MANAGING THE LINUX KERNEL AND LOADABLE KERNEL MODULES

All OS are made up of at least two major components. The first and most important of these is the kernel. The kernel is at the center if the OS and controls everything the OS does, including managing memory, controlling the CPU, and even controlling what the user sees on the screen.
And the second element is often referred to as user land and includes nearly everything else.
The kernel is designed to protected or privileged area that can only be access by root or other privileged accounts. Access to the kernel allows the user to change how the OS works, looks and feels. It also allows them to crash the OS, making it unworkable.

WHAT IS KERNEL MODULE?

The kernel is the central nervous system of our operating system, controlling everything it does, including managing interactions between hardware components and the starting the necessary services. The kernel operates between the user app and the hardware that runs everything, like the CPU, memory and hard drive.

Linux is a monolithic kernel that enables the addition of kernel modules. As such, modules can be added and removed from the kernel. The kernel will occasionally need updating, which might entail installing new device drivers (such as video cards, Bluetooth devices or USB devices), filesystem drivers, and even system extensions. These drivers must be embedded in the kernel to be fully functional. In some systems, to add a driver, we have to rebuild, compile and reboot the entire kernel, but Linux has the capability of adding some modules to the kernel without going through that entire process. These modules are referred to as LOADABLE KERNEL MODULES, or LKMs.

LKMs have the access to the lowest levels of the kernel by necessity, making them an incredibly vulnerable target. A particular type of malware known as rootkit embeds itself into the kernel of OS, often through these LKMs.

CHECKING THE KERNEL VERSION:

```
Kali> uname -a
Linux Kali 4.6.0kalilamd64 #1 SMP Debian 4.6.4lkalil (2016-07-21) x86_64
```

Distribution: Linux Kali,
The kernel build is 4.6.0 and the architecture it's built for is the x86_64 architecture.
SMP [Symmetric multiprocessing]: it can run on machines with multiple cores or processors.
Debian version: Debian 4.6.4 on July 21, 2016

Other way to get the details:

```
Kali> cat /proc/version
```

KERNEL TUNNING WITH SYSCTL:

We can tune our kernel, meaning we can change memory allocations, enable networking features, and even harden the kernel against outside attacks.

All changes we make with sysctl remain in effect only until we reboot the system. To make any changes permanent, we have to edit the configuration file for sysctl directly at /etc/sysctl.conf.

```
Kali> sysctl -a | less
```

In man-in-the middle (MITM) attack, the hacker places themselves between communicating hosts to intercept info. The traffic passes through the hacker's system, so they can view and possibly alter the communication. One way to achieve this routing is to enable packet forwarding.

```
Kali> sysctl -a | less | grep ipv4
```

The **"*line net.ipv4.ip_forward=0*"** is the kernel parameter that enables the kernel to forward on the packets it receives. In other words, the packets it receives, it sends back out. The default setting is 0, which means that packet forwarding is disabled.

To change it:

```
Kali> sysctl -w net.ipv4.ip_forward=1
```

And to make it permanent, we need to edit conf file /etc/sysctl.conf.

From an operating system-hardening perspective, we could use the file to disable ICMP echo requests by adding the line

```
net.ipv4.icmp_echo_ignore_all=1
```

to make it more difficult but not impossible for hackers to find our system. After adding the line we will need to run the command:

```
Kali> sysctl -p
```

MANAGING KERNEL MODULES:

Linux has at least two ways to manage kernel modules. The older way is to use a group of command built around the ***insmod*** suite- stands for *insert module* and is intended to deal with modules. The second way, using the ***modprobe*** command.

Here we use the ***lsmod*** command from the **insmod** suite to list the installed modules in the kernel:

```
Kali> lsmod
```

The ***lsmod*** command lists all the kernel modules as well as info on their size and what other modules may use of them. So, for instance, the ***nfnetlink*** module-a message-based protocol for communicating between the kernel and user space- is 16384 bytes and used by both the ***nfnetlink***_log module and nf_netlink_queue module.

From the insmod suite,
insmod = to load or insert a module
rmmod = to remove a module

These commands are not perfect and may not take into account module for dependencies, so using them can leave your kernel unstable or unusable. As a result, modern distributions of linux have now

added the ***modprobe*** command, which is automatically loads dependencies and makes loading and removing kernel modules less risky.

FINDING MORE INFORMATION WITH MODINFO:

To learn more about the kernel modules, we can use the ***modinfo*** command.

```
Syntax: modinfo <the name of the module>
Kali> modinfo bluetooth
```

As we can see, the command reveals significant information about this kernel module which is necessary to use Bluetooth on our system. Note that among many others things, it lists the module dependencies: ***rfkill*** and ***crc16***. Dependencies are modules that must be installed for the bluetooth module to function properly.

Typically, this is useful info when troubleshooting why a particular hardware device is not working. Besides nothing thing like the dependencies, we can get info about the version of the module and the version of the kernel the module was developed for and then make sure they match the version we are running.

ADDING AND REMOVING MODULES WITH MODPROBE:

Most newer distributions of Linux, including Kali Linux, include the ***modprobe*** command for LKM management. To add module to our kernel, we would use the command with -a (add) switch,

```
Kali> modprobe -a <module name>
```
Or to remove (-r)
```
Kali> modprobe -r <moduled to be removed>
```

A major advantage of using ***modprobe*** instead of ***insmod*** is that ***modprobe*** understands dependencies, options and installation and removal procedures and it takes all of these into account before making changes. Thus, it is easier and safe to add and remove kernel with ***modprobe***.

INSERTING AND REMOVING KERNEL MODULE:

Let's try inserting and removing a test module to help us familiarize ourself with this process. Let's imagine that we just installed a new video card and we need to install the drivers for it. Remember, the drivers for devices are usually installed directly into kernel to give them the necessary access to function properly. This also makes drivers fertile ground for malicious hackers to install a rootkit or other listening device.

Let's assume for demonstration purpose (don't actually run these commands) that we want to add a new video driver named ***NewVidCard***. We can add it to our kernel by entering the following:

```
Kali> modprobe -a NewVidCard
```

To test whether the new module loaded properly, we can run the command which points out the message buffer from the kernel, and then filter for "***card***" and look for any alerts that would indicate a problem:

```
Kali> dmesg | grep card
```

If there are any kernel message with the word "***card***" in them, they will be displayed here. If nothing appears, there are no messages containing that keyword.

Then to remove this same module,

```
Kali> modprobe -r NewVidCard
```

Remember the loadable kernel are convenience to a Linux user/admin, but they are also a major security weakness and one that professional hackers should be familiar with. The *LKMs* can be the perfect module vehicle to get our rootkit into the kernel and wreak havoc!