SOFTWARE
SECURITY

ASSIGNMENT 5

REVELIO

- Very similar to the reversing exercise
- Three stages:
    1. XOR the input string byte-by-byte
    2. Solve a substitution cipher by checking the map in the data section
    3. Traverse a binary tree in the correct order using a sequence of `L` and `R`

XOPPER

- Leak libc using `"Get to"`
- Exploit the buffer overflow using ROP
- Your ROP chain is XOR’d with a key (Vigenère-style), so you need to invert that first

UNSORTED

- You can leak the libc as discussed in the lecture (via →`fd` in the unsorted bin)
- You can leak a heap address in the same way
- Reduce the amount of checks by forcing notes into the smallbin
- Overwrite the freed smallbin note's →`bk` pointer (the smallbin is traversed backwards!)
- Allocate a chunk on the stack (the command buffer is large enough to put a fake chunk with valid pointers there)
- Write a ROP chain

HOW DO CHUNKS GET TO THE SMALLBIN?

1. Try to allocate a chunk that is too large for the fastbins (here, using `scanf`)
2. This will walk through the chunks in the unsortedbin
3. If it finds a chunk that is large enough it will use that chunk (and maybe split it)
4. On the way, every chunk that is too small will be sorted into the correct small- or largebin

RAGNAROK

- Lots of things wrong here:

```
// + 1 for \n (stripped by read_line) + 1 for \0
size_t length = strlen(warriors[index]) + 2;

int size = read_size();
printf("Warrior ");
write(STDOUT_FILENO, warriors[index], size);
```

- Unintended solution: just use the off-by-one to keep growing the string
- Intended solution: *House of Einherjar*

RAGNAROK

- Fill the tcache
- Create two consecutive chunks, with a fake chunk in the first chunk
- Clear the `PREV_IN_USE` bit in the second chunk using the overflow
- Consolidation will enqueue the fake chunk
- Free the fake chunk into the tcache
- Overwrite the tcache entry's →`fd` (via the original first chunk, which we never freed) to allocate over the libc GOT.

TINYALLOC

- Basically *ptmalloc* with fewer checks, and pointers instead of chunk sizes
- Single-byte overflow to corrupt a chunk's →`next` pointer

TINYALLOC

- Allocate three consecutive chunks: an overflow chunk, a victim chunk, and a guard chunk to prevent consolidation
- In the victim chunk, place a fake chunk with a huge size (via →`next`) and the `MCF_TOP` flag
- Free and then re-allocate the first chunk to overwrite the low byte of the victim chunk's next pointer (so that it points to the fake chunk instead of the guard chunk)
- Consolidate the victim chunk with the fake chunk to make it the new, huge top chunk
- Allocate a huge chunk to point the new top chunk directly into libc
- Allocate on top of the GOT