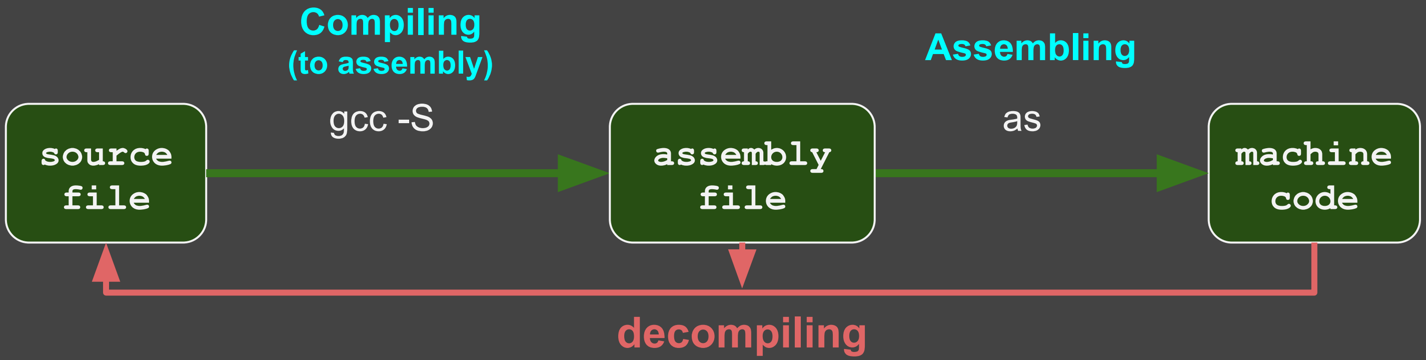


```
cs@kntu:lecture_reveng$ ./run.sh selfmod3
98
```

Decompilers

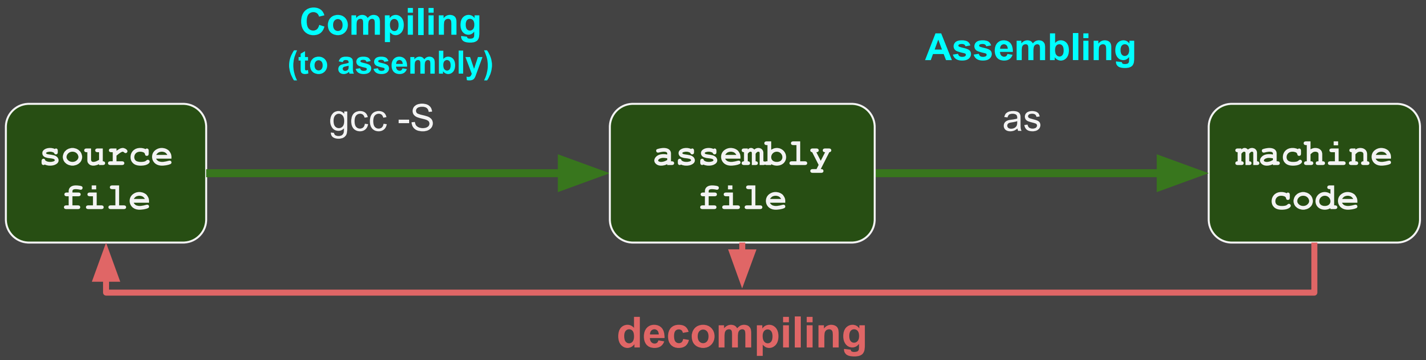


Decompilers



Decompilers

K. N. Toosi
University of Technology



- Many-to-many mapping between source and assembly codes
- Usually cannot generate original source code
- Optimizing/Obfuscating code make decompiling even harder
- Usually produce complex code

Complete Reverse Engineering Frameworks

K. N. Toosi
University of Technology

- IDA-Pro
- Radare2 (open source)
- OllyDbg
- Hopper
- binary ninja
- :

References

K. N. Toosi
University of Technology

- Introduction to Reverse Engineering Software in Linux
- https://medium.com/@rickharris_dev/reverse-engineering-using-linux-gdb-a99611ab2d32
- https://www.linux.com/blog/4-ways-password-could-be-hacked-using-common-linux-tools
- https://en.wikibooks.org/wiki/X86_Disassembly/Disassemblers_and_Decompilers
- https://www.youtube.com/watch?v=a2EkORFcSZo
- https://en.wikibooks.org/wiki/X86_Dis

References

- patching:
  - https://reverseengineering.stackexchange.com/questions/15042/making-changes-in-elf-file-after-dissassembly
  - https://superuser.com/questions/283008/binary-diff-patch-for-large-files-on-linux
  - https://stackoverflow.com/questions/1945075/how-do-i-create-binary-patches