Homework 6

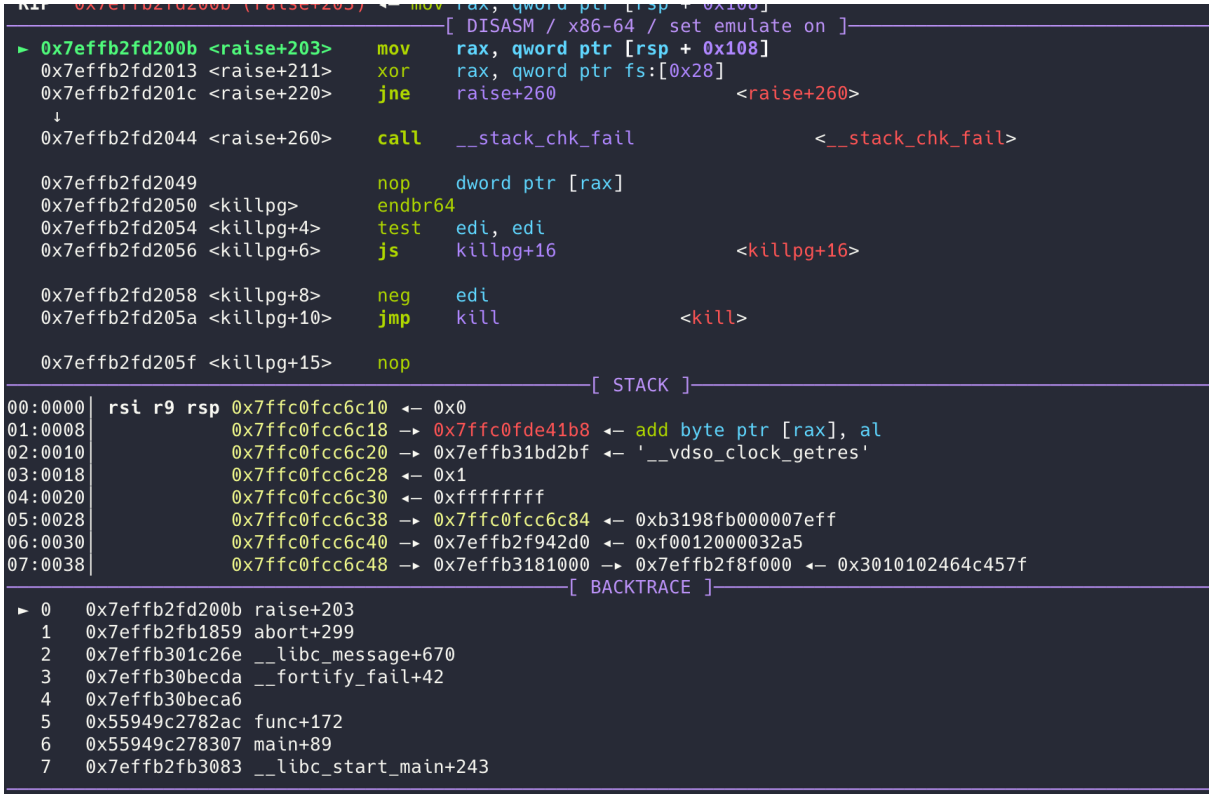
Lim Saeyeon (#21102054)

```
CS21102054@nshcdell ~/HW6 (0.021s)
ls
core.login.281402  login  login.c

CS21102054@nshcdell ~/HW6 (0.026s)
perl -e 'print "A"x32' | ./login
Key : Login FAILED!!
*** stack smashing detected ***: terminated
Aborted
```

```
CS21102054@nshcdell ~/HW6
gdb ./login core.login.281402
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.2) 9.2
Copyright (C) 2020 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"...
pwndbg: loaded 147 pwndbg commands and 47 shell commands. Type pwndbg [--shell | --all] [filter] for a list.
pwndbg: created $rebase, $ida GDB functions (can be used with print/break)
Reading symbols from ./login...
(No debugging symbols found in ./login)
[New LWP 281402]
Core was generated by `./login'.
Program terminated with signal SIGABRT, Aborted.
#0  __GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:50
50      ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
------- tip of the day (disable with set show-tips off) -------
GDB's follow-fork-mode parameter can be used to set whether to trace parent or child after fork() calls
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────────────
 RAX  0x0
 RBX  0x7effb3182540 ◂— 0x7effb3182540
 RCX  0x7effb2fd200b (raise+203) ◂— mov rax, qword ptr [rsp + 0x108]
 RDX  0x0
 RDI  0x2
 RSI  0x7ffc0fcc6c10 ◂— 0x0
 R8   0x0
 R9   0x7ffc0fcc6c10 ◂— 0x0
 R10  0x8
 R11  0x246
 R12  0x7ffc0fcc6e90 ◂— 0x1
 R13  0x20
 R14  0x7effb31c4000 ◂— 0x202a2a2a00001000
 R15  0x1
 RBP  0x7ffc0fcc6f90 —▸ 0x7effb314608f ◂— '*** %s ***: terminated\n'
 RSP  0x7ffc0fcc6c10 ◂— 0x0
 RIP  0x7effb2fd200b (raise+203) ◂— mov rax, qword ptr [rsp + 0x108]
──────────────────────────[ DISASM / x86-64 / set emulate on ]──────────────────────────
```


[DISASM / x86-64 / set emulate on]
► 0x7effb2fd200b <raise+203> mov rax, qword ptr [rsp + 0x108]
0x7effb2fd2013 <raise+211> xor rax, qword ptr fs:[0x28]
0x7effb2fd201c <raise+220> jne raise+260 <raise+260>
0x7effb2fd2044 <raise+260> call __stack_chk_fail <__stack_chk_fail>
0x7effb2fd2049 nop dword ptr [rax]
0x7effb2fd2050 <killpg> endbr64
0x7effb2fd2054 <killpg+4> test edi, edi
0x7effb2fd2056 <killpg+6> js killpg+16 <killpg+16>
0x7effb2fd2058 <killpg+8> neg edi
0x7effb2fd205a <killpg+10> jmp kill <kill>
0x7effb2fd205f <killpg+15> nop
[STACK]
00:0000 rsi r9 rsp 0x7ffc0fcc6c10 ◂— 0x0
01:0008 0x7ffc0fcc6c18 —▸ 0x7ffc0fde41b8 ◂— add byte ptr [rax], al
02:0010 0x7ffc0fcc6c20 —▸ 0x7effb31bd2bf ◂— '__vdso_clock_getres'
03:0018 0x7ffc0fcc6c28 ◂— 0x1
04:0020 0x7ffc0fcc6c30 ◂— 0xffffffff
05:0028 0x7ffc0fcc6c38 —▸ 0x7ffc0fcc6c84 ◂— 0xb3198fb000007eff
06:0030 0x7ffc0fcc6c40 —▸ 0x7effb2f942d0 ◂— 0xf0012000032a5
07:0038 0x7ffc0fcc6c48 —▸ 0x7effb3181000 —▸ 0x7effb2f8f000 ◂— 0x3010102464c457f
[BACKTRACE]
► 0 0x7effb2fd200b raise+203
1 0x7effb2fb1859 abort+299
2 0x7effb301c26e __libc_message+670
3 0x7effb30becda __fortify_fail+42
4 0x7effb30beca6
5 0x55949c2782ac func+172
6 0x55949c278307 main+89
7 0x7effb2fb3083 __libc_start_main+243

The program terminated with a SIGABRT (signal 6), which generally indicates that the progra m aborted intentionally, often due to an assertion failure, buffer overflow, or some other critic al error that could not be handled safely. The backtrace reveals the function calls that led to this crash.

1. raise

- A system call that sends a signal to the program itself, in this case, a SIGABRT.

2. abort

- The `abort` function is called, which sends SIGABRT to terminate the program.

3. __libc_message

- This function is part of the C library (glibc) and indicates an internal message generated du ring a critical error.

4. __fortify_fail

- This function is invoked when a fortified check in glibc detects a buffer overflow or simila r violation of memory safety.

5. func

- The `func` function in the program called some code that violated a memory boundary, trig gering the fortification error.

6. main

- The main function initiated the call to `func`.

The disassembly shows that the crash occurred during a stack check failure. This is indicated by the call to `__stack_chk_fail` after a failed buffer boundary check, which is consistent wit h the `__fortify_fail` message in the backtrace. The RIP register, which stores the instruction pointer, indicates the program was executing `raise` in the `__GI_raise` function when the cra sh occurred.

The crash is due to a buffer overflow in the `func` function.

- The function **`func`** attempts to read more data than the buffer can hold. The `read` fu nction call likely exceeds the buffer's allocated space, causing memory outside of the buffer t o be overwritten.

- This memory overwrite triggers a security feature (buffer fortification) in the C library, desi gned to prevent stack-based buffer overflows from leading to unpredictable behavior. This che ck fails, and the program calls `abort` to terminate itself for safety.

The core dump analysis indicates a buffer overflow vulnerabilit in the code, where data writte n to a buffer exceeds its allocated size. This overflow results in a critical security violation, which leads the program to abort via `SIGABRT`.