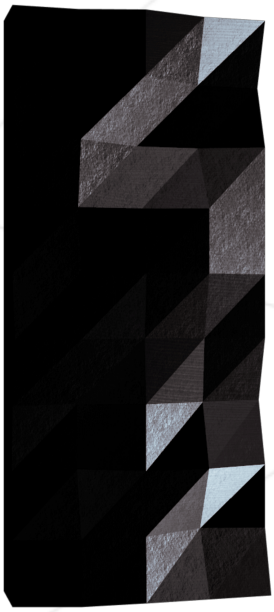

Driver and Keyboard

LK2

Louis Solofrizzo louis@ne02ptzero.me
42 Staff pedago@42.fr

Summary: *Keylogger h3ker*

Contents

Chapter I

Forewords

Chapter II

Introduction

II.1 Drivers

Driver ? What is a 'driver' ? Let's ask Wikipedia:

> *A driver is a kind of sail used on some sailboats. Smaller than a fore and after spanker on a square rigger, a driver is tied to the same spars.*

Wait a minute. Wrong driver !

> *In computing, a device driver (commonly referred to as a driver) is a computer program that operates or controls a particular type of device that is attached to a computer. A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details of the hardware being used.*
> *A driver communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Drivers are hardware dependent and operating-system-specific. They usually provide the interrupt handling required for any necessary asynchronous time-dependent hardware interface.*

So yeah, drivers. Kind of important, since they are the only bridge between the hardware and software.
Remember the Intro of lk_process_and_mem ?

> *Writing a kernel is only a part of writing an operating system, and to do so, one needs a robust and reliable interface between kernelspace and userspace. This interface is the syscalls.*

Lies, lies and lies. Kernel and Syscalls cannot be considered as an OS if there are no drivers. lk_process_and_mem author is a moron.

Chapter III

Goals

- Create a Linux driver
- Understand, register and work with Interrupt Requests
- Understand interrupts keys functions
- Associate a misc-device with a driver

The goal of this project is to create a keyboard driver that can register and log every key event on a keyboard. Basically a Keylogger.
Nothing to do with a r0xx0r thing, something warm, soft, nice and beautiful.
Yeah, like the cat above.

Chapter IV

General instructions

- For this subject, you must use your custom linux distribution, made in the ft_linux subject.
- You must use a kernel version >= 4.0 Stable or not, as long as it's a 4.0 >= version.
- You must create a keyboard driver.
- You must create a misc device associated with this driver.
- *ALL* intern kernel function calls must be verified if its needed. Don't want a Kernel Panic, do you ?
- *ALL* the memory allocated must be properly released. Note the *PROPERLY*
- *ALL* of the requests and registers declared to the Kernel must be properly destroyed when the module exit.
- A Makefile must be turn in.

Chapter V

Mandatory part

- Your driver must log all the key entries
- A key entry should look like this:
  - Key code
  - State of the key (Pressed || Released)
  - Name of the key. Yeah, in english, like 'Return' or 'Delete'.
  - Ascii value of the key.
  - Hour, minute and second when the IRQ is called.
- With those informations in mind, you also need to do a misc device.
- This misc device must be like /dev/module_keyboard. I don't care about the name at all, be creative.
- When open, this misc device should join all the key entries into a buffer with all the information available.
- Each key entry must be in one line like that:

```
HH:MM:SS Name of the key(key code) Pressed / Released
```

- Each entry must be split with a \n
- When read, the misc device should return this information.
- Lock on this file must be properly handled.
- When the driver is destroyed, you must print all of the log, user friendly like. (Be smart, don't print all of your entries)
- This information must be print in the kernel log file.

Chapter VI

Bonus part

- Log the final logging in a /tmp/file instead of the kernel file
- Be creative in that log. You can, for example, print some stats about most pressed keys, time pressed, etc.
- Create a real driver. Like remove the linux original one, and emulate entries into the tty. Must be perfect though :)
- Load this driver when a keyboard is plugged in. Ring a bell ?

Chapter VII

Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.

Your code will be running on your custom linux distribution. Keep in mind the necessary setup to migrate your code from your Mac to your VM.