In-class practice 4

● Graded

Student

HARRY KIM

Total Points

100 / 100 pts

Question 1

Practice 4

100 / 100 pts

- ✔ **+ 10 pts** Run the program WITHOUT using gdb and screenshot the execution results
- ✔ **+ 10 pts** Start the program using gdb (it will not run the program; instead, it will wait for your command)
- ✔ **+ 10 pts** Set up a breakpoint at the "main" function and then run the program; When the execution hits "main" function, take a screenshot
- ✔ **+ 10 pts** Disassemble the "main" function and take a screenshot
- ✔ **+ 10 pts** Execute "main" function in single-statement mode (i.e., run the main function statement by statement) until it returns (it should return at line 20 of the source code file); Take a screenshot right before it going to return
- ✔ **+ 10 pts** When you complete steps 2-6, you will see the execution in gdb has a different result from the execution without gdb. Specifically, when you run the program without gdb, the execution will return at line 27. However, if you run the program with gdb, it will return line 20.

  In this step, you are required to repeat steps 2-4. At step 4, you are required to set up a breakpoint at the entry address of the "child" function (please set up the breakpoint at the address of "child", instead of using the name "child"; you can disassemble the "main" function to learn about the address of the "child" function)
- ✔ **+ 10 pts** After step 7, you need to somehow make the execution to take the false branch of the conditional statement at line 17, so that the execution will go to the "child" function instead of return at line 20. THIS IS THE MOST CHALLENGING PART IN THIS PRACTICE -:) You are encouraged to do some research together with your classmates to figure this part out. In this step, you are not allowed to change the source code of the program and recompile the program. When you hit the "child" function, take a screenshot
- ✔ **+ 10 pts** When the execution hits the "child" function, you need to print the stack trace and take a screenshot (+10 points); you are also required to print the stack frame of the "child" function and take a screenshot
- ✔ **+ 10 pts** When the execution hits the "child" function, you are required to execute the function in single-instruction mode (i.e., executing the function instruction by instruction) until it tries to call the "rand" function. Take a screenshot when it tries to call the "rand" function
- ✔ **+ 10 pts** Before you return from the "child" function, you need to print the value in the variable named "secret" and take a screenshot

Question assigned to the following page: 1

Harry Kim
u1226472

CS 4440: Practice 4

Step 1:

- I downloaded and compiled the program.

```
[u1226472@lab2-5 cs4440practice4]$ gcc -g -O0 malware.c -o malware
```

Step 2:

- I ran the malware executable without using gdb and got the following result:

```
[u1226472@lab2-5 cs4440practice4]$ ./malware
hey, you passed the challenge
[u1226472@lab2-5 cs4440practice4]$
```

Step 3:

- I started the program using gdb.

```
[u1226472@lab2-5 cs4440practice4]$ gdb ./malware
GNU gdb (GDB) 9.1
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-pc-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"...
--Type <RET> for more, q to quit, c to continue without paging--
Reading symbols from ./malware...
(gdb)
```

Step 4:

- I set up a breakpoint at the “main” function and ran the program. It stopped at line 17 at main where I went up the breakpoint as expected.

```
(gdb) break main
Breakpoint 1 at 0x400649: file malware.c, line 17.
(gdb) run
Starting program: /home/u1226472/Desktop/cs4440practice4/malware

Breakpoint 1, main (argc=1, argv=0x7fffffffdab8) at malware.c:17
17              if (ptrace(PTRACE_TRACEME, 0, 1, 0) == -1)
(gdb)
```

Question assigned to the following page: 1

Harry Kim
u1226472

Step 5:

- I disassembled the “main” function with the command *disassem main* and got the following:

```
(gdb) disassem main
Dump of assembler code for function main:
   0x000000000040063a <+0>:     push   %rbp
   0x000000000040063b <+1>:     mov    %rsp,%rbp
   0x000000000040063e <+4>:     sub    $0x10,%rsp
   0x0000000000400642 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000000000400645 <+11>:    mov    %rsi,-0x10(%rbp)
=> 0x0000000000400649 <+15>:    mov    $0x0,%ecx
   0x000000000040064e <+20>:    mov    $0x1,%edx
   0x0000000000400653 <+25>:    mov    $0x0,%esi
   0x0000000000400658 <+30>:    mov    $0x0,%edi
   0x000000000040065d <+35>:    mov    $0x0,%eax
   0x0000000000400662 <+40>:    callq  0x400510 <ptrace@plt>
   0x0000000000400667 <+45>:    cmp    $0xffffffffffffffff,%rax
   0x000000000040066b <+49>:    jne    0x40067e <main+68>
   0x000000000040066d <+51>:    mov    $0x400738,%edi
   0x0000000000400672 <+56>:    callq  0x400500 <puts@plt>
   0x0000000000400677 <+61>:    mov    $0x1,%eax
   0x000000000040067c <+66>:    jmp    0x400697 <main+93>
   0x000000000040067e <+68>:    mov    $0x0,%eax
   0x0000000000400683 <+73>:    callq  0x400616 <child>
   0x0000000000400688 <+78>:    mov    $0x400758,%edi
   0x000000000040068d <+83>:    callq  0x400500 <puts@plt>
   0x0000000000400692 <+88>:    mov    $0x0,%eax
   0x0000000000400697 <+93>:    leaveq
   0x0000000000400698 <+94>:    retq
End of assembler dump.
(gdb)
```

Step 6:

- I executed the “main” function step by step until it returned on line 20.

```
Breakpoint 1, main (argc=1, argv=0x7fffffffdab8) at malware.c:17
17              if (ptrace(PTRACE_TRACEME, 0, 1, 0) == -1)
(gdb) step
19              printf("hey, please stop debugging me!!\n");
(gdb)
hey, please stop debugging me!!
20              return 1;
```

Question assigned to the following page: 1

Harry Kim
u1226472

Step 7:

- Set up the breakpoint at the address of the child function by finding it from the disassemble main step.

```
(gdb) b * 0x0000000000400683
Breakpoint 3 at 0x400683: file malware.c, line 23.
```

At this point, I stopped using the virtual machine and switched to connecting directly into the cade labs through my terminal on my personal computer. This is why the screenshot quality changes.

Step 8:

- I first got the address of the call to ptrace through the disassemble main step and put a breakpoint there. I then ran the code until it hit said breakpoint at which point I used the command *info registers* to find which register ptrace was being stored at. I then imputed the command *set &rax += 2* to change the ptrace variable, continued, and stopped at the breakpoint at the child function.

```
Breakpoint 1, main (argc=1, argv=0x7fffffffe318) at malware.c:17
17              if (ptrace(PTRACE_TRACEME, 0, 1, 0) == -1)
(gdb) disassem main
Dump of assembler code for function main:
   0x000000000040063a <+0>:     push   %rbp
   0x000000000040063b <+1>:     mov    %rsp,%rbp
   0x000000000040063e <+4>:     sub    $0x10,%rsp
   0x0000000000400642 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000000000400645 <+11>:    mov    %rsi,-0x10(%rbp)
=> 0x0000000000400649 <+15>:    mov    $0x0,%ecx
   0x000000000040064e <+20>:    mov    $0x1,%edx
   0x0000000000400653 <+25>:    mov    $0x0,%esi
   0x0000000000400658 <+30>:    mov    $0x0,%edi
   0x000000000040065d <+35>:    mov    $0x0,%eax
   0x0000000000400662 <+40>:    callq  0x400510 <ptrace@plt>
   0x0000000000400667 <+45>:    cmp    $0xffffffffffffffff,%rax
   0x000000000040066b <+49>:    jne    0x40067e <main+68>
   0x000000000040066d <+51>:    mov    $0x400738,%edi
   0x0000000000400672 <+56>:    callq  0x400500 <puts@plt>
   0x0000000000400677 <+61>:    mov    $0x1,%eax
   0x000000000040067c <+66>:    jmp    0x400697 <main+93>
   0x000000000040067e <+68>:    mov    $0x0,%eax
   0x0000000000400683 <+73>:    callq  0x400616 <child>
   0x0000000000400688 <+78>:    mov    $0x400758,%edi
   0x000000000040068d <+83>:    callq  0x400500 <puts@plt>
   0x0000000000400692 <+88>:    mov    $0x0,%eax
   0x0000000000400697 <+93>:    leaveq
   0x0000000000400698 <+94>:    retq
End of assembler dump.
(gdb) b *0x0000000000400667
Breakpoint 2 at 0x400667: file malware.c, line 17.
(gdb) c
Continuing.
```

Question assigned to the following page: 1

Harry Kim
u1226472

```
Breakpoint 2, main (argc=1, argv=0x7fffffffe318) at malware.c:17
17              if (ptrace(PTRACE_TRACEME, 0, 1, 0) == -1)
(gdb) disassem main
Dump of assembler code for function main:
   0x000000000040063a <+0>:     push   %rbp
   0x000000000040063b <+1>:     mov    %rsp,%rbp
   0x000000000040063e <+4>:     sub    $0x10,%rsp
   0x0000000000400642 <+8>:     mov    %edi,-0x4(%rbp)
   0x0000000000400645 <+11>:    mov    %rsi,-0x10(%rbp)
   0x0000000000400649 <+15>:    mov    $0x0,%ecx
   0x000000000040064e <+20>:    mov    $0x1,%edx
   0x0000000000400653 <+25>:    mov    $0x0,%esi
   0x0000000000400658 <+30>:    mov    $0x0,%edi
   0x000000000040065d <+35>:    mov    $0x0,%eax
   0x0000000000400662 <+40>:    callq  0x400510 <ptrace@plt>
=> 0x0000000000400667 <+45>:    cmp    $0xffffffffffffffff,%rax
   0x000000000040066b <+49>:    jne    0x40067e <main+68>
   0x000000000040066d <+51>:    mov    $0x400738,%edi
   0x0000000000400672 <+56>:    callq  0x400500 <puts@plt>
   0x0000000000400677 <+61>:    mov    $0x1,%eax
   0x000000000040067c <+66>:    jmp    0x400697 <main+93>
   0x000000000040067e <+68>:    mov    $0x0,%eax
   0x0000000000400683 <+73>:    callq  0x400616 <child>
   0x0000000000400688 <+78>:    mov    $0x400758,%edi
   0x000000000040068d <+83>:    callq  0x400500 <puts@plt>
   0x0000000000400692 <+88>:    mov    $0x0,%eax
   0x0000000000400697 <+93>:    leaveq
   0x0000000000400698 <+94>:    retq
End of assembler dump.
```

Question assigned to the following page: 1

Harry Kim
u1226472

```
(gdb) info registers
rax            0xffffffffffffffff  -1
rbx            0x0                 0
rcx            0x0                 0
rdx            0xffffffffffffff80  -128
rsi            0x0                 0
rdi            0x0                 0
rbp            0x7fffffffe230      0x7fffffffe230
rsp            0x7fffffffe220      0x7fffffffe220
r8             0xffffffff          4294967295
r9             0x7ffff7dcbd20      140737351826720
r10            0x0                 0
r11            0x286               646
r12            0x400530            4195632
r13            0x7fffffffe310      140737488347920
r14            0x0                 0
r15            0x0                 0
rip            0x400667            0x400667 <main+45>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) set $rax += 2
(gdb) info registers
rax            0x1                 1
rbx            0x0                 0
rcx            0x0                 0
rdx            0xffffffffffffff80  -128
rsi            0x0                 0
rdi            0x0                 0
rbp            0x7fffffffe230      0x7fffffffe230
rsp            0x7fffffffe220      0x7fffffffe220
r8             0xffffffff          4294967295
r9             0x7ffff7dcbd20      140737351826720
r10            0x0                 0
r11            0x286               646
r12            0x400530            4195632
r13            0x7fffffffe310      140737488347920
r14            0x0                 0
r15            0x0                 0
rip            0x400667            0x400667 <main+45>
eflags         0x206               [ PF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
(gdb) b *0x0000000000400683
Breakpoint 3 at 0x400683: file malware.c, line 23.
(gdb) c
Continuing.

Breakpoint 3, 0x0000000000400683 in main (argc=1, argv=0x7fffffffe318) at malware.c:23
23          child();
(gdb)
```

Question assigned to the following page: 1

Harry Kim
u1226472

Step 9:

- I printed the stack trace via the command *backtrace child*. I then printed the stack frame via the command *info frame*.

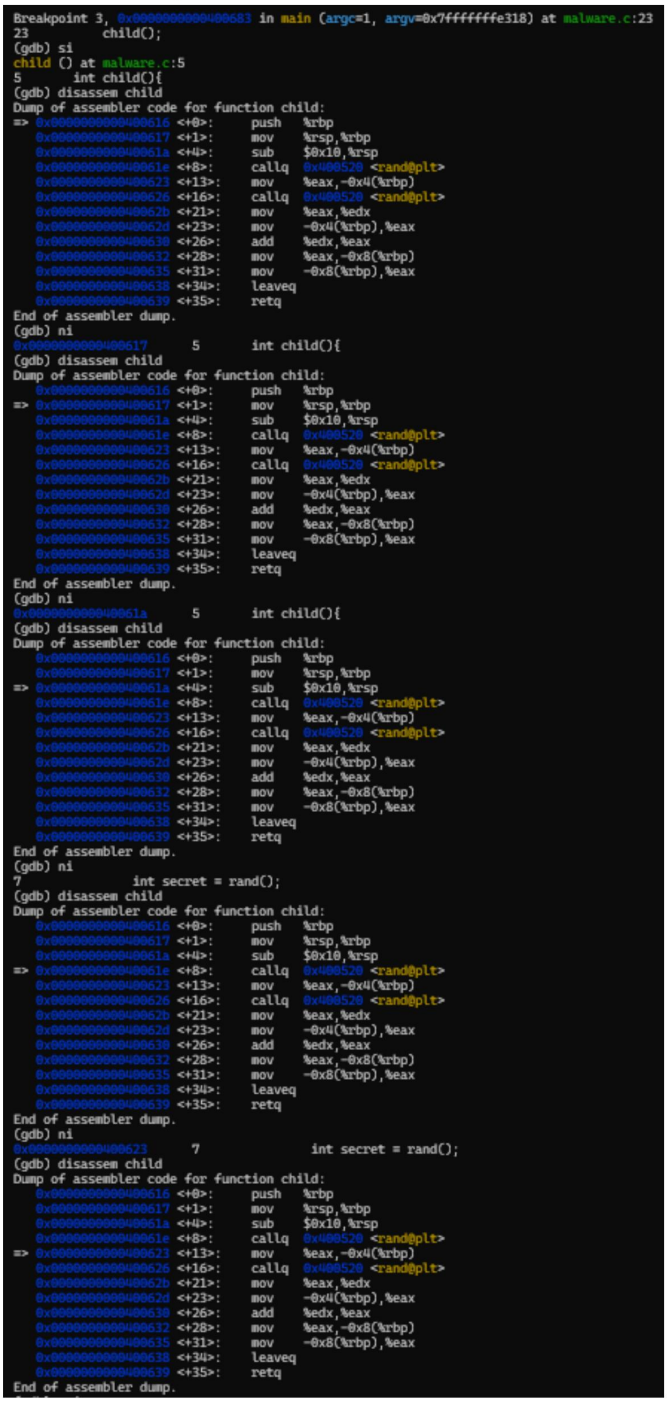
```
child () at malware.c:5
5       int child(){
(gdb) backtrace child
#0  child () at malware.c:5
#1  0x0000000000400688 in main (argc=1, argv=0x7fffffffe318) at malware.c:23
(gdb) infoframe child
Undefined command: "infoframe".  Try "help".
(gdb) info frame
Stack level 0, frame at 0x7fffffffe220:
 rip = 0x400616 in child (malware.c:5); saved rip = 0x400688
 called by frame at 0x7fffffffe240
 source language c.
 Arglist at 0x7fffffffe210, args:
 Locals at 0x7fffffffe210, Previous frame's sp is 0x7fffffffe220
 Saved registers:
  rip at 0x7fffffffe218
(gdb)
```

Question assigned to the following page: 1

Harry Kim
u1226472

Step 10:

- I stepped through child until it was about to call the “rand” function by continuously checking the current instruction via *disassem child*.



Question assigned to the following page: 1

Harry Kim
u1226472

Step 11:

- I printed the value “secret” right before returning from the “child” function. It was a random number as expected.