Week 11 – Heaps of Fun

Introduction to Offensive Security

Administrivia

- No class next week (Thanksgiving)
  - HW that's out today will be due at the start of next class (6p Wednesday *after* Thanksgiving)
- The CTF writeup was not counted as part of the midterm grade
  - Would pull the majority of people's grades down
- RC3 is this weekend
  - DO THE WRITEUP!!1
  - Their challenges are probably easier than the last homework…

Homework Review

Common Issues

- `sendline(…)` to the `fgets` call
  - `fgets`, `read`, etc. read up to the given number of bytes, or to the first newline, whichever comes first
  - `sendline('A'*0x30)` sends 0x31 bytes!
    - 0x30 A's
    - Newline
  - `fgets` will read the 0x30 A's, but leave the newline in the stdin buffer
  - Then when you ROP to `read()`, the newline is immediately read
- Trying to pivot to the start of the `name` buffer
  - Stack grows down, overwrites GOT
  - Stack continues to grow down, tries to write to RO memory

Common Issues Cont'd

- Jumps need to be to the PLT entry, not the GOT entry
  - Remember, GOT entries are addresses themselves, *not* code

Possible Techniques

- Read over GOT entry
  - 1 chain
  - No re-pivoting
  - Optional libc magic (only 9 QWORDS for the full exploit!)
- Read a second ROP chain in just after the first
  - No re-pivoting
- Read a second ROP chain in somewhere else in `name`
  - Have to re-pivot

Demos

The Heap

What is the Heap?

- Another section in memory
  - ASLR
  - RW
- Dynamically allocated through `malloc()` and `free()`
- Potential issues:
  - Not freeing an allocation – memory leak
  - Double freeing an allocation – possibly exploitable
  - Using a free'd allocation – very possibly exploitable

Features of Modern Heaps

- Low overhead allocations
- Sequential free'd regions are coalesced into 1 large free region
- "Best fit" against previously free'd allocations
- Memory is not zeroed out (for performance)

Heap Demos

Use After Free (UAF)

- *We* are responsible for keeping track if something has been free'd
  - No memory safety in C
- If we forget to NULL out pointers, we have a "dangling pointer"
- This can cause issues!

More Demos

Show and Tell