Data Structure

Name: inputInfo
Type: Struct
Location of declaration: include/inputstr.h
Location of definition: dix/events.c as global

inputstr.h:

```
typedef struct {
    int numDevices;             /* total number of devices */
    DeviceIntPtr devices;       /* all devices turned on */
    DeviceIntPtr off_devices;   /* all devices turned off */
    DeviceIntPtr keyboard;      /* the main one for the server */
    DeviceIntPtr pointer;
    DeviceIntPtr all_devices;
    DeviceIntPtr all_master_devices;
} InputInfo;

extern _X_EXPORT InputInfo inputInfo;        ← Declaration, but not allocated (extern)
```

events.c

```
InputInfo inputInfo;                         ← Defined as global
```

# of references: 286 (grep -R "inputInfo." | wc -l)
 # Reads: 258
 # Writes: 28 (grep -R "inputInfo.* =" | grep -v "for" | grep -v "==" | wc -l)

Fields:

inputInfo.pointer: Core pointer. The first master pointer device and cannot be deleted
inputInfo.keyboard: virtual core keyboard
inputInfo.devices: linked list of all devices
inputInfo.off_devices: Devices not initialized
inputInfo.all_master_devices: extra field to track ids of passive grabs?

List of writes:

```
Xi/extinit.c:        inputInfo.all_devices = &xi_all_devices;
Xi/extinit.c:        inputInfo.all_master_devices = &xi_all_master_devices;
test/xi2/protocol-common.c:    inputInfo.pointer = local_devices.vcp;
test/xi2/protocol-common.c:    inputInfo.keyboard = local_devices.vck;
test/xi2/xi2.c:    inputInfo.all_devices = &all_devices;
test/xi2/xi2.c:    inputInfo.all_master_devices = &all_master_devices;
test/input.c:    inputInfo.all_devices = &xi_all_devices;
test/input.c:    inputInfo.all_master_devices = &xi_all_master_devices;
```

```
test/input.c:    inputInfo.devices = &dev;
test/input.c:    inputInfo.devices = NULL;
test/touch.c:    inputInfo.devices = &dev;
test/touch.c:    inputInfo.devices = &dev;
test/touch.c:    inputInfo.devices = &dev;
dix/devices.c:    inputInfo.off_devices = dev;
dix/devices.c:    inputInfo.devices = NULL;
dix/devices.c:    inputInfo.off_devices = NULL;
dix/devices.c:    inputInfo.keyboard = NULL;
dix/devices.c:    inputInfo.pointer = NULL;
dix/devices.c:                inputInfo.devices = next;
dix/devices.c:                inputInfo.off_devices = next;
dix/events.c:            inputInfo.keyboard->spriteInfo->sprite = pDev->spriteInfo->sprite;
dix/events.c:    inputInfo.numDevices = 0;
dix/events.c:    inputInfo.devices = (DeviceIntPtr) NULL;
dix/events.c:    inputInfo.off_devices = (DeviceIntPtr) NULL;
dix/events.c:    inputInfo.keyboard = (DeviceIntPtr) NULL;
dix/events.c:    inputInfo.pointer = (DeviceIntPtr) NULL;
hw/xquartz/quartz.c:    inputInfo.pointer->spriteInfo->sprite->physLimits = bounds;
hw/xquartz/quartz.c:    inputInfo.pointer->spriteInfo->sprite->hotLimits = bounds;
```

Exploitability?

Extern declaration on a struct does not allocate memory, but declares the struct. Once defined in events.c, will this structure become global. This data structure appears to be core in defining input devices amongst others. Writes appear benign and are initialized. Not sure about Quartz or what Quartz is designed to do. Data structure is globally defined, so an untrusted component could easily control this data. Modifying input devices could enable an adversary to swap an input device to a keyboard of their choosing? Or bake a keylogger into XServer?
