We will be using Ubuntu 14.04.2 LTS 32-bit

Documentation Resources

Linux Source Code - The source code of the Linux.

Linux Device Drivers, 3rd Edition PDF - Chapter 2 looks to be particularly interesting for the start of the module creation.

The Linux Kernel Module Programming Guide - I might say this is a top useful resource. It might be slightly outdated (2007), but explains well, is detailed, and provides a lot of examples.

Understanding Rootkits

The Basics of Rootkits: Leave No Trace - A high-level overview and description of rootkits.

Linux syscall hooking the IDT - IDT syscall stuff

https://info.fs.tum.de/images/2/21/2011-01-19-kernel-hacking.pdf - good code samples on how to write rootkit and LKM

Linux syscall Reference

http://turbochaos.blogspot.com/2013/09/linux-rootkits-101-1-of-3.html - stealth with lsmod and /sys/module

Getting a kernel module to persist on reboot
↳Using exec helper in linux/kmod.h

Delaying work with Workqueues

Creating a Linux Module in Ubuntu 14.04

~~I'm not sure if we need the Linux source or just the Linux headers~~ It looks like we only need the up-to-date Linux headers.
This is how you get each (you only need the headers though):
~~$ apt-get source linux-source-3.13.0~~
$ apt-get install linux-headers-$(uname -r)

I used a combination of both of these resources for creating, Making, and inserting a module:
How to make the kernel module with a Makefile

Another resource for kernel module: from C to inserting module
NOTE: When you use printk(), it does not print to console, it prints to a log file. To view:
$ tail -f /var/log/syslog
Or you can use this, too:
$ cat /var/log/syslog | less

StackOverflow: How to read and write in a kernel module
↳ Doc for filp_open

Working Makefile (with kernel headers installed)

```
obj-m += my_module.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Useful Module Commands

All of these commands assume that we are dealing with a module called **hello**

$ sudo insmod hello.ko - inserts module “hello”
$ sudo rmmod hello - removes module “hello”
$ sudo lsmod - lists all kernel modules
$ sudo lsmod | grep hello - lists kernel module “hello” if it is inserted

---

Features:

- have rootkit cycle through list of names on boot in the init file
- Defense mechanism: the delete_module() function follows 3 steps. The first one is especially relevant to us:
    - If there are other loaded modules that depend on (i.e., refer to symbols defined in) this module, then the call fails

  So our program should insert 2 or more modules. One “actual/payload” module that does the work, and the rest are “dummy” modules that reference this one to make removal difficult unless you know the names of all of them.

- Potentially hook init_module so that a different module cannot be created to remove this one.