Lec06: Bypassing ASLR

Chengyu Song

Slides based on CS6265 taught by Prof. Taesoo Kim
Scoreboard
Administrative

• Due: Lab05 is extended to **Feb 26** at midnight
• Due: Lab06 is out and its due on **Mar 2** at midnight
• 7 normal challenges + 3 bonus
Lab05: ROP

- rop-basic, rop-32, rop-64, pop: 8-10
- upto-retaddr, puzzle, find-gadget, rop-sorting: 4-5
- sprintf, inc1: 3
Discussion: Lab05

- What's the most "interesting" bug or challenge?
- Has DEP makes exploit more difficult?
- ROP in practice isn't that hard, right?
  - Why?
Discussion: puzzle

- Did you choose the hard way or the easy way?
- Don't forget to pull.
Discussion: sprintf

- Let's revisit after lab07
Discussion: rop-sorting

- Which way did you choose? The easier one or the harder one?
Discussion: inc1

- Which memory address to you choose?
Discussion: Lab05

- Would your exploit still work if location of stack and code is randomized?
Address Space Layout Randomization (ASLR)

• Randomize the location of memory content to throttle attacks
  • First developed by PaX team for Linux (better one!)
  • Widely deployed by all OS, some also do kernel ASLR (KASLR)

• Memory content
  • Code: main executable, libraries, etc.
  • Data: stack, heap, memory mapped files, etc.
How does ASLR work?

- Linux
  - Randomize the base of mmap, stack, and heap (brk)
  - Executables are loaded by mmap so their location is also randomized

- Windows
  - Before Windows 8, similar
  - High entropy ASLR, check references
ASLR Weakness 1

• Entropy, entropy, entropy!

• Without enough "randomness", attackers can just guess

• Two attack strategies
  • Brute-force → hacking blind
  • Spray
ASLR Weakness 2

- Not fine-grained

- Remember how you can derive the address of `system()` based on the address of `printf()`
ASLR Weakness 3

- Legacy, not randomizable/randomized content
- Tutorial: position-independent code
ASLR Weakness 4

- Memory disclosure
  - Uninitialized memory
  - Use-after-free
  - Type confusion
  - Heartbleed
  - Arbitrary read
ASLR Weakness 5

- Side-channels
  - Mapped vs. non-mapped
  - Code vs. data
  - Shared cache
Lab06
In-class Tutorial

- PIE: what makes executable re-locatable?
- Dynamic linking: how to attack non-pie code?
- Information leak
References

- Exploit Mitigation Improvements in Windows 8
- Bypassing ASLR
- ELF