Linux' receive_fd_replace() semantics confusing

Tycho Andersen <tandersen@netflix.com>
Alok Tiagi <atiagi@netflix.com>

We want to intercept connect()

```
BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K, __NR_connect, 0, 1),
BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_USER_NOTIF),
```

We want to intercept connect()

```
BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K, __NR_connect, 0, 1),
BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_USER_NOTIF),

And then replace the fd,

ioctl(n, SECCOMP_IOCTL_NOTIF_ADDFD, ...);
```

We want to intercept connect()

```
BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K, __NR_connect, 0, 1),
BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_USER_NOTIF),

And then replace the fd,

ioctl(n, SECCOMP_IOCTL_NOTIF_ADDFD, ...);

Which eventually invokes

receive_fd_replace(...);
/* seccomp is only user */
```

Applications want to…

```
int epfd = epoll_create();
int sock = socket();
```

Applications want to…

```
int epfd = epoll_create();
int sock = socket();

epoll_ctl(epfd, EPOLL_CTL_ADD, sock, ...);
connect(sock, ...);
```

First, epoll in the kernel

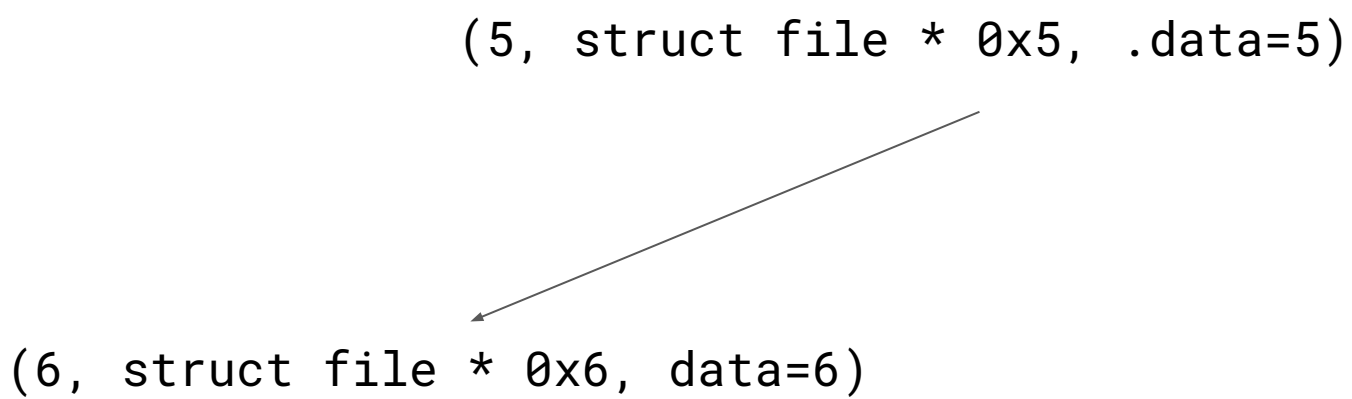
What does epoll do?

```
epoll_ctl(epfd, ADD, sock1 /* 5 */, epoll_data.fd = 5);
```

```
(5, struct file * 0x5, .data=5)
```

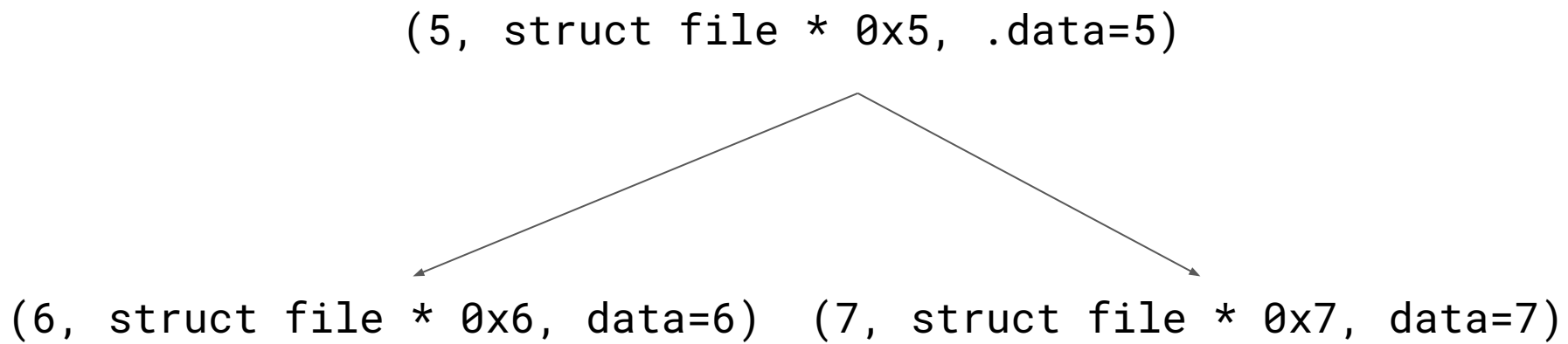
What does epoll do?

```
epoll_ctl(epfd, ADD, sock1 /* 5 */, epoll_data.fd = 5);
epoll_ctl(epfd, ADD, sock2 /* 6 */, epoll_data.fd = 6);
```



What does epoll do?

```
epoll_ctl(epfd, ADD, sock1 /* 5 */, epoll_data.fd = 5);
epoll_ctl(epfd, ADD, sock2 /* 6 */, epoll_data.fd = 6);
epoll_ctl(epfd, ADD, sock3 /* 7 */, epoll_data.fd = 7);
```



When a socket receives data…

```
epoll_wait(epfd, &event);
// data received on 5
```

When a socket receives data…

```
epoll_wait(epfd, &event);
// data received on 5
event.data == 5
read(5);
// profit
```

Mix in seccomp ADDFD

Applications want to…

```
int epfd = epoll_create();
int sock = socket();

epoll_ctl(epfd, EPOLL_CTL_ADD, sock, ...);
connect(sock, ...);
```

What happens with seccomp ADDFD?

```
connect(5, ...);
```

What happens with seccomp ADDFD?

```
connect(5, ...);
```

```
                                                ioctl(notify, SECCOMP_RECV, ...);
```

What happens with seccomp ADDFD?

```
connect(5, ...);
```

```
ioctl(notify, SECCOMP_RECV, ...);
int newsock = socket(...);
```

What happens with seccomp ADDFD?

```
connect(5, ...);
```

```
ioctl(notify, SECCOMP_RECV, ...);
int newsock = socket(...);
struct seccomp_addfd addfd = {
    .srcfd = newsock /* 0x8 */,
    .newfd = 5,
    .flags = SECCOMP_SETFD,
};
ioctl(notify, SECCOMP_ADDFD, &addfd);
```

What happens with seccomp ADDFD?

```
connect(5, ...);
```

```
ioctl(notify, SECCOMP_RECV, ...);
int newsock = socket(...);
struct seccomp_addfd addfd = {
    .srcfd = newsock /* 0x8 */,
    .newfd = 5,
    .flags = SECCOMP_SETFD,
};
ioctl(notify, SECCOMP_ADDFD, &addfd);
-> receive_fd_replace(
        5, struct file * / 0x8 */);
-> ...
-> rcu_assign_pointer(fdt->fd[fd],
                      file);
```

What happens with seccomp ADDFD?

```
connect(5, ...);
```

```
ioctl(notify, SECCOMP_RECV, ...);
int newsock = socket(...);
struct seccomp_addfd addfd = {
    .srcfd = newsock /* 0x8 */,
    .newfd = 5,
    .flags = SECCOMP_SETFD,
};
ioctl(notify, SECCOMP_ADDFD, &addfd);
-> receive_fd_replace(
        5, struct file * / 0x8 */);
-> ...
-> rcu_assign_pointer(fdt->fd[fd],
                      file);
```

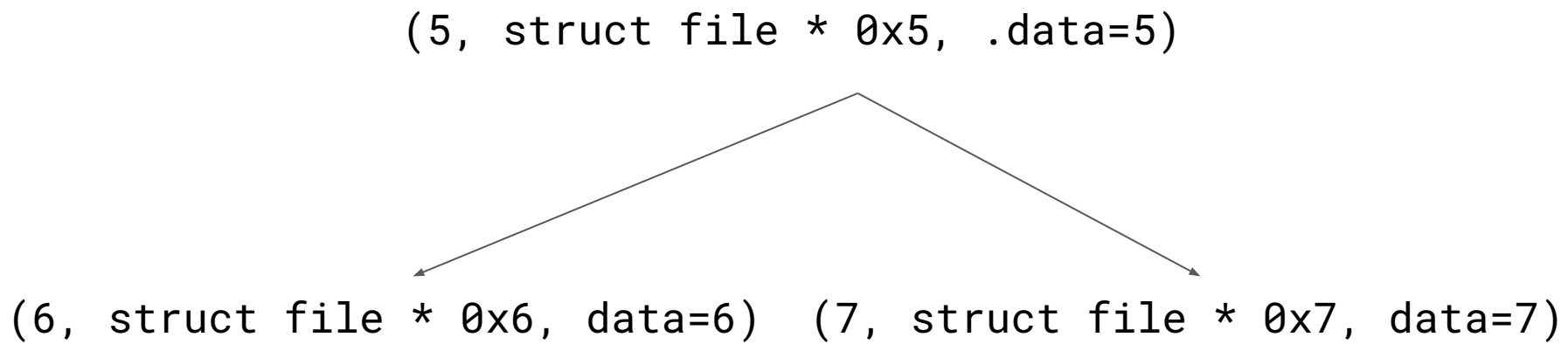
```
read(5); /* reads struct file 0x8 */
```

What doesn’t happen?

1. This is the only copy of the `struct file *`
   a. File is removed from epoll instance since it is closed when replaced via `__fput() -> epoll_release()`
2. There are multiple copies of the `struct file *`
   a. The file *remains* in the epoll instance

What ideally would happen?

```
connect(5, ...);
-> receive_fd_replace(5, struct file /* 0x8 */);
```

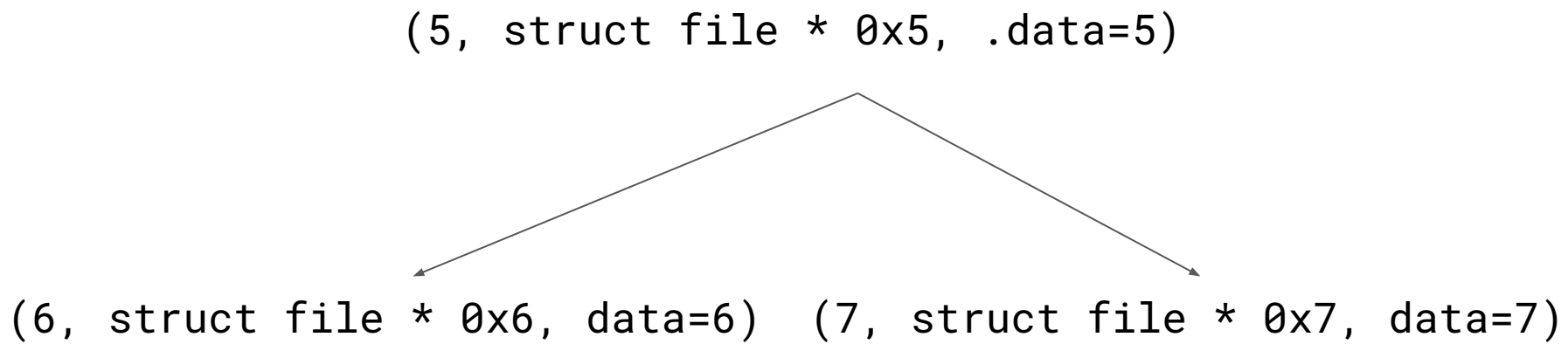


Now you are in a very weird state

```
read(5); /* reads struct file 0x8 */
// reports data on struct file 0x5
epoll_wait(epfd, &event);
event.data == 5;
read(5); /* reads 0x8, sad panda */
```

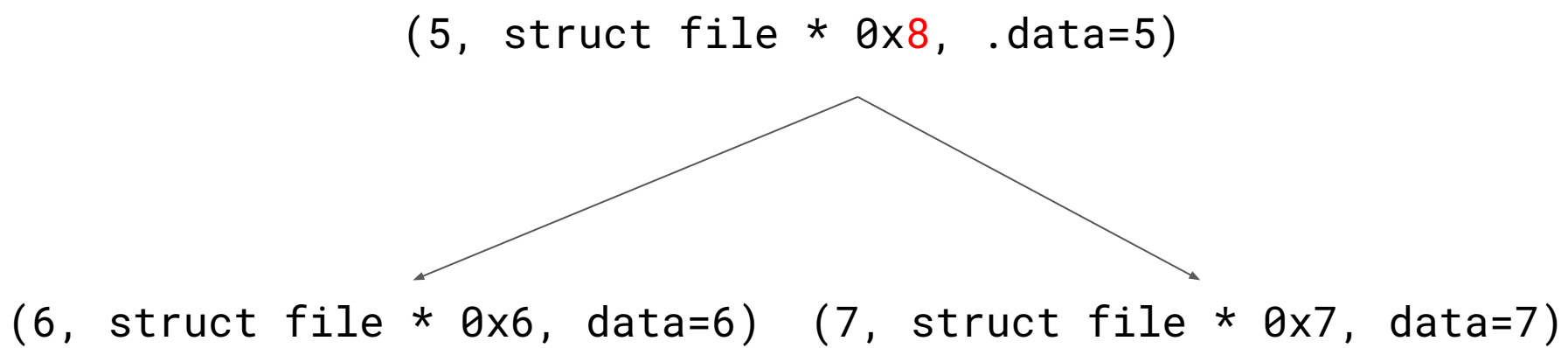
What ideally would happen?

```
connect(5, ...);
-> receive_fd_replace(5, struct file /* 0x8 */);
```



What ideally would happen?

```
connect(5, ...);
-> receive_fd_replace(5, struct file /* 0x8 */);
```



How to fix this?

From userspace (aka The CRIU Way)

```
to_replace = 5
for each potential_epoll_fd:
    look in fdinfo for ‘tfd:’ == to_replace:
        epfd = potential_epoll_fd
        epolls[epfd] = (tfd, data /* also from fdinfo */)
        /* no DEL needed, handled in fput() */
do_replace()
for fd, (tfd, data) in epolls:
    epoll_ctl(ADD, tfd, fd, data)
```

From userspace (aka The CRIU Way)

- Iterating (+parsing strings) for each fd slow
- 40k fds -> 100+ms for a non-blocking connect

With extra fdinfo?

- The kernel knows (via struct file->f_ep) what epolls this file is linked to
- Wouldn't have to do `for each open fd`
- Still requires string slinging
- Everyone who uses this API has to do it
- Fdinfo is currently file type specific, this would add "generic" fields

From receive_replace_fd()?

- Then everyone wouldn’t have to write this code
- Patch series here: https://lore.kernel.org/lkml/20230318060248.848099-1-aloktiagi@gmail.com/
- Layering violation (fdtable touching epoll)
- Seccomp is the only user in the tree (currently) of receive_replace_fd()
- Christian suggests in: https://lore.kernel.org/lkml/20230327090106.zylztuk77vble7ye@wittgenstein/ something like,

From receive_replace_fd()?

```
if (addfd->ioctl_flags & SECCOMP_IOCTL_EPOLL_FIXUP) {
    epoll_seccomp_notify(...);
}
```

- Other users of receive_replace_fd() have to figure this out
- Flags for receive_replace_fd()?
  - Still a layering violation, but at least more obvious to callers what’s going on

Similar problems / future work

- ADDFD replaces file at one fd, what about dup/dup2?
  - Such an REPLACE_ALL_STRUCT_FILE is unavoidably O(fds)
  - Can figure out in userspace via kcmp() in one task
- Other files-of-files (io_uring, select, etc.)

Merci

Tycho Andersen <tandersen@netflix.com>
Alok Tiagi <atiagi@netflix.com>