What is a Dynamic Linker + Two Dynamic Linker exploits

Tristan Wiesepape

Basic information + vocab

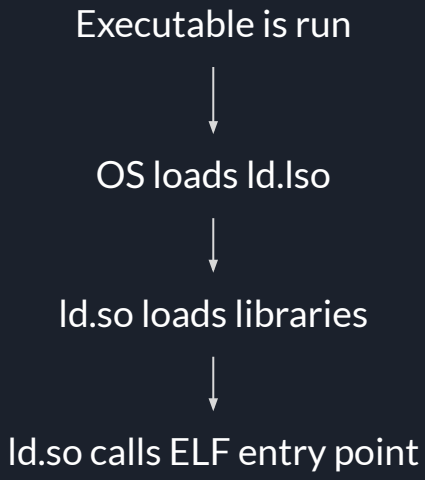
- ELF: The binary format used for linux executables
  - Organises the executable into data, code, symbols, etc
- Shared Library: A collection of data and functions used by multiple programs (i.e. libc)
  - .so files
- Symbol: A name for some data or code in a library (i.e. “gets”)
- Linker: A program in charge of connecting symbol usages to definitions
- Static Linker: All symbol usages are resolved at compile time. Functions used are copy-pasted into the ELF.

What is a dynamic linker

- Allows executables to resolve symbols from shared libraries at runtime
- Multiple programs can rely on the same library
- Static linker runs at compile time, dynamic linker runs at run time

ld.so as a executable

- The “Interpreter”
- The first thing run when you exec an ELF
- Finds and loads shared libraries
- “Lazy”



ld.so as a library

- After it calls entry, ld.so sticks around
- Shared library loaded at runtime
- Contains several useful functions
- Not generally called into by programmer code

Dynamic Linking Process Overview

1. ld.so starts
2. ld.so finds and loads all shared libraries into memory
3. ld.so jumps to user code
4. User code calls a function that is dynamically linked
5. User code jumps to PLT
6. PLT jumps to the data in the GOT, which contains a resolver function
7. Resolver function finds address and saves it to the GOT
8. Returns to user code
9. Future calls will skip the resolution step and jump straight to the function

PLT Entry for “gets”

PLT = List of jump instructions

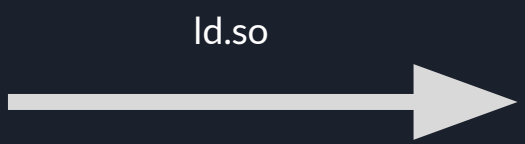
GOT = List of function pointers

The PLT entry for function n jumps to the nth GOT entry

GOT

Before first call to puts

puts	dlresolve
system	dlresolve
gets	dlresolve



After first call to puts

puts	Address of puts
system	dlresolve
gets	dlresolve

GOT/PLT Attacks

- GOT is at a static address
- GOT is writeable
- We can overwrite a GOT entry for one function with another
- Can be used to turn a small arbitrary write (even just 1 byte) into RCE

```
while(true) {
    fgets(buf, 100, stdin);
    printf(buf);
}
```

ret2dlresolve

- There exists a function in ld.so that takes a symbol and returns its address
- Must be loaded at a fixed address
- Allows us to defeat ASLR and find libc without any info leaks
- Allows us to turn a buffer overflow into RCE without any other functions
- We have to construct fake ELF sections to use as arguments to the function
- Pwntools handles this for us

Problems

- Get my GOT from ForeverCTF
- Resolve from UTCTF 2021 (available on ForeverCTF)