Intercepting system calls with LD_PRELOAD

osterlund.xyz/posts/2018-03-12-interceptiong-functions-c.html

Posted on March 12, 2018

Tags: compilers, c, linux

Recently I needed to store some metadata about files read in a program I had. Altering all the `fread` calls seemed laborous, and what if I didn't have the source code? Luckily it is possible to intercept calls to dynamically linked libraries by abusing LD_PRELOAD.

By preloading your own shared library, its functions will be linked before the similarly named functions from libC. Then by looking up the actual function using `dlsym(RTLD_NEXT,...)`, we can insert our own instrumentation before and after calls to such functions.

Suppose you have a simple C program that opens a file and reads its content, like so:

```
#include <stdio.h>
int main(int argc, char **argv)
{
    FILE *f;
    char buf[1024];
    f = fopen(argv[1], "rb");
    fread(buf, 1024, 1, f);
    return 0;
}
```

Now, suppose, you want to instrument the program in such a manner that we print the name of the file we read from.

Create a new file `mylib.c`:

```
#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <dlfcn.h>

// Only works on Linux
char *recover_filename(FILE *f)
{
    char fd_path[256];
    int fd = fileno(f);
    sprintf(fd_path, "/proc/self/fd/%d", fd);
    char *filename = malloc(256);
    int n;
    if ((n = readlink(fd_path, filename, 255)) < 0)
        return NULL;
    filename[n] = '\0';
    return filename;
}

size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
{
    size_t (*lfread)(void *, size_t, size_t, FILE*) = dlsym(RTLD_NEXT, "fread");
    char *fname = recover_filename(stream);
    printf("Read from file %s\n", fname);
    free(fname);
    return lfread(ptr, size, nmemb, stream);
}
```

Compile it using

```
cc -fPIC -shared -o mylib.so mylib.c -ldl
```

This will create a shared object called `mylib.so`. You can now re-direct calls to libc's `fread` to your own library using LD_PRELOAD.

Execute the program as follows:

```
LD_PRELOAD=./mylib.so ./myprogram filename.txt
```

When the program calls `fread`, the name of the file will now be printed!

The possibilities with this are endless. For example, recently, I had to do taint analysis on all input bytes of a file. I simply initialized the taint-flow analysis on all bytes read using `fread` in a similar way to this. Easy!