Lids - Deploying enhanced kernel security in Linux

Thayne Allen

February 12, 2001

What is LIDS?

Lids(Linux Intrusion Detection/Defense System) is an intrusion detection and prevention kernel patch. It is designed to enhance the Linux kernel's security and prevent unwanted use of root priveleges. It does this by disabling system calls in the kernel itself. LIDS can be used to protect important files and processes, logging any attempt to alter them, and can also prevent certain system actions such as installing a packet sniffer or changing firewall rules. LIDS is very customizable, allowing you to choose the Linux capabilities you want to remove. Protection can also be disabled to allow administration of the system. You can obtain LIDS from http://www.lids.org.

Why use LIDS?

More and more security holes are found in the current GNU/Linux system all the time. Most often, a new bug will be exposed to the general public before a patch or fix can be released. This leaves many systems exposed to new threats every day. So how do we protect ourselves from the constant influx of new vulnerabilities? Defense-In-Depth. In order to effectively protect a system, multiple layers of security must be utilized. LIDS simply offers an additional layer of security that a potential intruder must get through in order to effectively compromise a system; host-base protection and detection. LIDS not only protects your system from remote intruders, but also from potentially evil superusers, sudoers, and unauthorized root shells.

Features of LIDS

Features of LIDS include increased file system protection, protection against direct port access or direct memory access, protection against raw disk access, protection of log files, and portscan detection. LIDS also allows you to hide files/directories, hide processes, and prevent certain processes from being killed.

Installing LIDS

Installing LIDS requires some knowledge of the Linux kernel, and how to recompile it. If you don't know how to recompile the Linux kernel, see http://www.redhat.com/support/docs/howto/kernel-upgrade/kernel-upgrade.html for a how-to on the subject.

First, you must download the appropriate LIDS patch for your kernel version. For best results, make sure you have the latest release of the LIDS patch. LIDS is currently targeted toward the 2.2.14 kernels. I installed LIDS on a Redhat 6.2 system that included the 2.2.14 version kernel. You can download the LIDS patch at http://www.lids.org/download.html. Note that there isn't a LIDS release for kernel versions below 2.2.12, so you may need to update your kernel.

After obtaining the patch, copy it to the /usr/src directory and unzip it.

```
    # tar xvfz lids-0.x-2.2.xx.tar.gz
```

The LIDS files will be placed in the lids-0.x directory.

```
    # cd /usr/src/linux
    # patch -p1 < /usr/src/lids-0.x/lids-0.x-2.2.xx.patch
```

Compile the lidsadm program.

```
    # cd /usr/src/lids-0.x/lidsadm-0.x
    # make
    # make install
```

After compiling the lidsadm program, use it to generate a RipeMD-160 password that will be installed in the kernel.

```
    # lidsadm -P
```

Next, configure the kernel.

```
    # make menuconfig
```

You'll want to make sure the following options are selected in the config menu:

```
Code maturity level options ---
[*] Prompt for development and/or incomplete code/drivers

General Setup ---
[*] Sysctl support
```

There will be a selection at the bottom of the kernel configuration menu called Linux Intrusion Detection System. This is where you will configure your LIDS features. Below are the options I chose for my system.. The options that you choose for your system may vary. Also, make sure you copy the long RipeMD-160 password that you previously created into the kernel config.

```
[*] Linux Intrusion Detection System support
--- LIDS Features
[*] Hang up console when raising a security alert
[*] Security alert when execing unprotected programs before sealing
[ ] Do not execute unprotected programs before sealing LIDS
[*] Enable init children lock feature
[*] Try not to flood logs
(30) Authorised time between two identical logs (seconds)
[*] Allow switching LIDS protections
```

```
RipeMD-160 encrypted password: "yourencryptedpasswordhere"
(3) Number of attempts to submit password
(3) Time to wait after a fail
[ ] Allow remote users to switch LIDS protections
[ ] Allow any program to switch LIDS protections
[*] Allow reloading config. file
[ ] Hide some known processes
[*] Port Scanner Detector in kernel
[ ] Send security alerts through network
--- Special Autorizations
[ ] Allow some known processes to access /dev/mem (xfree, etc.)
[ ] Allow some known processes to access raw disk devices
[ ] Allow some known processes to access io ports
[ ] Allow some known processes to change routes
--- Special UPS
[ ] Allow some known processes to unmount devices
[ ] Allow some known processes to kill init children
```

For more detailed descriptions of these options and what they mean, go to the help menu in menuconfig. You can also try http://www.lids.org/lids-howto/node8.html

Next, compile the kernel

```
    # make dep clean
    # make bzImage
```

After compiling the kernel, copy the bzImage to /boot/ and edit /etc/lilo.conf

Running and configuring LIDS

You must configure lids BEFORE you reboot! Use lidsadm to configure LIDS. lidsadm -h will give you a help screen on how to use the lidsadm program. I recommend going over it before configuring LIDS.

You must determine which files/directories you want to protect with LIDS.

To add a file/directory:

```
    # lidsadm -A -a/r/z file/directory/device
```

a = append only
r = read only
z = ignore protection

To delete a file/directory:

```
    # lidsadm -D file/directory/device
```

Use -Z to delete all entries, -U to update dev/inode numbers, and -L to list all entries.

Keep in mind that configuring LIDS properly can be a bit tricky. If you make the wrong files read-only, the system may not boot properly. For instance, if you make the entire /etc directory read only, disks won't mount normally because they can't write to /etc/mtab. Once you protect an entire directory, all files in that directory will be protected as well. You can unprotect certain files within a protected directory by using the -z option. For example:

```
    # lidsadm -A -a /etc
```
to protect the entire /var/log directory(read-only).

```
    # lidsadm -A -z /etc/mtab
```
to allow writing to /var/log/lastlog.

You'll generally want to protect directories that contain your boot files and executables. (/usr/local/sbin, /usr/local/bin, /usr/bin, /usr/sbin, /bin, /boot) You will also want to protect your individual configuration and boot files. e.g. /etc/profile, /etc/syslog.conf, /etc/inetd.conf, etc. Protect your log files as well(generally /var/log), but be sure to make them append-only so the system can write to them. Add any other files/directories you want to keep safe. Remember that you will have to disable LIDS to be able to write to any of your read-only files. To disable lids:

```
    # /sbin/lidsadm -S -- -LIDS
```

You will need to input your password to do this.

There are many capabilities options for lids. I will show some examples of the ones I use here. See the README file in your LIDS directory for more. The help screen (lidsadm -h) also tells you a little bit about the capabilities options for LIDS to help you decide which capabilities you want to leave on or turn off.

The capabilities are:

CAP_CHOWN
CAP_DAC_OVERRIDE
CAP_DAC_READ_SEARCH
CAP_FOWNER
CAP_IPC_OWNER
CAP_SYS_PACCT
CAP_SYS_RESOURCE
CAP_FSETID
CAP_FS_MASK
CAP_KILL
CAP_SETGID
CAP_SYS_MODULE
CAP_SYS_ADMIN
CAP_SYS_TIME
CAP_SETUID
CAP_SETPCAP
CAP_LINUX_IMMUTABLE
CAP_NET_BIND_SERVICE
CAP_SYS_RAWIO
CAP_SYS_BOOT
CAP_SYS_TTY_CONFIG
CAP_NET_BROADC
CAP_NET_ADMIN
CAP_NET_RAW
CAP_IPC_LOCK
CAP_SYS_CHROOT
CAP_SYS_NICE

I'm sure the list will grow with new versions of LIDS.

Here is an explenation of each of the options that I use to help you decide which cababilities you may want to use for your system:

-CAP_SYS_MODULE disables
* kernel modules

-CAP_SYS_RAWIO disables
* ioperm/iopl access
* /dev/port access
* /dev/mem access
* /dev/kmem access
* raw block device (/dev/[sh]d??) access

-CAP_SYS_ADMIN disables
* configuration of the secure attention key
* administration of a random device
* device administration (mknod)
* examination and configuration of disk quotas
* configuring the kernel's syslog (printk behavior)
* setting the domain name
* setting the host name
* calling bdflush(), mount() and umount()
* setting up a new smb connection
* some autofs root ioctls
* nfsservctl
* VM86_REQUEST_IRQ to read/write pci config on alpha
* irix_prctl on mips (setstacksize)
* flushing all cache on m68k (sys_cacheflush)
* removing semaphores
* locking/unlocking of shared memory segments
* turning swap on/off
* passing forged pids on socket credentials
* setting readahead and flushing buffers on block devices
* setting geometry in floppy driver
* turning DMA on/off in xd driver
* administration of md devices (mostly the above, but some extra ioctls)
* tuning the ide driver
* access to the nvram device
* administration of apm_bios
* serial and bttv (TV) device
* manufacturer commands in isdn CAPI support driver
* reading non-standardized portions of pci configuration space
* DDI debug ioctl on sbpcd driver
* setting up serial ports
* sending raw qic-117 commands
* enabling/disabling tagged queuing on SCSI controllers

* sending arbitrary SCSI commands
* setting encryption key on loopback filesystem.

-CAP_NET_ADMIN disables
* interface configuration
* administration of IP firewall
* masquerading and accounting
* setting debug option on sockets
* modification of routing tables
* setting arbitrary process/process group ownership on sockets
* binding to any address for transparent proxying
* setting TOS (type of service)
* setting promiscuous mode
* clearing driver statistics
* multicasting
* reading/writing of device-specific registers.

-CAP_LINUX_IMMUTABLE
* disables modification of S_IMMUTABLE and S_APPEND file attributes.

-CAP_KILL
*enforces the restriction that the real or effective user ID of a process sending a signal must match the real or effective user ID of the process receiving the signal.

-CAP_SYS_RESOURCE enforces
* resource limits
* quota limits
* size restrictions on IPC message queues
* max number of consoles on console allocation
* max number of keymaps.
* This option also disallows more than 64hz interrupts from the real-time clock.

-CAP_SYS_TIME disallows
* manipulation of system clock
* irix_stime on mips
* setting the real-time clock.

-CAP_SYS_TTY_CONFIG disallows
* configuration of tty devices
* vhangup() of tty.

As you can see, there are many options to consider when configuring LIDS. The above options, if used, will help protect agains trivial attacks on your system, among other things.

For my setup, I use the following command to start LIDS:

```
    # lidsadm -I -- -CAP_SYS_MODULE -CAP_SYS_RAWIO -CAP_SYS_ADMIN -
CAP_NET_ADMIN -CAP_SYS_PTRACE \
                   -CAP_KILL -CAP_SYS_RESOURCE -CAP_SYS_TIME -
CAP_SYS_TTY_CONFIG
```

The syntax for the options is a bit confusing. Each of the options listed in CAPS is associated with certain capabilities in Linux. When you give lidsadm a - before an option, you are disabling those capabilities. When you give a + before an option, you are turning those capabilities back on.

Note: If you use the -CAP_SYS_ADMIN option, you will have to turn it off before you shut down in order to umount the disks cleanly, as it restricts umounting;

```
    # lidsadm -S -- +CAP_SYS_ADMIN
```

Once you have decided which options to use, it's time to enter the LIDS command in your startup script. I put mine at the bottom of rc.local:

```
#lidsadm -I -- -CAP_SYS_MODULE -CAP_SYS_RAWIO -CAP_SYS_ADMIN \
        -CAP_NET_ADMIN -CAP_SYS_PTRACE -CAP_KILL -CAP_SYS_RESOURCE \
        -CAP_SYS_TIME -CAP_SYS_TTY_CONFIG
```

Once that's done, reboot the system to make sure everything is set up properly. If something breaks, enter 'security=0' at the LILO prompt to disable LIDS. Check your kernel message logs to find out where the problem is. After you get up and running, monitor your kernel message logs for programs that may not be functioning correctly. You may have to play around with different options for a while to get everything running smoothly. Test the system for a while before doing anything critical, and subscribe to the LIDS mailing list to stay current.

Conclusion

LIDS can be an integral part of the defense-in-depth strategy. It provides an additional layer of security to protect the integrity of your system from intruders, both remote and local. Although the process of implementing LIDS can be a bit complex, it provides an excellent no cost solution for indivuduals or companies looking for additional protection of mission critical systems. LIDS is easily obtained, and there is plenty of documentation on how to install, maintain, and utilize each of LIDS' features.

References

1. "LIDS Project, A Linux semel security snhanced system."
URL: http://www.lids.org (Feb. 2, 2001)

2. Bremer, Steve. "LIDS FAQ" v.01 Dec. 18, 2000
URL: http://www.lids.org/document/LIDS-FAQ.html (Feb. 2, 2001)

3. Huagang, Xie. "Build a Secure System with LIDS" May 16, 2000
URL: http://www.linuxsecurity.com/feature_stories/feature_story-12.html (Feb. 5, 2001)

4. Elson, David. "Focus On Linux: Intrusion Detection on Linux" May 22, 2000
URL: http://www.securityfocus.com/focus/linux/articles/linux-ids.html (Feb. 5, 2001)

5. Slocombe, Emily. "LIDS - surviving bad documentation" July 24, 2000
URL: http://www.ubermachine.com/lids/ (Feb. 6, 2001)

6. Red Hat Support. "Upgrading the Linux Kernel on Red Hat Linux systems" May 17, 2000
URL: http://www.redhat.com/support/docs/howto/kernel-upgrade/kernel-upgrade.html (Feb 13, 2001)