Nish Patel

Part 3 Documentation

Methodology and Thought Process:

Buffer Overflow: To successfully exploit the buffer overflow vulnerability – `strcpy()` in `test()` – I used the `gdb` debugger to examine the binary and determine the proper payload to overwrite the return address of `test()`. Using a breakpoint at the end of `test()`, I checked the memory around the local array `test[17]` to see how large the buffer and gap were for writing the necessary `A`'s in the input. Eventually, with comparing to the usual saved return address, I deduced the number of A's and appended the string with the address of `call open()` in `log_result()`, `0x08048E85`.

The general approach was to use a bunch of letters (in this case, `A`'s) to ultimately overflow the buffer, since `strcpy()` would just 'dump' the whole input into test regardless of the expected size, and ultimately replace the saved return address `($ebp+4)` with the address of `call open()`…

```
0xffffdd9f:    0x41414141    0x41414141    0x41414141    0x41414141
0xffffddaf:    0x41414141    0x41414141    0x41414141    0x08048e85
```

*(Rough snippet; picture may not **exactly** reflect stack frame when working on the project)*

Similar to part 2, multiple values for the function need to be overwritten as well: a pointer to the address for the string, a second argument for `open()` to function as expected, and the designated string `uid_1025_crack_super`. Ultimately, the format of the payload came down to `AAAA...AA <log_result(): call open()> <pointer to string that comes after second arg> <string "uid_1025_crack_super">`. The function arguments appear sequentially, so there were no intermediate `A`'s and the format mentioned earlier was sufficient for planting desired values into the parameters and creating the file with the specified file name.

Additionally, because of differences between the testing environment (*gdb*) and the file host (*cs165-internal*), the payload had to be adjusted for the different stack address. Mainly, the different file path meant changing the pointer address. I incremented the address by 8-byte offsets until reaching the desired result (after adding 30 bytes in my case).

---

*Input Provided to Binary (**Exploit Payload**)*:

```
env -i LINES=23 COLUMNS=76 SHELL=/bin/bash /home/admin/try_me
AAAAAAAAAAAAAAAAAAAAAAAAAAAA$'\x85\x8E\x04\x08\xD8\xDD\xFF\xFF\x40\x0
4\x01\x01\x75\x69\x64\x5F\x31\x30\x32\x35\x5F\x63\x72\x61\x63\x6B\x5F\
x73\x75\x70\x65\x72\x00'
```