esse

CTF Contests: Hands-On Experience of the IT Security Culture – UE 07: Memory Corruptions

Clemens Hlauschek, Daniel Marth, Florian Fankhauser

Memory Corruptions

- Bugs allow targeted manipulation of memory
- May lead to crashes, data corruption or code execution
- Arms race between exploitation and protection techniques

Memory Corruption Tutorial

- https://www.proggen.org/doku.php?id=security:memory-corruption:start
- Created in the context of an ESSE project

Stack-based Buffer Overflows

- Overflow into local variables
- Overwrite function return address

No Operation (NOP) sleds

- Calculating the exact address of a buffer might be challenging
- Prepend NOP instructions as “landing zone”
- Make an educated guess for an address somewhere in the NOP sled

External Buffers

- Environment variables
- Command-line arguments
- May help to exploit small buffers

Return Oriented Programming (ROP)

- No new instructions inserted
- Existing instructions in the executable or linked libraries are reused

return-to-libc

- `libc` widely available
- Overwrite return address with those of `libc` functions like `system`
- Pay attention to the `libc` version!

ROP Chains

- Return into sequence of instructions ("gadget") ending with `ret`
- Combine multiple gadgets to a "chain"
- Stack pointer determines gadget to execute
- `ret` instruction at end of gadgets increments the stack pointer

Heap Corruptions

- Manual memory management is error-prone
- Corrupt the state of the allocator
  - Buffer overflows
  - Use-after-free
  - Double-free

Format String Attacks

- Use format control strings to generate output strings
- Functions: `printf` family (`fprintf`, ...)
- Different format control characters, see `man sprintf`
- Read from memory: `%s`
- Write to memory: `%n`

Thank you!

https://security.inso.tuwien.ac.at/ctf-2022s/