System Security (201700086)
Secure Systems Engineering #2 - Add seccomp rules

Æde Symen Hoekstra, Luigi Coniglio (Group 24)

June 18, 2018

1 Syscall empolyed

In order to obtain the list of syscalls employed by the child process responsible of handling the query we traced the execution using the command *strace*:

```
$ strace -f ./randomcatserver
```

The listing below shows an example of output of the above command:

```
[pid 16266] close(3)                    = 0
[pid 16218] close(4)                    = 0
[pid 16218] accept(3,  <unfinished ...>
[pid 16266] read(4, "G", 1)             = 1
[pid 16266] read(4, "E", 1)             = 1
[pid 16266] read(4, "T", 1)             = 1
            ...
[pid 16266] read(4, "\n", 1)            = 1
[pid 16266] write(1, "Random cat\n", 11) = 11
[pid 16266] write(4, "HTTP/1.0 200 OK\r\nContent-Type: i"..., 64) = 64
[pid 16266] open("images", O_RDONLY)    = 3
[pid 16266] fstat(3, {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
[pid 16266] fcntl(3, F_GETFL)           = 0x8000 (flags O_RDONLY|O_LARGEFILE)
[pid 16266] fcntl(3, F_SETFD, FD_CLOEXEC) = 0
[pid 16266] brk(0)                      = 0x169f000
[pid 16266] brk(0x16c8000)              = 0x16c8000
[pid 16266] getdents(3, /* 23 entries */, 32768) = 1104
[pid 16266] getdents(3, /* 0 entries */, 32768) = 0
[pid 16266] openat(3, "pexels-photo-326875.jpeg", O_RDONLY) = 5
[pid 16266] brk(0x16c0000)              = 0x16c0000
[pid 16266] close(3)                    = 0
[pid 16266] read(5, "\377\330\377\340\0\20JFIF\0\1\1\1"..., 4096) = 4096
[pid 16266] write(4, "\377\330\377\340\0\20JFIF\0\1\1\1"..., 4096) = 4096
            ...
[pid 16266] read(5, "\224\17\367/&\367\30\366\253vWV"..., 4096) = 1662
```

```
[pid 16266] write(4, "\224\17\367/&\367\30\366\253vWV"..., 1662) = 1662
[pid 16266] read(5, "", 4096)           = 0
[pid 16266] shutdown(4, SHUT_WR)        = 0
[pid 16266] close(4)                    = 0
[pid 16266] exit_group(0)               = ?
[pid 16266] +++ exited with 0 +++
```

Those are the syscalls executed by the child process serving a GET request for the resource */randomcat* .

As we can see there server uses the *read* and *write* syscalls to communicate with the client and read the image, the syscall *open* to open the images directory and the cat image, the syscalls *fstat* and *fcntl* to retrieve some informations about the images directory, the syscall *brk* to allocate some memory, the syscall *getdents* to get what files are inside the images directory, *openat* to open the image file, *close* to close the file descriptors not longer needed, *shutdown* to shut down the socket and *exit_group* to terminate the execution.

2 Adding seccomp rules

In order to stop the process from using different syscall than those specified in the previous section, we execute the following function before the core operations:

```c
int setup_seccomp() {
    scmp_filter_ctx ctx;
    ctx = seccomp_init(SCMP_ACT_KILL);
    int calls[] = {SCMP_SYS(close),    SCMP_SYS(read),      SCMP_SYS(write),
                   SCMP_SYS(open),     SCMP_SYS(fstat),     SCMP_SYS(fcntl),
                   SCMP_SYS(brk),      SCMP_SYS(getdents),  SCMP_SYS(openat),
                   SCMP_SYS(shutdown), SCMP_SYS(exit_group)};
    int calls_length = sizeof(calls) / sizeof(calls[0]);
    int i;

    if (ctx == NULL) {
        return -1;
    }

    for (i = 0; i < calls_length; i++) {
        if (seccomp_rule_add(ctx, SCMP_ACT_ALLOW, calls[i], 0) < 0) {
            seccomp_release(ctx);
            fprintf(stderr, "adding rule %d failed \n", i);
            return -1;
        }
    }

    if (seccomp_load(ctx) < 0) {
        seccomp_release(ctx);
        fprintf(stderr, "loading seccomp failed\n");
```

```
26            return -1;
27        }
28        return 0;
29    }
```