

Overview of LIDS, Part One

Brian Hatch 2001-10-17

An Overview of LIDS, Part One

by *Brian Hatch*

last updated October 17, 2001

What is LIDS

In traditional Unix models, the root user is all-powerful. Root is exempt from the rules and regulations of the filesystem, and has abilities that other users do not: putting interfaces into promiscuous mode, for example. Many folks realized that this uncontrolled access could be a bad thing. Should a vulnerability be found in a program that is run as root, it could cause boundless damage. More importantly, should an attacker manage to gain root access to your machine, there is no limit to what they could do.

The Linux Intrusion Detection System (LIDS) is a Linux kernel patch that will allow users to take away the all-powerful nature of root. They will be able to give programs exactly the access they need, and no more. The root user can be stripped of all his majesty until he is no more powerful than any other user. In the end, it is possible to have a completely functioning system, without worry that some wayward process or malicious cracker can destroy a machine beyond reparability.

This article is the first part of a four-part series that will offer an overview of LIDS. This installment will offer an introduction to LIDS, including how it works, booting LIDS, sealing the kernel, and configuring LIDS.

How LIDS Works

A LIDS kernel has several features. The first is a built-in portscan detector, which can be used to alert users to the warning signs of a possible intruder. The portscan detector is an option that can be chosen at compile time, and is something that can be accomplished with other software, such as scanlogd, if that is preferred. The portscan detector simply logs the offending IP address via syslog.

The most unique ability of LIDS is to enforce strong access restrictions, which I'll simply abbreviate as ACLs. LIDS has two kinds of ACLs, those that restrict actions that can be performed on files such as read/write/append, and those that restrict capabilities a process may possess, such as changing network interface addresses, or changing userids. We'll be covering each of these in their own sections.

LIDS File ACLs are enforced as soon as the kernel boots, whereas the Capability ACLs are not turned on until the kernel is 'sealed' (more on that later.)

LIDS comes in the form of a kernel patch. There are two branches of the patch, one for the 2.2 kernel and one for 2.4. If the user has compiled a kernel before, installation is straightforward, but it's not for the timid. We do not cover the installation of LIDS in this article, instead refer to this SecurityFocus article [Focus On Linux: Intrusion Detection on Linux](#) by David "Del" Elson, or the [LIDS FAQ](#).

Many LIDS features can be enabled at kernel compile time, and if the user did not choose them they would not be available when the kernel is running. Readers may want to read this entire article, as well as the LIDS-FAQ, in order to decide which features are needed before compiling the kernel.

Booting LIDS

Once you have installed LIDS and lidsadm (the LIDS configuration program,) you will have created four new files:

<code>/etc/lids/lids.cap</code>	LIDS Capability bounding set.
<code>/etc/lids/lids.conf</code>	LIDS ACL configuration file.
<code>/etc/lids/lids.pw</code>	LIDS password (used to create a LFS).
<code>/etc/lids/lids.net</code>	LIDS Network notification configuration.

The `/etc/lids` directory is always protected by LIDS, so that it is invisible (DENY) to all users, even root. So if you didn't create `lids.pw` by running the command `'lidsadm -P'` yet, do so now or you will have no way to affect LIDS once we start.

You'll need to do a system reboot to start the new kernel. If at any time you need to boot the LIDS kernel with LIDS turned off (for example, if you are having trouble with the rules you set,) then you need to add `'security=0'` to the lilo/silo prompt:

```
lilo: linux security=0
```

This will boot the kernel with LIDS disabled. If you do not include the `security=0` option, LIDS will be in effect. Hope you remembered that LIDS password...

Sealing The Kernel

A normal Linux kernel can be a fluid thing. Linux Kernel Modules (LKMs) can be installed to add functionality to the kernel at run time, when needed, rather than having the functionality compiled into the kernel itself. Linux firewalls, for example, will often include LKMs that help support proxies for complex protocols. LKMs are wonderful things in that you can wrap and extend existing kernel interfaces to support new functions. However, LKMs can also be used in mischievous ways. For example you could write an LKM that will reply with a fixed system date and hostname when a specific userid requests them (great for fooling licensing schemes), or one that grants you a login shell via the network without any associated daemon.

Rootkits are starting to ship with LKMs that can be used to hide cracker-owned processes and files, or grant them special root access by allowing them to use `setuid()` to become root if they know a magic number. LKMs run in kernel space, so you cannot see what they are doing. Sophisticated LKMs can hide the fact that they've been installed at all.

The integrity of a LIDS kernel would be suspect if inappropriate LKMs were allowed to load. Thus LIDS allows LKMs to be loaded during startup until it is 'sealed' with the `lidsadm -I` command. At the time the kernel is sealed, no more LKMs can be inserted or removed from the running kernel.

`Lidsadm -I` is traditionally placed in a startup script to seal the kernel as soon as all the other startup scripts are complete. For example, if your machine boots into init state 2, you would place something similar to the following in `/etc/rc2.d/S99sealkernel`:

```
#!/bin/sh

case "$1" in
    start) /sbin/lidsadm -I ;;
    stop)  ;;
    *)     echo "Usage: $0 start" >&2; exit 1 ;;
esac
exit 0;
```

Once this is run, no kernel modules can affect the running kernel. Make sure that any modules you depend on, such as those that support your network cards, hard drives, ppp interfaces, etc., are loaded before the kernel is sealed by manually loading them in `/etc/rcX.d` scripts with the `insmod` command.

LIDS Free Session

The most important thing you will need to configure LIDS is a session that is free of the LIDS

constraints. The LIDS configuration files are kept in `/etc/lids`, which is not accessible to any users or programs unless LIDS is not in effect. A LIDS Free Session (LFS) is a session (login) in which you are immune to LIDS restrictions. You establish a LFS by using the `lidsadm` command as follows:

```
root# lidsadm -S -- -LIDS
enter password:
```

You now enter your password and, if correct, you are able to configure the LIDS system and override any restrictions it imposes. For a quick verification, simply try to list the hidden `/etc/lids` directory:

```
lfs# ls /etc/lids
lids.cap  lids.conf  lids.net  lids.pw
```

For this article I will use the prompt `'lfs#'` for any command that must be run from a LIDS Free Session, `'root#'` for a non-LFS root login, and `'user$'` for a generic user login. The `lidsadm` program will not change your prompt for you, of course.

Configuring LIDS With Lidsadm

Once a LIDS-enabled kernel is running, you have only one way to affect LIDS and its configuration, via the program `lidsadm`. The most important uses of the `lidsadm` command are as follows:

<code>lidsadm -Z</code>	Clear delete all LIDS ACLs.
<code>lidsadm -L</code>	List the current ACLs.
<code>lidsadm -V</code>	View the current capability bounding set.
<code>lidsadm -U</code>	Update LIDS' inode table in
<code>/etc/lids/lids.conf.</code>	
<code>lidsadm -I</code>	Seal the kernel.
<code>lidsadm -A ...</code>	Add new LIDS ACLs.
<code>lidsadm -D ...</code>	Delete existing LIDS ACLs.
<code>lidsadm -S -- -LIDS</code>	Start a LIDS Free Session (LFS).
<code>lidsadm -S -- -LIDS_GLOBAL</code>	Turn off LIDS entirely and behave like a
<code>standard Linux kernel.</code>	
<code>lidsadm -S -- +LIDS_GLOBAL</code>	Turn LIDS back on.
<code>lidsadm -S -- +RELOAD_CONF</code>	Reload the LIDS configuration.

Most commands require that you be in a LFS. Those with a `'-S'` additionally require that you supply the LIDS password.

Lids.conf

The LIDS ACLs that you create are stored in the file `/etc/lids/lids.conf`

```
lfs# head /etc/lids/lids.conf
#
#       It is auto generated by lidsadm
#       Please do not modify this file by hand
#
0:0::1:0:23555:770:/sbin
0:0::1:0:1025:770:/bin
0:0::1:0:2:769:/boot
0:0::1:0:57345:770:/lib
0:0::1:0:2:771:/usr
0:0::1:0:11265:770:/etc
```

This file is edited each time you make a modification with the `lidsadm -A` or `lidsadm -D` command. You should not make changes to this file manually!

Most folks prefer to put their ACL-creating `lidsadm` commands in a separate file that they can run, traditionally `/etc/lids/lids.sh`. By putting your commands here you can be sure to have one location to modify whichever ACLs you need, in a format that is still humanly readable - `/etc/lids/lids.conf` is anything but. Since the `/etc/lids` directory is hidden by a default DENY rule, the `lids.sh` program is unavailable to anyone unless they are in a LFS, which is a nice side benefit.

Make sure that you put `'lidsadm -Z'` at the head of `lids.sh` to clear out all the existing ACLs, otherwise you will be adding ACLs to the current set. Whenever you make changes to this file, make sure you reload the LIDS configuration with `'lidsadm -S -- +RELOAD_CONF'`.

Viewing LIDS ACLs

LIDS stores its ACLs in `/etc/lids/lids.conf`. However, due to the cryptic nature of this file, it is not very helpful to view it manually. If you wish to view your current ACLs, use the `lidsadm -L` command:

```
lfs# lidsadm -L
```

Subject	ACCESS(inherit)	Object
Any File	READ:0	/sbin
Any File	READ:0	/bin
Any File	READ:0	/usr
Any File	APPEND:0	/var/log
/bin/su	WRITE:0	/var/log/wtmp

```

/bin/su          READ:0          /etc/shadow
/bin/su          GRANT:0        CAP_SETGID
/bin/su          GRANT:0        CAP_SETUID

```

Both the File and Capability ACLs are shown, in the order in which they are listed in lids.conf.

Reloading LIDS Configuration

The files in /etc/lids are only consulted at bootup time. Additions and deletions to the ACLs via 'lidsadm -A' and 'lidsadm -D' only affect the file /etc/lids/lids.conf file, not the running kernel. In order to tell the kernel to re-read the configuration files, you run the following command:

```
lfs# lidsadm -S -- +RELOAD_CONF
```

This option is only available if you compiled your LIDS kernel with this ability. ('Allow switching LIDS protections' and 'Allow reloading config file' must have been checked at compile time.) If you are administering a machine that you want to be completely paranoid about, then you should not have compiled in this ability, and you will need to reboot in order to have the changes take place.

Unless you reload the LIDS configuration, any changes you make (adding/deleting ACLs, changing the capability bounding set, etc) are not in use.

Lids.net

LIDS includes the ability to send you alerts via the network if you configure it to do so at compilation time by selecting 'Send Security alerts through network'. It uses the file /etc/lids/lids.net to define SMTP settings that it will use to send security alerts via e-mail. The file simply sets the following variables:

```

MAIL_SWITCH=1          # 0 == no alerts, 1 == send alerts
MAIL_RELAY=127.0.0.1:25 # IP: port of the SMTP server
MAIL_SOURCE=machine.example.com # Used for the SMTP HELO
MAIL_FROM=LIDS_ALERT@example.com # The mail From:
MAIL_TO=root@example.com # The mail recipient
MAIL_SUBJECT= LIDS Alert # Subject of the mail

```

It is easy to use the e-mail alert capabilities of LIDS; however, we find that it is usually more useful to use a true log analysis and reporting tool such as swatch, logcheck, logsurfer, or others. These tools can watch the system logs in real time and alert you to only the events you care about. Not all LIDS messages are important immediately, and you cannot configure LIDS to send or ignore

messages based on content, nor configure the addresses to which the mail is sent. For example you may not care when an LFS session is terminated, but you likely want to know when one is started - especially if it wasn't by you. A more configurable log parsing tool can give you this functionality better than LIDS can. And there's no reason to build an SMTP client into the kernel.

What Do ACLs Look Like?

LIDS ACLs come in two flavors - File ACLs, which can regulate access to files, and Capability ACLs which regulate the superuser privileges available to programs.

ACLs restrictions are based on two things, the Object, which is the file or capability to which the ACL applies, and optionally a Subject, which a program that wishes to gain access to the restricted Object. Here are a few quick examples of adding LIDS ACLs with lidsadm:

```
lfs# lidsadm -A -o /var/log/messages -j
APPEND
lfs# lidsadm -A -s /usr/sbin/httpd -o CAP_NET_BIND_SERVICE -j
GRANT
```

The first adds a rule that sets the file /var/log/messages to be available for reading and appending only. The second adds a rule granting the apache process (httpd) the CAP_NET_BIND_SERVICE capability. If you are familiar with ipchains/iptables, you will notice the similarity in the arguments. We will see how these work in the File and Capabilities sections in the subsequent installments in this series.

Next Time?

That concludes the first installment in the three-part series on LIDS. In the next installment of this series, we will look at file restrictions, LIDS File ACLs, and LIDS enhancements of Linux capabilities.

To read **An Overview of LIDS, Part Two** , click [here](#).

Brian Hatch is an obsessive security freak and lead author of [Hacking Linux Exposed](#) and co-author of [Building Linux VPNs](#). While he frequently stays up late to write or hack code, he thinks it's much more fun to go to the park and push his daughter in the swing as he delivers horrible puns to his fiancée.

Relevant Links

[Focus On Linux: Intrusion Detection on Linux](#)

David "Del" Elson, SecurityFocus

[LIDS FAQ](#)

ClubLinux.org

[Privacy Statement](#)

Copyright 2006, SecurityFocus