

MS degree in Computer Engineering
University of Rome Tor Vergata 
Lecturer: Francesco Quaglia

Cross ring data move:

1. Segmentation based protection breaks in data move
2. Kernel level actual data move facilities
3. Kernel level service replication

memcpy vs kernel internals

- Data move between user and kernel level buffers cannot rely on base buffer-management implementations such as `memcpy()`
- The reasons are:
 - ✓ ring based protection
 - ✓ segmentation based addressing
- Particularly, segments that are mapped to the same base are fully accessible while running at ring 0
 - ✓ Check and resolution of discrepancies needs to be carried out at run-time

User/kernel level data move (i)

```
unsigned long copy_from_user(void *to, const  
void *from, unsigned long n)
```

Copies n bytes from the user address(from) to the kernel address space(to).

```
unsigned long copy_to_user(void *to, const void  
*from, unsigned long n)
```

Copies n bytes from the kernel address(from) to the user address space(to).

```
void get_user(void *to, void *from)
```

Copies an integer value from userspace (from) to kernel space (to).

```
void put_user(void *from, void *to)
```

Copies an integer value from kernel space (from) to userspace (to).

User/kernel level data move (ii)

```
long strncpy_from_user(char *dst, const char  
    *src, long count)
```

Copies a null terminated string of at most count bytes long from userspace (src) to kernel space (dst)

```
int access_ok(int type, unsigned long addr,  
    unsigned long size)
```

Returns nonzero if the userspace block of memory is valid and zero otherwise

Main tasks

- Segment fixup (if segmentation takes a real role in the composition of the addresses)
- Check on address ranges related to user level
- **Note:** associating physical to virtual memory is demanded to the page-fault handler
 - Performance impact due to (possible) non-atomicity while finalizing the handling

Service redundancy approaches

- Check e fixup are required only in case we need to link activities across different privilege levels within the ring model (as when calling system calls)
- Particularly, this occurs when the execution semantic crosses the boundaries of individual segments
- Bypassing check e fixup when no crossing of segment boundaries occurs takes place via “service redundancy” (for performance reasons)
- The kernel layer entails an internal API for executing activities that are typically triggered when running in user mode

Classical examples

- `kernel_read()` is a redundancy for `read()`
- `kernel_write()` is a redundancy for `write()`

`read()` – syscall



`sys_read()`



`read()` – file operation
real data movement

This requires
a patch

call from the kernel



`kernel_read()`