Advanced Network Security

Assignment - Hackme

Student Name: P. Anojithan

Student Number: MS19814766

Course: M.Sc. Information Technology specialization in Cyber Security

Introduction

This assignment has been done to have some sort of hands on experience in Assembly Language. Here I used Kali Linux to do the practical. I have completed this report with the screenshots I have taken during the lab.

Objective of the lab is to find the password using the knowledge in Assembly language.

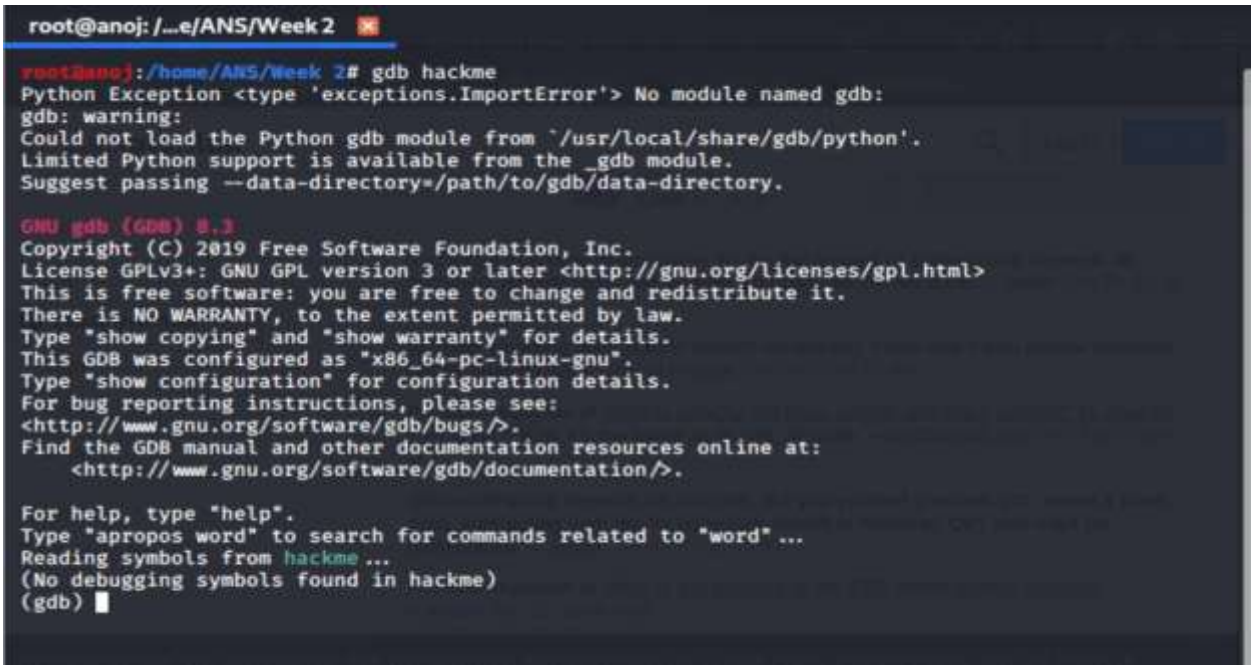
Procedure

Step 1: Navigate the file and run the file.

./hackme

Step 2: using gdb opened the file.

Gdb hackme



Step 3: Disassemble the main function.

disass main

```
(gdb) disass
-function                __cxa_finalize          deregister_tm_clones   memcmp@plt
-label                   __cxa_finalize@plt      exit                   printf
-line                    __do_global_dtors_aux   exit@plt               printf@plt
-probe                   __libc_csu_fini         frame_dummy            puts
-probe-dtrace            __libc_csu_init         gets                   puts@plt
-probe-stap              _fini                   gets@plt               register_tm_clones
-qualified               _init                   main                   secret
-source                  _start                  memcmp
(gdb) disass main
Dump of assembler code for function main:
   0x00000000000011a7 <+0>:     push   %rbp
   0x00000000000011a8 <+1>:     mov    %rsp,%rbp
   0x00000000000011ab <+4>:     sub    $0x40,%rsp
   0x00000000000011af <+8>:     movabs $0x6e69676e33766552,%rax
```

```
root@anoj: /...e/ANS/Week 2

(gdb) disass main
Dump of assembler code for function main:
   0x00000000000011a7 <+0>:     push   %rbp
   0x00000000000011a8 <+1>:     mov    %rsp,%rbp
   0x00000000000011ab <+4>:     sub    $0x40,%rsp
   0x00000000000011af <+8>:     movabs $0x6e69676e33766552,%rax
   0x00000000000011b9 <+18>:    movabs $0x7331676e69726565,%rdx
   0x00000000000011c3 <+28>:    mov    %rax,-0x20(%rbp)
   0x00000000000011c7 <+32>:    mov    %rdx,-0x18(%rbp)
   0x00000000000011cb <+36>:    movl   $0x6c303063,-0x10(%rbp)
   0x00000000000011d2 <+43>:    movw   $0x21,-0x c(%rbp)
   0x00000000000011d8 <+49>:    movl   $0x0,-0x4(%rbp)
   0x00000000000011df <+56>:    lea    0x e79(%rip),%rdi        # 0x205f
   0x00000000000011e6 <+63>:    mov    $0x0,%eax
   0x00000000000011eb <+68>:    callq  0x1040 <printf@plt>
   0x00000000000011f0 <+73>:    lea    -0x40(%rbp),%rax
   0x00000000000011f4 <+77>:    mov    %rax,%rdi
   0x00000000000011f7 <+80>:    mov    $0x0,%eax
   0x00000000000011fc <+85>:    callq  0x1060 <gets@plt>
   0x0000000000001201 <+90>:    lea    -0x20(%rbp),%rcx
   0x0000000000001205 <+94>:    lea    -0x40(%rbp),%rax
   0x0000000000001209 <+98>:    mov    $0x16,%edx
   0x000000000000120e <+103>:   mov    %rcx,%rsi
   0x0000000000001211 <+106>:   mov    %rax,%rdi
   0x0000000000001214 <+109>:   callq  0x1050 <memcmp@plt>
   0x0000000000001219 <+114>:   test   %eax,%eax
```

```
   0x0000000000001219 <+114>:   test   %eax,%eax
   0x000000000000121b <+116>:   jne    0x1224 <main+125>
   0x000000000000121d <+118>:   movl   $0x1,-0x4(%rbp)
   0x0000000000001224 <+125>:   lea    0x e49(%rip),%rdi        # 0x2074
--Type <RET> for more, q to quit, c to continue without paging--
   0x000000000000122b <+132>:   callq  0x1030 <puts@plt>
   0x0000000000001230 <+137>:   cmpl   $0x0,-0x4(%rbp)
   0x0000000000001234 <+141>:   je     0x124c <main+165>
   0x0000000000001236 <+143>:   lea    0x e53(%rip),%rdi        # 0x2090
   0x000000000000123d <+150>:   callq  0x1030 <puts@plt>
   0x0000000000001242 <+155>:   mov    $0x0,%edi
   0x0000000000001247 <+160>:   callq  0x1070 <exit@plt>
   0x000000000000124c <+165>:   lea    0x e5f(%rip),%rdi        # 0x20b2
   0x0000000000001253 <+172>:   callq  0x1030 <puts@plt>
   0x0000000000001258 <+177>:   mov    $0x0,%eax
   0x000000000000125d <+182>:   leaveq
   0x000000000000125e <+183>:   retq
End of assembler dump.
(gdb)
```

Step 4: look for string. Normally the string location may start with something like 0x……. Randomly checked the values (the first 2)

```
(gdb)
(gdb)
(gdb)
(gdb)
(gdb) x/s 0x2074
0x2074: "\nChecking password ... \n"
(gdb) x/s 0x205f
0x205f: "Enter the password! "
(gdb)
```

The value 0x2074 give the text “checking password…” that means the password has been already taken. So checked the value above that 0x205f, which was “Enter the password”. This means something should happen in between this.

x/s 0x2074

x/s 0x205f

Step 5: checked for possible functions left.

disass <tab> <tab> or disass –functions <tab> <tab>

```
(gdb) disass
-function                __cxa_finalize          deregister_tm_clones    memcmp@plt
-label                   __cxa_finalize@plt      exit                    printf
-line                    __do_global_dtors_aux   exit@plt                printf@plt
-probe                   __libc_csu_fini         frame_dummy             puts
-probe-dtrace            __libc_csu_init         gets                    puts@plt
-probe-stap              _fini                   gets@plt                register_tm_clones
-qualified               _init                   main                    secret
-source                  _start                  memcmp
```

From that could see a function “secret”, disassembled that secret function.

disass secret

```
(gdb) disass secret
Dump of assembler code for function secret:
   0x0000000000001175 <+0>:     push   %rbp
   0x0000000000001176 <+1>:     mov    %rsp,%rbp
   0x0000000000001179 <+4>:     lea    0xe88(%rip),%rdi        # 0x2008
   0x0000000000001180 <+11>:    callq  0x1030 <puts@plt>
   0x0000000000001185 <+16>:    lea    0xe9b(%rip),%rdi        # 0x2027
   0x000000000000118c <+23>:    callq  0x1030 <puts@plt>
   0x0000000000001191 <+28>:    lea    0xea0(%rip),%rdi        # 0x2038
   0x0000000000001198 <+35>:    callq  0x1030 <puts@plt>
   0x000000000000119d <+40>:    mov    $0x0,%edi
   0x00000000000011a2 <+45>:    callq  0x1070 <exit@plt>
End of assembler dump.
(gdb)
```

Step 6: Checked for the three values for any readable strings. And found the password at third attempt.

x/s 0x2008

x/s 0x2027

x/s 0x2038

```
(gdb) disass secret
Dump of assembler code for function secret:
   0x0000000000001175 <+0>:     push   %rbp
   0x0000000000001176 <+1>:     mov    %rsp,%rbp
   0x0000000000001179 <+4>:     lea    0xe88(%rip),%rdi        # 0x2008
   0x0000000000001180 <+11>:    callq  0x1030 <puts@plt>
   0x0000000000001185 <+16>:    lea    0xe9b(%rip),%rdi        # 0x2027
   0x000000000000118c <+23>:    callq  0x1030 <puts@plt>
   0x0000000000001191 <+28>:    lea    0xea0(%rip),%rdi        # 0x2038
   0x0000000000001198 <+35>:    callq  0x1030 <puts@plt>
   0x000000000000119d <+40>:    mov    $0x0,%edi
   0x00000000000011a2 <+45>:    callq  0x1070 <exit@plt>
End of assembler dump.
(gdb) x/s 0x2008
0x2008: "You found the secret function!"
(gdb) x/s 0x2027
0x2027: "Congrats!"
(gdb) x/s 0x2032
0x2032: ""
(gdb) x/s 0x2038
0x2038: "The password is: Rev3ngineering1sc00l!"
(gdb)
```

The password was found. The password is: **Rev3ngineering1sc00l!**

Step 7: run the program hackme again, and entered the password.

```
(No debugging symbols found in hackme)
(gdb) disass secret
Dump of assembler code for function secret:
   0x0000000000001175 <+0>:     push   %rbp
   0x0000000000001176 <+1>:     mov    %rsp,%rbp
   0x0000000000001179 <+4>:     lea    0xe88(%rip),%rdi        # 0x2008
   0x0000000000001180 <+11>:    callq  0x1030 <puts@plt>
   0x0000000000001185 <+16>:    lea    0xe9b(%rip),%rdi        # 0x2027
   0x000000000000118c <+23>:    callq  0x1030 <puts@plt>
   0x0000000000001191 <+28>:    lea    0xea0(%rip),%rdi        # 0x2038
   0x0000000000001198 <+35>:    callq  0x1030 <puts@plt>
   0x000000000000119d <+40>:    mov    $0x0,%edi
   0x00000000000011a2 <+45>:    callq  0x1070 <exit@plt>
End of assembler dump.
(gdb) x/s 0x2008
0x2008: "You found the secret function!"
(gdb) x/s 0x2027
0x2027: "Congrats!"
(gdb) x/s 0x2038
0x2038: "The password is: Rev3ngineering1sc00l!"
(gdb) q
root@anoj:/home/ANS/Week 2# ./hackme
Enter the password! Rev3ngineering1sc00l!

Checking password...

Successfully logged in!
Good job!
root@anoj:/home/ANS/Week 2#
```