COMP201 – Assignment 4 Report

Ahmet Koca 0076779

Phase-1

First, I found the address of touch1 with the command "`objdump -d ctarget | grep touch1`". The output was:

```
00000000004018cc <touch1>:
```

So, the address of touch1 is 0x4018cc. Then, inside the gdb, I put a breakpoint on getbuf and disassembled the code there. The output was:

```
Dump of assembler code for function getbuf:
   0x00000000004018b1 <+0>:     push   %rbp
   0x00000000004018b2 <+1>:     mov    %rsp,%rbp
   0x00000000004018b5 <+4>:     sub    $0x20,%rsp
=> 0x00000000004018b9 <+8>:     lea    -0x20(%rbp),%rax
   0x00000000004018bd <+12>:    mov    %rax,%rdi
   0x00000000004018c0 <+15>:    callq  0x401cb0 <Gets>
   0x00000000004018c5 <+20>:    mov    $0x1,%eax
   0x00000000004018ca <+25>:    leaveq
   0x00000000004018cb <+26>:    retq
```

From here, we see that the 0x20 bytes (32 bytes in decimal) is reserved for the stack. Because of the mov %rsp, %rbp command, the return address is located above the saved base pointer. Therefore, I created a 40-byte long input and the address of the touch1. Then, I converted it using hex2raw and fed it to the ctarget.

Phase-2

First, I found the address of touch 2 with the comman "`objdump -d ctarget | grep touch2`". The output was:

```
00000000004018f8 <touch2>:
```

So, the address of touch2 is 0x4018f8. Then, inside the gdb, again, I put a breakpoint on getbuf and disassembled the code there. I continued with nexti until it requires an input from me. I typed a 40-byte long input and check the rsp register right after, using "x/s $rsp". The output was:

```
(gdb) x/s $rsp
0x5566f768:     'a' <repeats 40 times>
```

So, the address of rsp is 0x5566f768. Then, since we want to inject a code that will call touch2 with cookie argument, I created an assembly code for that, then get the byte representation of the code using the commands:

```
[akoca20@linux03 target55]$ cat cookie.txt
0x36d52185
[akoca20@linux03 target55]$ vim phase2.s
[akoca20@linux03 target55]$ gcc -c phase2.s
[akoca20@linux03 target55]$ objdump -d phase2.o > phase2.d
```

Finally, I created a txt file before using hex2raw and feeding it into ctarget, using the following pattern line by line:

Byte representation of the setting an argument as cookie
Padding with 32 bytes of 0s (to make it 40 bytes)
address of register rsp
address of touch2 function.

Phase-3

Similar to the phase 2, we need to pass the cookie as hex numbers to the function touch3. Since hexmatch and strncmp pushes data onto stack, I calculated the address I need to pass as buffer size(32 +8 from previous phases) + 8 bytes + 8 bytes, which is equal to 56, 0x38 in hex. I calculated the relevant address by adding this offset to rsp address:

Rsp before: 0x5566f768, After : 0x5566F7A0

Then, similar to the phase 2, I wrote an assembly code which set rdi register to the address of the string. Then I found the address of touch3 and hex representation of cookie. Finally, I created a txt file before using hex2raw and feeding it into ctarget, using the following pattern line by line:

Byte representation of the setting an argument as cookie (rsp + 56)
Padding with 32 bytes of 0s (to make it 40 bytes)
return address (rsp)
address of touch3 function
cookie string (`33 36 64 35 32 31 38 35`)

Bonus

For this part, I searched for gadgets that I can use. In phase 2, we were passing the cookie as a parameter to touch2, which was stored in rdi register. Therefore, I searched for 5f, which was equivalent to popq %rdi. I could not find it in the specified region, therefore I checked its equivalent version, which is popq %rax and it's representation was 58. I found a 58 in:

```
0000000000401aaa <setval_387>:
  401aaa:       55                      push   %rbp
  401aab:       48 89 e5                mov    %rsp,%rbp
  401aae:       48 89 7d f8             mov    %rdi,-0x8(%rbp)
  401ab2:       48 8b 45 f8             mov    -0x8(%rbp),%rax
  401ab6:       c7 00 58 c3 08 57       movl   $0x5708c358,(%rax)
  401abc:       5d                      pop    %rbp
  401abd:       c3                      retq
```

Since 58 was 2 bytes after 0x401ab6, I hold 0x401ab8. Then, I checked for the equivalent of movq %rax,%edi which was 48 89 c7 c3. I found:

```
0000000000401abe <getval_439>:
  401abe:       55                      push   %rbp
  401abf:       48 89 e5                mov    %rsp,%rbp
  401ac2:       b8 48 89 c7 c3          mov    $0xc3c78948,%eax
  401ac7:       5d                      pop    %rbp
  401ac8:       c3                      retq
```

Since it is 1 byte after 0x401ac2, I hold 0x401ac3. Then, I got the address of Touch2 as 0x401898 and checked the bonus_cookie.txt. Finally, I created the following formatted text file to use hex2raw on and pass to rtartget:

0's of the size of stack (40 bytes in my case)
gadget1: popq %rax
bonus_cookie
gadget2: move %rax, %rdi
address of touch2