Increase kernel integrity with disabled Linux kernel modules loading

By Michael Boelen

Published: 2015-05-12 · Archived: 2026-04-05 23:48:02 UTC

« Back to Kernel

This article has last been updated at March 12, 2025.

Disable loading kernel module on Linux systems

The Linux kernel can be configured to disallow loading new kernel modules. This feature is especially useful for high secure systems, or if you care about securing your system to the fullest. In this article, we will have a look at the configuration of this option. At the same time allowing legitimate kernel modules to be loaded.

Disable kernel modules

Newer kernel modules have a sysctl variable named kernel.modules_disabled.

Sysctl is the tool which allows you to see and change kernel settings of a running system. The related /etc/sysctl.conf file is used to ensure that your settings are also used at the next boot of the system.

The sysctl key **kernel.modules_disabled** is very straightforward. If it contains a “1” it will disable loading new modules, where a “0” will still allow loading them.

Using this option will be a great protection against loading malicious kernel modules. For example, it may help to counter rootkits. Needless to say, but when someone was already been able to gain root access, you have a serious problem. Still, setting this security measure can be useful to achieve maximum hardening of your Linux system. An altered script or program has no chance of loading things you didn’t specifically approve.

Loading modules

To show this functionality, we first will load a module and then see how the related sysctl value works. For this demo purpose, we will use the XOR module, which is part of the crypto category.

```
 # cd /lib/modules/3.13.0-24-generic/kernel/crypto/
# ls -l
total 1012
-rw-r--r-- 1 root root  8516 may 3 2014 ablk_helper.ko
-rw-r--r-- 1 root root 20436 may 3 2014 af_alg.ko
-rw-r--r-- 1 root root 13892 may 3 2014 algif_hash.ko
-rw-r--r-- 1 root root 17748 may 3 2014 algif_skcipher.ko
-rw-r--r-- 1 root root 11772 may 3 2014 ansi_cprng.ko
-rw-r--r-- 1 root root 15756 may 3 2014 anubis.ko
-rw-r--r-- 1 root root  6828 may 3 2014 arc4.ko
-rw-r--r-- 1 root root 17180 may 3 2014 authencesn.ko
..snip..
-rw-r--r-- 1 root root 26076 may 3 2014 twofish_common.ko
-rw-r--r-- 1 root root 10028 may 3 2014 twofish_generic.ko
-rw-r--r-- 1 root root 13540 may 3 2014 vmac.ko
-rw-r--r-- 1 root root 31372 may 3 2014 wp512.ko
-rw-r--r-- 1 root root  9028 may 3 2014 xcbc.ko
-rw-r--r-- 1 root root 21124 may 3 2014 xor.ko
-rw-r--r-- 1 root root 10596 may 3 2014 xts.ko
-rw-r--r-- 1 root root 16396 may 3 2014 zlib.ko
```

Background information

The XOR kernel module uses the “exclusive OR” function, which returns True when an odd number of the given arguments equals to be true, and False when an even are true. Usually it is used with two arguments, so to return True, only 1 of the options can be positive.

Back to the loading of the module. First we ensure this module is not loaded:

```
lsmod | grep xor
```

Now we load the module with the insmod command and then check if loading was successful:

```
 # insmod xor.ko
# lsmod | grep xor
xor 21411 0
```

The module is loaded, as expected. To show normal behavior, we now will remove the kernel module with the rmmod command.

```
 # rmmod xor
# lsmod | grep xor
```

The module is released. Time to disable this functionality, to increase protection against loading malicious modules.

Activating kernel.modules_disabled

By default, the sysctl key is set to “0”, which means new modules can be loaded. This is a safe default for systems but also allows malicious modules to be loaded.

```
 # sysctl -a | grep modules
kernel.modules_disabled = 0
```

Now we disable loading new modules, by using the sysctl key and set it to “1”. There are two ways of doing it, using sysctl directly or echo the value to a file on the pseudo file system /proc, which holds the kernel settings.

```
echo 1 > /proc/sys/kernel/modules_disabled
```

Now we try loading our XOR module again:

```
 # insmod xor.ko
insmod: ERROR: could not insert module xor.ko: Operation not permitted
```

Loading the module is now no longer allowed, exactly what we wanted.

Protection against re-enabling

You might think that loading a kernel module is as simple as re-enabling the option and then still load your kernel module. The kernel has a built-in protection, to avoid this from happening. Trying to set the value back to “0” will result in an “invalid argument” message.

Sysctl showing invalid argument when trying to set value

As can be seen, sysctl will say the value is set to “0”. However, the value isn’t applied, as this key is read-only. Slightly confusing, and therefore always good to check the value again.

```
 # sysctl kernel.modules_disabled
kernel.modules_disabled = 1
```

As expected, the value is still set to “1”.

Disable module loading after boot time

By configuring the /etc/sysctl.conf file we can disallow the loading of kernel modules at boot time. Simply add the related line, with the value “1” as shown in the example.

Caveat: Things might break

Depending on your environment, you might be careful with using this option. It may be working very well on servers, but not on desktop systems. The reason is the type of usage is different, especially when it comes with loading new kernel modules. For example inserting a USB drive, mouse or network functionality might break. So before deploying the option, make sure you test these common use cases.

Hybrid option

Instead of enabling the option directly via /etc/sysctl.conf, it might be better to activate this setting after booting and loading required modules.

Your startup script could be looking like:

```
 #!/bin/sh/
sleep 300
insmod MODULENAME
echo 1 > /proc/sys/kernel/modules_disabled
```

Usually to get iptables working, these are the related modules: **iptables**, **x_tables**, **iptable_filter**.

Depending on your Linux distribution, the startup should be loaded as late as possible. If you have /etc/rc.local available, that is usually a safe bet.

Do you use this option already? Or found some other caveats? Like to hear!
