Linux Security-Related Kernel-Related Stuff: LIDS

1. LIDS Acknowledgements

Brian Hatch, *Overview of LIDS* for `www.securityfocus.com` (particularly Quick Start [Page 5])...

Huagang Xie, Philippe Biondi...

Sander Klein, `man` pages, FAQ...

2. LIDS Overview

2.1. Overview

LIDS is an enhancement for the Linux kernel, currently maintained by Xie Huagang and Philippe Biondi, which implements additional security features including:

- file access control rules [Page 7] on a programme-by-programme basis;
- fine-grained control of *capabilities* [Page 12] of `root`-owned processes, again on a programme-by-programme basis.

`rpms` are available for some systems, but the usual way to install a LIDS-enabled Kernel is to patch a "vanilla" source tree and compile.

For an introduction to *file access control rules* and *capabilities* see the Quick Start [Page 5], below.

3. LIDS Installation

3.1. Patching Kernel Source and Building

LIDS functionality comes from changes to the standard Linux kernel. Binary LIDS-enabled kernels are not available, so installation requires the patching of kernel source code, then building/compiling and installation of the new kernel.

The definitive documentation of building and installing a Linux kernel is given in The Kernel HOWTO[1]. If you are not familiar with the procedure, you are strongly encouraged to read it!

3.2. Required Filesystem Attributes

Question: Do we need the filesystem attributes at install time (when installing the LIDS Tools — for setting the LIDS password) or only when booted into a LIDS kernel?

On a most Unix/Linux filesystems files are uniquely identified by an *inode* which contains metadata for the file, e.g., ownership and access control information. A standard Unix/Linux permission check uses only information present within the inode.

[1] `http://www.tldp.org/HOWTO/Kernel-HOWTO/`

LIDS makes use of *Extended Attributes* [Page 8] — xattrs. Not all kernels are compiled with support for xattrs; for LIDS to function correctly your LIDs-enabled kernel must be so compiled (see below).

3.2.1. Ext2/3

To obtain xattr functionality on Ext2/3 filesystems necessary for correct LIDS operation, ensure your kernel is compiled with

```
CONFIG_EXT2_FS=y
CONFIG_EXT2_FS_XATTR=y
CONFIG_EXT2_FS_POSIX_ACL=y
CONFIG_EXT2_FS_SECURITY=y


CONFIG_EXT3_FS=y
CONFIG_EXT3_FS_XATTR=y
CONFIG_EXT3_FS_POSIX_ACL=y
CONFIG_EXT3_FS_SECURITY=y
```

and then mount filesystems with the `acl` option, i.e., `/etc/fstab`:

```
proc            /proc           proc    defaults,acl        0   0
/dev/hda10      /               ext3    defaults,acl,errors=remount-ro 0       1
/dev/hda9       /boot           ext3    defaults,acl        0   2
/dev/hda14      /scratch        ext3    defaults,acl        0   2
/dev/hda13      /tmp            ext3    defaults,acl        0   2
/dev/hda11      /usr            ext3    defaults,acl        0   2
/dev/hda12      /var            ext3    defaults,acl        0   2
/dev/hda3       none            swap    sw                  0   0
/dev/hdc        /media/cdrom0   udf,iso9660 ro,user,noauto  0   0
/dev/fd0        /media/floppy0  auto    rw,user,noauto      0   0
```

3.2.2. ReiserFS and XFS

Hans Reiser has said that there will never be official support for xattr on ReiserFS v3 — though a patch is available — but ReiserFS v4 *will* support xattr.

XFS supports xattr — to do this efficiently, use an inode size of 512 rather than the standard 256.

3.3. Other Kernel Configuration Requirements

The `Kconfig` (e.g., `/usr/local/src/linux-2.6.14/security/lids/Kconfig`) which comes with the LIDS kernel patch contains

```
depends on EXPERIMENTAL && SYSCTL && SECURITY && SECURITY_SECLVL!=y
                        && SECURITY_ROOTPLUG!=y && SECURITY_SELINUX!=y
                        && SECURITY_CAPABILITIES!=y
```

therefore, in `make config|menuconfig|xconfig`, choose:

```
Code maturity level options
    "Prompt for development and/or..."  = yes
```

```
General setup
    "Sysctl support" = yes


Security options
    "Enable different security models" = yes
    "Default Linux Capabilities"       = no
    "BSD Secure Levels"                = no
    "NSA SELinux Support"              = no


Cryptographic Options
    "SHA256 digest algorithm" = yes
```

i.e., in `.config`:

```
CONFIG_EXPERIMENTAL=y


CONFIG_SYSCTL=y


CONFIG_SECURITY=y
# CONFIG_SECURITY_NETWORK is not set
# CONFIG_SECURITY_CAPABILITIES is not set
# CONFIG_SECURITY_SECLVL is not set
# CONFIG_SECURITY_SELINUX is not set


CONFIG_CRYPTO=y
CONFIG_CRYPTO_SHA256=y
```

3.4. LIDS Kernel Configuration

```
CONFIG_LIDS=y


CONFIG_LIDS_NO_FLOOD_LOG=y
CONFIG_LIDS_ALLOW_SWITCH=y
CONFIG_LIDS_ALLOW_LFS=y
CONFIG_LIDS_RESTRICT_MODE_SWITCH=y
CONFIG_LIDS_MODE_SWITCH_CONSOLE=y
CONFIG_LIDS_MODE_SWITCH_SERIAL=y
CONFIG_LIDS_MODE_SWITCH_PTY=y
```

3.5. Installation Recipe

If you are not familiar with the Linux kernel-building procedure, read the HOWTO[2].

[2] http://www.tldp.org/HOWTO/Kernel-HOWTO/

Most (all?) LIDS-related documentation talks about modular kernels. LIDS-patched kernels may be monolithic.

3.5.1. Patch a Vanilla Kernel Source

Download a "vanilla" kernel source from `www.kernel.org` and patch it with the corresponding LIDS patch:

1. Unpack `linux-x.y.z.tar.gz` into `<DIR>linux-x.y.z` where `DIR` is usually `/usr/local/src/`.
2. Unpack `lids-p.q.r-x.y.z.tar.gz` into `<DIR>lids-p.q.r-x.y.z` — make sure you are using kernel and patch sources which correspond, i.e., that `x`, `y` and `z` match.
3. `cd` into the kernel source directory and patch with `patch -p1 < <DIR>/lids-p.q.r-x.y.z`.

Ensure there are no errors — you should see something like:

```
patching file security/lids/include/linux/lidsif.h
patching file security/lids/include/linux/lidsext.h
patching file security/lids/include/linux/lids.h
patching file security/lids/include/linux/lids_sysctl.h
patching file security/lids/lids_lsm.c
patching file security/lids/lids_acl.c
patching file security/lids/lids_cap.c
patching file security/lids/lids_init.c
patching file security/lids/lids_logs.c
patching file security/lids/lids_sysctl.c
patching file security/lids/Kconfig
patching file security/lids/Makefile
patching file security/lids/Makefile.in
patching file security/Makefile
patching file security/Kconfig
patching file Makefile
```

3.5.2. Configure the Patched Source

Next, configure the kernel:

1. `make config|menuconfig|xconfig`
2. Configure filesystem extended attributes as described above — **unless the above filesystem requirements are met, LIDS will not work properly**.

   Question: Do we need acl-mounted filesystems at LIDS Tools installation time, or only when running a LIDS kernel?

   a. Configure `EXPERIMENTAL`, `SYSCTL` and `SHA256` into the kernel, as described above.
   b. Configure the `SECURITY` options in to the kernel as described above.
   c. Configure the `LIDS` options in to the kernel as described above.

Now build your kernel and if necessary your modules, and install — **do not reboot into this new kernel yet**.

3.5.3. Build the Tools

The final installation step is to build the LIDS Tools. Download and unpack into `<DIR>/lidstools-u.v.w`, then:

1. `cd` into `<DIR>/lidstools-u.v.w`
2. `./configure KERNEL_DIR=C<DIR>/linux-x.y.z`
3. make
4. make install

N.B. The `configure` script currently (as of v2.2.7) installs `lidsadm` and `lidsconf` in `/sbin`, ignoring any `--prefix` options — **Question:** Check this!.

3.5.3.1. LIDS Password

As part of the `make install` you will be asked for a LIDS password. This is used to make changes to your LIDS configuration and also to start LIDS-free sessions, or to switch off LIDS entirely (or switch it back on). **Do not forget this!**

3.5.4. Configure GRUB

The last step before booting your newly-installed LIDS-enabled kernel is to configure your bootloader. It is a good idea to have a couple of choices — here are the Grub entries (for kernels which do not require `initrd`s, e.g., monolithic kernels):

```
title           Vanilla 2.6.14-lids (lids=0)
root            (hd0,8)
kernel          /vmlinuz-2.6.14-lids lids=0 root=/dev/hda10


title           Vanilla 2.6.14-lids
root            (hd0,8)
kernel          /vmlinuz-2.6.14-lids root=/dev/hda10
```

Notice that the first contains the kernel option `lids=0`: this turns LIDS off completely — this can be used if your configuration gets so messed up you cannot do anything, or you forget your LIDS password.

4. LIDS Quick Start

4.1. Booting

4.1.1. Boot into LIDS

After installing [Page 1] your LIDS-enabled kernel and LIDS tools, and adding appropropriate entries to Grub [Page 5], boot into your new kernel (the one with*out* `lids=0`).

4.1.2. `lidsadm -I`

After booting into a LIDS-enabled kernel (not with `lids=0`) LIDS is not yet fully functional: while File ACLs [Page 7] are enforced, Capability ACLs [Page 12] are not yet enforced and the kernel is not yet “sealed” — modules can be loaded and unloaded (if you have a modular kernel).

To fully-enable LIDS, issue the command (as `root`),

```
lidsadm -I
```

In `/var/log/messages` (or similar, depending on your `syslog` configuration) you will see

```
.
.
Apr 18 16:23:36 pinback kernel: LIDS: GLOBAL and POSTBOOT state Config. \
    files loaded
Apr 18 16:23:36 pinback kernel: LIDS: Switching to POSTBOOT state
```

LIDS has now switched from the `BOOT` state to the `POSTBOOT` state — all ACLs are now enforced.

4.2. Logs

`syslog` messages are your friends. When diagnosing issues with a LIDS-enabled kernel its a good idea to have an `xterm` open with

```
tail -f /var/log/syslog|messages|kern.log | grep -i LIDS
```

(exact details depending on your `syslog` configuration).

Messages come in two types: those telling you what LIDS is doing, e.g.,

```
Apr 19 11:45:47 pinback kernel: LIDS: GLOBAL and POSTBOOT state \
                                                    Config. files loaded
Apr 19 11:45:47 pinback kernel: LIDS: Switching to POSTBOOT state
```

and "intrusion detection"-related messages, e.g.,

```
Apr 19 12:14:44 pinback kernel: LIDS: exim4 (dev 3:11 inode 31917) \
    pid 3131 ppid 3013 uid/gid (101/103) on ( tty) : \
    violated CAP_SETUID

Apr 19 12:17:01 pinback kernel: LIDS: cron (dev 3:11 inode 31887) \
    pid 3132 ppid 3045 uid/gid (0/0) on ( tty) : \
    attempt to open shadow for reading
```

4.3. Switching LIDS On/Off Globally

If you have booted into a LIDS-enabled kernel but which to none of the currently-configured ACLS to be enforced — in effect, you wish to "switch off" LIDS across the system, then, issue the command

```
lidsadm -S -- -LIDS_GLOBAL
```

and enter the LIDS password when prompted. To switch enforcement back on across the system,

```
lidsadm -S -- +LIDS_GLOBAL
```

4.4. LIDS-Free Session

There is usually no reason to disable LIDS across the system — a LIDS-free session only is needed most of the time, e.g., to add or remove ACLs. To start a LIDS-free session issue the command

```
lidsadm -S -- -LIDS
```

and enter the LIDS password when prompted. Commands entered in this login/session are immune from LIDS ACLs.

LIDS-free sessions are useful patching and installing new software.

Question: Is it possible to have two or more LIDS-free sessions running on a system, or is there a limit of one?

4.5. Changing the Password

As part of the installation process a password was configured which is required for LIDS-related administration. To change this start a LIDS-free session, then issue

```
lidsconf -P
```

and enter the new password when prompted, then type

```
lidsadm -S -- +RELOAD_CONF
```

to load the new configuration.

4.6. File ACLs

4.6.1. Tradition Unix File ACLs

Traditional Unix supports only owner/group/world read/write/execute permissions, e.g.,

```
prompt> ls -l a.out
-rwxr-xr-- 1 mc users 6008 Apr 18 09:48 a.out
prompt>
```

4.6.2. Filesystem ACLs

Some filesystems add a layer of permissions. For example Ext2/3:

```
prompt> lsattr /var/log/messages
--------
prompt> chattr +a /var/log/messages
prompt> lsattr /var/log/messages
-----a--
prompt>
```

Above we have added the “append only” attribute to a file (so that potential intruders cannot edit out evidence of their existence). For more see `man 1 lsattr` and `man 1 chattr`.

4.6.3. Linux Virtual Filesystem (VFS)

To quote from *The Linux Virtual Filesystem Layer*[3], by Neil Brown:

The Linux operating system supports multiple different file-systems, including Ext2 ..., NFS..., FAT... and others. To enable the upper levels of the kernel to deal equally with all of these and other filesystems, Linux defines an abstract layer, known as the Virtual Filesystem, or VFS. Each lower level filesystem must present an interface which conforms to this Virtual Filesystem.

(For an introduction to the VFS API, read *A Tour of the Linux VFS*[4], by Michael K. Johnson.)

4.6.4. Extended Attributes (xattr)

These are filesystem attributes — name/value pairs associated with files — additional to normal inode-based attributes. xattrs are designed as a filesystem-independent way of adding functionality such as POSIX ACLs. POSIX ACLs can, for example, control file access permissions in a much more fine-grained way than traditional Unix permissions, for example:

```
prompt> getfacl lgtoclnt-7.1.2-1.i686.rpm
# file: lgtoclnt-7.1.2-1.i686.rpm
# owner: root
# group: root
user::rw-
group::r--
mask::r--
other::r--


prompt> setfacl -m u:zzcgupa:r lgtoclnt-7.1.2-1.i686.rpm


prompt> getfacl lgtoclnt-7.1.2-1.i686.rpm
# file: lgtoclnt-7.1.2-1.i686.rpm
# owner: root
# group: root
user::rw-
user:zzcgupa:r--
group::r--
mask::r--
other::r--
prompt>
```

Note that the existence of such ACLs are hinted at in the traditional Unix `ls` output by a `+`:

```
prompt> ls -l lgtoclnt-7.1.2-1.i686.rpm
-rw-r--r--+  1 root  root  14473581 Sep 30  2005 lgtoclnt-7.1.2-1.i686.rpm
```

For more on xattr and POSIX ACLs see `man 5 acl`, `man 3 getfacl` and `man 3 setfacl`.

Question: Does the VFS API require xattr support?

[3] `http://www.cse.unsw.edu.au/~neilb/oss/linux-commentary/vfs.html`
[4] `http://www.tldp.org/LDP/khg/HyperNews/get/fs/vfstour.html`

4.6.5. LIDS File ACLs

LIDS uses its own brand of file ACLs in the kernel — they are integrated into the VFS and so do not depend on the filesystem type (though LIDS requires xattrs and not all filesystems support these).

LIDS File ACLs are enforced as soon as the system boots, though can be turned off in a LIDS-free session (or if LIDS is switched off globally).

There are four types of LIDS File ACLs:

- `DENY` — the existence of the object is denied by the kernel;
- `READONLY` — the object is accessible read-only;
- `APPEND` — the object is read-accessible and can be appended to, but cannot be written to in any other way;
- `WRITE` — the object is read/write-accessible (i.e., not protected by LIDS at all).

4.7. Viewing File ACLs

To view the current set of LIDS File ACLs issue `lidsconf -L`, e.g.,

```
prompt> lidsconf -L
            Subject   ACCESS  inherit               Object
-------------------------------------------------------------------------
           Any file  READONLY:  0                 /sbin
           Any file  READONLY:  0                  /bin
           Any file  READONLY:  0                 /boot
           Any file  READONLY:  0                  /lib
           Any file  READONLY:  0                  /usr
           Any file  READONLY:  0                  /etc
           Any file      DENY:  0             /etc/lids
           Any file      DENY:  0           /etc/shadow
           Any file    APPEND:  0              /var/log
           Any file     WRITE:  0         /var/log/wtmp
         /bin/login  READONLY:  0           /etc/shadow
            /bin/su  READONLY:  0           /etc/shadow
         /bin/login     GRANT:  0            CAP_SETUID
            /bin/su     GRANT:  0            CAP_SETUID
```

N.B. This command **lists the rules listed in** `/etc/lids/lids.conf`**, not those currently enforced** — to be enforced such rules need to have been compiled and loaded by the kernel (see *Adding File ACLs* [Page 9], below).

4.8. Adding File ACLs

The simplest way to add a rule is to specify an object and the associated permissions, e.g.,

```
lidsconf -A -o /etc/shadow -j DENY
```

The ACL above states that nothing can access `/etc/shadow` in any way. A subject can also be specified, e.g.,

```
lidsconf -A -s /bin/login -o /etc/shadow -j READONLY
```

The above ACL states that /bin/login has read-only access to /etc/shadow.

We now have two rules which apply to /etc/shadow. *When faced with two rules which could apply to access to a file, LIDS picks the most specific*, thus the second rule is an exception to the first as you might expect.

The above rules are not yet not yet enforced; the rules must be compiled

```
lidsconf -C
```

and then the newly-compiled configuration loaded lidsadm -S – +RELOAD_CONF </PRE> into the kernel.

4.9. Protect Your Subjects

LIDS has a certain amount of intelligence built in — it will not let you create exceptions (see above) if the subject itself is not protected. For example, assuming the rule-set listed above,

```
prompt> lidsconf -A -s /home/simonh/bin/shell -o /var/log/wtmp -j WRITE
lidsconf: You must protect the subject file /home/simonh/bin/shell or \
                                        its directory as READONLY or DENY.
```

but

```
prompt> lidsconf -A -o /home/simonh/bin/shell -j READ
prompt> lidsconf -A -s /home/simonh/bin/shell -o /var/log/wtmp -j WRITE
prompt>
```

4.10. Inode-Based — Update Your File ACL Configuration After Changes

. . . **for example, after patching the system and after adding new users**, in fact after any changes to files with associated LIDS File ACLs.

LIDs maintains its File ACLs using the filsystem, referenced by major and minor device numbers, and the file's inode number, rather than the path and filename. (See the contents of /etc/lids/lids.conf [Page 38], for example.) The inode of a file will likely change whenever the file is changed, e.g., when adding a user to /etc/passwd and /etc/shadow, or when patching the system.

To update the inodes held by LIDS:

```
lidsadm -U
lidsadm -S -- +RELOAD_CONF
```

4.11. LIDS States and ACL Types: BOOT, GLOBAL, POSTBOOT and SHUTDOWN

LIDS operates in one of three states: BOOT, POSTBOOT and SHUTDOWN — each state has its own set of ACLs, i.e., one can configure one set of rules appropriate for the boot process, a second for normal operation, POSTBOOT, and a third set for shutting the system down. When adding (or deleting) an ACL, either a particular state is specified, or the ACL is assumed to be GLOBAL, i.e., persist across all states.

4.11.1. Switching from BOOT to POSTBOOT

After booting — after all startup scripts in `/etc/init.d/rc[2|3|5].d` have run — a LIDS-enabled kernel is in the `BOOT` state. The usual process is to "seal" the kernel and switch to the `POSTBOOT` state by issuing the command

```
lidsadm -I
```

though one can simply switch states via

```
lidsadm -S -- +POSTBOOT
```

and entering the LIDS password when prompted. It is a good idea to create a suitable startup script [Page 38] and sym-link it appropriately in order to issue `lidsadm -I` automatically at the end of the boot sequence.

4.11.2. Switching from POSTBOOT to SHUTDOWN

Before shutting down the machine, assuming appropriate `SHUTDOWN` ACLs have been configured, switch state:

```
lidsadm -S -- +SHUTDOWN
```

It is a good idea to have an `init` script [Page 38] run this command.

4.12. Listing ACLs

For example

```
prompt> lidsconf -L

            Subject   ACCESS  inherit               Object
-------------------------------------------------------------------------
           Any file  READONLY:  0                 /sbin
           Any file  READONLY:  0                  /bin
           Any file  READONLY:  0                 /boot
           Any file  READONLY:  0                  /lib
           Any file  READONLY:  0                  /usr
           .
           .
```

the *global* ACLs prevent any programme from writing to `/lib`, but

```
prompt> lidsconf -L BOOT

             Subject   ACCESS  inherit               Object
-------------------------------------------------------------------------
           Any file  READONLY:  0          /sbin/depmod
           Any file  READONLY:  0                  /lib
      /sbin/depmod     WRITE:  0                  /lib
```

during the boot process /sbin/depmod is allowed WRITE access to /lib. This is removed in the POSTBOOT state:

```
prompt> lidsconf -L POSTBOOT

            Subject   ACCESS  inherit               Object
-------------------------------------------------------------------------
```

4.13. Adding ACLs

When adding an ACL it is by default global (persists across all states)

```
lidsconf -A -s <subject> -o <object> -j <action>
```

but an ACL type can be specified, e.g.,

```
lidsconf -A BOOT -s <subject> -o <object> -j <action>
```

In most (all?) examples in this section, *Quick Start*, we do not specify acl_type so our ACLs are GLOBAL, i.e., apply to all states.

4.14. Unix Capabilities

Traditionally, the Unix root user is all-powerful; *Capabilities* are a more fine-grained approach. (Unix Capabilities are described in the POSIX 1003.1e draft, which unfortunately has not become a standard.)

In the Capability model, permissions are categorised — some examples are:

CAP_KILL	Allow signals to be sent to processes you don't own.
CAP_LINUX_IMMUTABLE	Allow modification of immutable and append file attributes.
CAP_SETUID	Allow unrestricted setuid.
CAP_SYS_CHROOT	Allow use of chroot.

In a default Linux kernel root-owned processes are granted all Capabilities, while other processes are granted none, having access based on such things as uid restrictions and/or file permissions.
Question: In fact in a LIDS-enabled kernel only root-owned processes are granted Capabilities, yes?

For a more complete list and details of each capability, see *LIDS Capability ACLs* [Page 26], below. For a definitive list of Capabilities, see /usr/include/linux/capability.h.

4.15. LIDS Capability ACLs

LIDS extends the Capability model in two ways. First, you can remove a Capability from the *Bounding Set* (i.e., even `root`-owned processes cannot use that Capability) then later replace that Capability — with a standard kernel such replacement is not possible. Secondly, Capabilities can be granted on a programme-by-programme basis, allowing for fine-grained capability access-control.

LIDS Capability ACLs are not enforced after boot, but after the command `lidsadm -I` is issued, i.e., after changing from `BOOT` state to `POSTBOOT` state.

4.16. Global Capability ACLs

The *Capability Bounding Set* — in LIDS the default Capabilities — for a LIDS-enabled kernel is configured by manually editing `/etc/lids/lids.cap` (or `/etc/lids/lids.boot.cap`, `/etc/lids/lids.postboot.cap`, or `/etc/lids/lids.shutdown.cap`, settings in which override those in `lids.cap`, in the corresponding state).

Question: Is the previous sentence (e.g., settings in `lids.boot.cap` override those in `lids.cap` in the `BOOT` state) correct?

A fragment of (one of) the file(s) could look like this:

```
### 5: Overrides the restriction that the real or effective user ID of a
### 5: process sending a signal must match the real or effective user
### 5: ID of the process receiving the signal.
#
+5:CAP_KILL


### 6: - Allows setgid(2) manipulation
### 6: - Allows setgroups(2)
### 6: - Allows forged gids on socket credentials passing.
#
+6:CAP_SETGID


### 7: - Allows set*uid(2) manipulation (including fsuid).
### 7: - Allows forged pids on socket credentials passing.
#
-7:CAP_SETUID


### 8: Transfer any capability in your permitted set to any pid, remove
### 8: any capability in your permitted set from any pid.
#
-8:CAP_SETPCAP


### 9: Allow modification of S_IMMUTABLE and S_APPEND file attributes.
#
+9:CAP_LINUX_IMMUTABLE


### 10: Allows binding to TCP/UDP sockets below 1024.
#
+10:CAP_NET_BIND_SERVICE
```

The general format is

```
# anything after a "#" is a comment
[+-]:CAP_NAME
```

- `+CAP_NAME` indicates that the capability is left in the bounding set and processes running as `root` have that capability.
- `-CAP_NAME` indicates that the capability is not available to any processes (unless a Capability Exception [Page 14] has been granted).

4.17. Viewing Capability ACLs

To see the current Capability Bounding Set issue

```
lidsadm -V
```

To see the Capabilities granted on a programme-by-programme basis, issue

```
lidsconf -L
```

in a LIDS-free session.

4.18. Adding LIDS Capability ACLs — Granting Capability Exceptions

Adding (granting) a LIDS Capability to a programme is similar to adding a File ACL, for example

```
lidsadm -S -s /bin/login -o CAP_SETUID -j GRANT
```

The above ACL states that `/bin/login` is granted the `CAP_SETUID` Capability.

Above we have allowed `/bin/login` access to the `CAP_SETUID` on a per-programme basis. This overrides the Capability Bounding Set and is therefore called a Capability Exception.

(Here we are assuming that `-7:CAP_SETUID` appears in the prevailing Capability configuration file — after booting and issuing `lidsadm -I` this would be `/etc/lids/lids.postboot.cap`.)

5. Tweaking ACLS — An Example: The OpenSSH Server

After booting a freshly-installed LIDS-enabled system, issuing `lidsadm -I` and attemping to access it remotely via SSH, with the OpenSSH daemon running, access is denied with `Read from socket failed: Connection reset by peer`; the following appears on the console (or `/var/log/syslog` and/or `/var/log/kern.log`, depending on your `syslog` configuration):

```
Apr 10 17:21:21 pinback kernel: LIDS: sshd (dev 3:11 inode 32098) \
    pid 3536 ppid 3535 uid/gid (0/65534) on ( tty) : \
    violated CAP_SETUID
```

So we allow this capability:

```
lidsadm -S -- -LIDS
lidsconf -A -s /usr/sbin/sshd -o CAP_SETUID -j GRANT
lidsconf -C
lidsadm -S -- +RELOAD_CONF
lidsadm -S -- +LIDS
```

This time access is denied with `Permission denied, please try again`, correct credentials, or not, and the following appears on the console (or in the `syslogs`):

```
Apr 11 13:01:01 pinback kernel: LIDS: sshd (dev 3:11 inode 32098) \
    pid 3275 ppid 3217 uid/gid (0/0) on ( tty) : \
    attempt to open shadow for reading
```

Following the same procedure as above but replacing the capability-granting line with

```
lidsconf -A -s /usr/sbin/sshd -o /etc/shadow -j READONLY
```

solves our problem — remote access via OpenSSH is not possible.

For interest, these lines are added to `/etc/lids/lids.conf`:

```
32098:779:/usr/sbin/sshd:16:0:-1:7:CAP_SETUID:0-0
32098:779:/usr/sbin/sshd:1:0:33191:778:/etc/shadow:0-0
```

6. LIDS ACL Inheritance

Sometimes it is useful for a programme to pass its permissions along to programmes it calls — this is common with scripts which call system binaries.

Consider syslog rotation, often called each night as a `cron` job. In general, it should be possibly to append to syslog logs, only:

```
logconf -A -o /var/log -j APPEND
```

but `/etc/cron.daily/logrotate` requires `WRITE` access

```
lidsconf -A -s /etc/cron.daily/logrotate -o /var/log -j WRITE
```

After adding this ACL, compiling and reloading the configuration, a call to this script fails:

```
error: failed to rename /var/log/exim4/mainlog to /var/log/exim4/mainlog.1: Operation not permitte
error: error creating /var/log/exim4/mainlog: Operation not permitted
.
.
```

A look at the script reveals why:

```
#!/bin/sh

test -x /usr/sbin/logrotate || exit 0
/usr/sbin/logrotate /etc/logrotate.conf
```

The script is simply a wrapper for /usr/sbin/logrotate. We want the latter to *inherit* WRITE permission from the script. Therefor we use this ACL instead

```
lidsconf -A -s /etc/cron.daily/logrotate -o /var/log -i 1 -j WRITE
```

The -i 1 option means that /etc/cron.daily/logrotate's children inherit its ACLs, but not its grandchildren — use -i 2 for that; use -i <n> for n levels of inheritance. For unlimited inheritance specify -i -1.

7. LIDS Things Not To Forget

Update your inode references

Remember to update the inodes [Page 10] held by LIDS after patching or adding users to the system, or making other changes.

Switch to POSTBOOT and "seal" the kernel

. . . after the boot sequence finishes:

```
lidsadm -I
```

8. LIDS Command-Line Tools — Examples and Usage

The user-space tools for configuring and administering a LIDS-enabled kernel are lidsconf and lidsadm.

8.1. lidsconf

All the information contained in this section is available from the man page for lidsconf (man 8 lidsconf) or from command-line help (standard: lidsconf -h, or more information: lidsconf -H).

8.1.1. Examples

`lidsconf -A BOOT -o /var/log/message -j APPEND`

Protects `/var/log/message` as append only in BOOT state.

`lidsconf -A POSTBOOT -o /sbin/test -j IGNORE`

Specifies that the read-only protection of `/sbin` doesn't apply to `/sbin/test` in POSTBOOT state.

`lidsconf -A POSTBOOT -o /etc/shadow -j DENY`

Make `/etc/shadow` hidden from everyone only in BOOT state. Nothing can see the file (`open`, `stat`,..).

`lidsconf -A POSTBOOT -s /bin/login -o /etc/shadow -j READ`

Allows the `/bin/login` program to read the `/etc/passwd` even though it has been defined as hidden above. In this case, only `/bin/login` can read `/etc/passwd`. No other program or user can see the file (`/etc/passwd`).

`lidsconf -A -s /usr/sbin/httpd -o /home/httpd -j READ`

Protects the server root of a web server (`/home/httpd`) as DENY...

`lidsconf -A -s /usr/sbin/httpd -o CAP_NET_BIND_SERVICE 80 -i -1 -j GRANT`

...and allow only the httpd binary (`/usr/sbin/httpd`) to read the server root (`/home/httpd`), and the `httpd` can only bind to port 80.

`lidsconf -A SHUTDOWN -s /bin/program -i 2 -o CAP_NET_ADMIN -j GRANT`

Grant the `/bin/program` the capability of `CAP_NET_ADMIN`, and the inheritance level is 2 only in SHUTDOWN state.

`lidsconf -A -s /usr/X11/bin/XF86_SVGA -o CAP_SYS_RAWIO -j GRANT`

Grants the program `XF86_SVGA` the capability of `CAP_SYS_RAWIO` if the `CAP_SYS_RAWIO` has been disabled in `/etc/lids/lids.cap`.

8.1.2. Usage

```
lidsconf -A [acl_type] [-s subject] -o object [-d] [-i level] -j ACTION
lidsconf -C
lidsconf -D [acl_type] [-s file] [-o file]
lidsconf -Z [acl_type]
lidsconf -U
lidsconf -L [acl_type] [-e]
lidsconf -P
lidsconf -S [acl_type]
lidsconf -v
lidsconf -[h|H]
```

where

```
-A, --add To add an entry
-C, --check       To check all entries
-D, --delete      To delete an entry
-Z, --zero        To delete all entries
-U, --update      To update dev/inode numbers
-L, --list        To list all entries
-P, --passwd      To set a new password
-S, --script      To write a script for all entries
-v, --version     To show the version
-h, --help        To list this help
-H, --morehelp    To list this help with CAP/SOCKET name
```

and

```
-s, --subject subj
          can be any program, must be a file

-o, --object [obj]
          can be a file, directory or Capability, Socket Name
```

and `ACTION` can be

```
-j, --jump

   DENY      deny access
   READONLY  read only
   APPEND    append only
   WRITE     writable
   GRANT     grant capability to subject
   IGNORE    ignore any permissions set on this object
   DISABLE   disable some extension feature
```

Finally:

```
  -i, --inheritance  Inheritance level
  -e, --extended     Extended list
```

8.2. lidsadm

All the information contained in this section is available from the `man` page for `lidsadm` (`man 8 lidsadm`) or from command-line help (`lidsconf -h`).

8.2.1. Examples

`lidsadm -I`

Seal the kernel with the default capabilities set in `/etc/lids/lids.cap`. You should edit that file manually.

`lidsadm -S -- -LIDS`

Switch to a LIDS-free session.

`lidsadm -S -- -LIDS_GLOBAL`

Switch LIDS off across the system — your system is no longer protected by LIDS.

`lidsadm -S -- +SHUTDOWN`

Switch to hte SHUTDOWN state.

`lidsadm -S -- +ACL_DISCOVERY`

Turn on the ACL discovery mode.

8.2.2. Usage

```
lidsadm -[S|I] -- [+|-][LIDS_FLAG] [...]
lidsadm -V
lidsadm -h
```

where

```
-S  To submit a password to switch some protections
-I  To switch some protections without submitting password (sealing time)
-V  To view current LIDS state (caps/flags)
-v  To show the version
-h  To list this help
```

and the available LIDS flags are

```
LIDS           de-/activate LIDS locally (the shell & childs)
LIDS_GLOBAL    de-/activate LIDS entirely
RELOAD_CONF    reload config. file and inode/dev of protected programs
POSTBOOT       de-/activate LIDS learning mode
SHUTDOWN       de-/activate LIDS learning mode
ACL_DISCOVERY  de-/activate LIDS learning mode
```

9. LIDS ACL Initialisation Script

It is not practical to enter a complete set of ACLs at the shell prompt; a better approach is to store ACLs in a shell-script. After initialising ACLs from such a script, the rules “compiled” and stored in `/etc/lids/*.acl` for future system boots.

(*The script below is not* `/sbin/init` *script* — see above for one of those [Page 38].)

```
#!/bin/sh


# ------------------------------------------------------------------------
# -- Check where we're running :

echo "  "
echo "    Is the kernel LIDS-enabled?"
echo "        Is \"lidsconf -I\" done?"
echo "            Is this a LIDS-free session?"
echo "  "

echo -n "    If yes, yes and yes, enter \"yes\" : "

read user_response

if [ ! "$user_response" = "yes" ]
then
    echo "Answer not equal to \"yes\", so exiting."
    exit 1
fi


# ------------------------------------------------------------------------
# -- Clean out the bath before using it :

echo "  "
echo "     ...should be okay to ignore any "
echo "        \"lidsconf: the file does not exist in the acl file\""
echo "        message here..."
echo "  "

lidsconf -D SHUTDOWN
lidsconf -D POSTBOOT
lidsconf -D BOOT
lidsconf -D
    # ...deletes all current ACLs (if there are currently no ACLs, may get
    #    error "lidsconf: the file does not exist in the acl file" which
    #    can safely be ignored)

echo "     ...end ignore."
echo "  "


# ------------------------------------------------------------------------
# -- ACLs, GLOBAL --- system-wide stuff :

lidsconf -A  -o /bin   -j READONLY
lidsconf -A  -o /boot  -j READONLY
lidsconf -A  -o /etc   -j READONLY
lidsconf -A  -o /lib   -j READONLY
lidsconf -A  -o /sbin  -j READONLY
lidsconf -A  -o /usr   -j READONLY


# ------------------------------------------------------------------------
# -- ACLs, GLOBAL --- /etc :

lidsconf -A  -o /etc/lids    -j DENY
lidsconf -A  -o /etc/shadow  -j DENY

lidsconf -A  -s /bin/login      -o /etc/shadow -j READONLY
lidsconf -A  -s /bin/su         -o /etc/shadow  -j READONLY
lidsconf -A  -s /sbin/sulogin   -o /etc/shadow  -j READONLY
lidsconf -A  -s /usr/sbin/sshd  -o /etc/shadow  -j READONLY
```

10. Miscellaneous LIDS Features/Examples

10.1. Protect Filesystem Devices

LIDS File ACLs protect access to files through the normal channels — using path and filename. However, such channels can be circumvented by reading from or writing to mounted block devices (e.g., `/dev/hda1`) directly. To prevent such access set

```
-17:CAP_SYS_RAWIO
```

in `/etc/lids/lids.cap` and `/etc/lids/lids.*.cap`.

It is rare that Capability Exceptions to this need to be granted; most commonly such exceptions are for X servers or “multimedia</C> software (e.g., `xmms` may access your CD-ROM via `/dev/hd?` rather than `/mnt/cdrom`).

10.2. CAP_BIND_NET_SERVICE

In “vanilla” kernels any programme with the `CAP_BIND_NET_SERVICE` capability can bind to a port number less than 1024. The capability is extended in LIDS-enabled kernels to allow particular ports, or a port range, to be specified. For example

```
lidsadm -A -s /usr/sbin/httpd -o CAP_BIND_NET_SERVICE -j GRANT
```

allows `httpd` to bind to any port, but

```
lidsadm -A -s /usr/sbin/httpd -o CAP_BIND_NET_SERVICE 80-80, 443-443 -j GRANT
lidsadm -A -s /usr/local/sbin/httpproxy -o CAP_BIND_NET_SERVICE 80-88 -j GRANT
```

allow `httpd` to bind to its standard ports only and `httpproxy` to bind to ports in the range 80 to 88 (inclusive).

10.3. Prevent Processes Being Killed

Using the LIDS-specific capability, `CAP_PROTECTED`, a process (daemon) can be protected from userspace signals — such processes cannot therefore be killed. This is useful for protecting monitoring and intrusion-detection tools! Example:

```
root> lidsadm -S -- -LIDS
root> lidsconf -A -s /usr/bin/yes -o CAP_PROTECTED -j GRANT
root> lidsconf -C
root> lidsadm -S -- +RELOAD_CONF
root> lidsadm -S -- +LIDS
```

Now run /usr/bin/yes in another terminal.

```
root> ps auxww | grep yes
    root      3712 13.8  0.0   1876   456 pts/0    R+   12:38   0:02 yes
root> kill 3712
bash: kill: (3712) - Operation not permitted
root> lidsadm -S -- -LIDS
root> kill 3712
root> ps auxww | grep yes
root>
```

N.B. *Remember that only* root *processes have Capabilities* so you cannot protect non-root-owned processes/daemons in this way. For example, exim4 drops is root privilege after binding to port 25 and so cannot be protected. **Question:** Is this paragraph correct?

11. LIDS ACL Discovery

12. LIDS and Patching

Daily (nightly) cron-driven patching does not sit well with LIDS. The only practical approach is to temporarily disable LIDS, patch and then immediately re-enable the system. The script below will does exactly this for a Debian system, *but should NOT be used as is* since it contains the LIDS password in plain text. Usage:

1. replace <password> with the actual LIDS password;
2. encrypt/compile the script using shc, the shell script compiler[5] written by FJR Garcia;
3. run the encrypted/compiled script via cron to patch daily/nightly.

[5] http://www.datsi.fi.upm.es/~frosal/

```
#!/bin/bash

#
# 1. Runs "apt-get update" and "apt-get --download-only upgrade" before
#    issuing "lidsadm -S -- -LIDS_GLOBAL", then "apt-get -u upgrade" and
#    finally "lidsadm -S -- +LIDS_GLOBAL", thus minimising the time
#    for which LIDS is disabled.
#
# 2. Creates temporary expect scripts to temporarily disable and later
#    enable LIDS.
#

# -- update : ------------------------------------------------------------
#
/usr/bin/apt-get update
    # ...writes to /var/cache/apt/pkgcache.bin
    #                            srcpkgcache.bin


# -- download : ----------------------------------------------------------
#
/usr/bin/apt-get --download-only upgrade
    # ...writes to /var/cache/apt/archives/


# -- create expect script to enable LIDS : -------------------------------
#
echo "#!/usr/bin/expect"                >   /tmp/simonh.simonh
echo "  "                               >>  /tmp/simonh.simonh
echo "set timeout 5000  "               >>  /tmp/simonh.simonh
echo "  "                               >>  /tmp/simonh.simonh
echo "spawn lidsadm -S -- -LIDS_GLOBAL" >>  /tmp/simonh.simonh
echo "expect \"password: \""            >>  /tmp/simonh.simonh
echo "send \"<password>\r\""            >>  /tmp/simonh.simonh
echo "expect \"changed.\""              >>  /tmp/simonh.simonh
echo "exit"                             >>  /tmp/simonh.simonh


# -- switch to "-LIDS_GLOBAL" : ------------------------------------------
#
chmod 700 /tmp/simonh.simonh
/tmp/simonh.simonh
rm -f /tmp/simonh.simonh


# -- install : -----------------------------------------------------------
#
/usr/bin/apt-get -y upgrade


# -- create expect script to disable LIDS : ------------------------------
#
echo "#!/usr/bin/expect"                >   /tmp/simonh.simonh
echo "  "                               >>  /tmp/simonh.simonh
echo "set timeout 5000  "               >>  /tmp/simonh.simonh
echo "  "                               >>  /tmp/simonh.simonh
echo "spawn lidsadm -S -- +LIDS_GLOBAL" >>  /tmp/simonh.simonh
echo "expect \"password: \""            >>  /tmp/simonh.simonh
echo "send \"<password>\r\""            >>  /tmp/simonh.simonh
echo "expect \"changed.\""              >>  /tmp/simonh.simonh
echo "exit"                             >>  /tmp/simonh.simonh

# -- switch to "+LIDS_GLOBAL" : ------------------------------------------
#
```

</IMP>

13. LIDS Sockets

Question: cf. `lidsconf -H...`

14. LIDS Example ACLs

The LIDS Wiki[6] contains a section with example ACLs[7].

15. LIDS Sandbox Capabilities — Trusted Domain Enforcement

The LIDS Wiki[8] contains a section describing LIDS Trusted Domain Enforcement in detail[9].

16. LIDS How-Tos, FAQs and Troubleshooting

16.1. Visibility of `/etc/lids`

Sometimes `/etc/lids` is visible — it shouldn't be!

`/etc/lids` is visible in `BOOT` mode; it is not visible in `POSTBOOT`mode, so check you have switched (e.g., `lidsadm -I`). The directory is also visible in a LIDS-free session.

16.2. LIDS Password/Authentication Oddities

16.2.1. `lidsadm -S...` works the second time but not the first!

For example, with `lidsadm -S -- -LIDS`, it fails with "switching lids failed" the first time, but a second time it works fine with "no global capabilities changed". This is because you need to be in `POSTBOOT` mode — this is usually reached via `lidsadm -I`, but `lidsadm -S...`, even a non-authenticated failure, changes mode to `POSTBOOT`.

16.2.2. Sometimes the wrong password is accepted!

Change to `POSTBOOT` mode — use `lidsadm -I`.

16.2.3. It just doesn't work what ever I do!

Okay, you've installed `lidstools` without an apparent hitch, setting the password when prompted, but no matter what you do, `lidsadm -S` won't authenticate — you've even tried un-installing `lidstools` and re-installed, to no avail.

[6] `http://wiki.lids.org`
[7] `http://wiki.lids.org/index.php/LIDS_2.2_ACLs`
[8] `http://wiki.lids.org`
[9] `http://wiki.lids.org/index.php/LIDS_Trusted_Domain_Enforcement`

Check you have the extended attributes set in your kernel

```
CONFIG_EXT3_FS=y              # ...and/or EXT2, depending on /etc/fstab
CONFIG_EXT3_FS_XATTR=y
CONFIG_EXT3_FS_POSIX_ACL=y
CONFIG_EXT3_FS_SECURITY=y
```

and that filesystems are mounted using these — ensure `acl` is included under the mount options in `/etc/fstab` and that `mount` shows this:

```
prompt> mount
/dev/hda10 on / type ext3 (rw,acl,errors=remount-ro)
proc on /proc type proc (rw)
.
.
prompt>
```

17. LIDS Discontinued Features

Portscan Detector

This has been removed.

Time-Dependent ACLs

Before LIDS v2.2 it was possible to for ACLs to be time-dependent, for example

```
lidsconf -A -s /usr/sbin/cron -o /var/log -t 0018-0019 -i 2 -j WRITE
```

would allow `cron` to write to `/var/log` between 00:18 and 00:19. (`-i 2` allows `logrotate` and its children write access; directly granting write access to `logrotate` would be a mistake — this would allow an intruder to repeatedly rotate logs...)

`CAP_HIDDEN`

This was a LIDS-specific Capability. A process with this Capability was not visible in `/proc` (and thus not available to `ps`, etc.). But:

```
> sorry, CAP_HIDDEN will not be working on LIDS 2.2.x..Smile This is because
> LSM do not provide the nesseary hooks that we can use to hide files as
> well as the process(via /proc file system).
>
> I will removed the CAP_HIDDEN in source and lidstools to let is obsoleted.
>
> Thanks,
> huagang
```

It is no longer possible to hide a process using LIDS functionality. Other kernel-patches exist (e.g., GR Security) which prevent users from seeing processes other than their own. To hide particular processes from all users (including root), use a specially-crafted kernel module — a root kit!

`CAP_INIT_KILL`

This was a LIDS-specific Capability. It has been replaced by `CAP_PROTECTED` (another LIDS-specific Capability).

`CAP_KILL_PROTECTED`

This was a LIDS-specific Capability. It allowed programmes to kill `CAP_PROTECTED` processes.

18. Unix/LIDS Capabilities

This list is based on the declaration of `*lids_caps_desc[]`from `lids_cap.c`, from LIDS v2.2.2-2.6.14, and from the `man` page for `capabilities(7)`, rather than the LIDS `man` pages, `lidsconf -H`, or other LIDS documentation, which is sometimes out of date.

`CAP_AUDIT_CONTROL`

Enable and disable kernel auditing; change auditing filter rules; retrieve auditing status and filtering rules.

`CAP_AUDIT_WRITE`

Allow records to be written to kernel auditing log.

`CAP_CHOWN`

Allow arbitrary changes to file UIDs and GIDs (see `chown(2)`, `chgrp(2)`)

`CAP_DAC_OVERRIDE`

Bypass file read, write, and execute permission checks. (DAC = "discretionary access control".)

`CAP_DAC_READ_SEARCH`

Bypass file read permission checks and directory read and execute permission checks.

`CAP_FOWNER`

Bypass permission checks on operations that normally require the file system UID of the process to match the UID of the file (e.g., `chmod(2)`, `utime(2)`), excluding those operations covered by the `CAP_DAC_OVERRIDE` and `CAP_DAC_READ_SEARCH`; set extended file attributes (see `chattr(1)`) on arbitrary files; set Access Control Lists (ACLs) on arbitrary files; ignore directory sticky bit on file deletion; specify `O_NOATIME` for arbitrary files in `open(2)` and `fcntl(2)`.

`CAP_FSETID`

Don't clear set-user-ID and set-group-ID bits when a file is modified; permit setting of the set-group-ID bit for a file whose GID does not match the file system or any of the supplementary GIDs of the calling process.

`CAP_IPC_LOCK`

Permit memory locking (cf. `mlock(2)`, `mlockall(2)`, `mmap(2)` and `shmctl(2)`)

`CAP_IPC_OWNER`

Bypass permission checks for operations on System V IPC objects.

`CAP_KILL`

Bypass permission checks for sending signals (see `kill(2)`). This includes use of the `KDSIGACCEPT` `ioctl`.

`CAP_LEASE`

Allow file leases to be established on arbitrary files (see `fcntl(2)`).

`CAP_LINUX_IMMUTABLE`

Allow setting of the `EXT2_APPEND_FL` and `EXT2_IMMUTABLE_FL` (and `EXT3_` filesystem attributes (see `chattr(1)`).

`CAP_MKNOD`

Allow creation of special files using `mknod(2)` .

`CAP_NET_ADMIN`

Allow various network-related operations (e.g., setting privileged socket options, enabling multicasting, interface configuration, modifying routing tables).

`CAP_NET_BIND_SERVICE`

Allow binding to Internet domain reserved socket ports (port numbers less than 1024).

`CAP_NET_BROADCAST`

Allow socket broadcasting, and listening multicasts.

`CAP_NET_RAW`

Permit use of `RAW` and `PACKET` sockets.

`CAP_PROTECTED` (*specific to LIDS*)

Protect the process from signals.
Question: CAP_KILL vs CAP_PROTECTED?

`CAP_SETGID`

Allow arbitrary manipulations of process GIDs and supplementary GID list; allow forged GID when passing socket credentials via Unix domain sockets

`CAP_SETPCAP`

Grant or remove any capability in the caller's permitted capability set to or from any other process.

`CAP_SETUID`

Allow arbitrary manipulations of process UIDs (`setuid(2)`, `setreuid(2)`, `setresuid(2)`, `setfsuid(2)`); allow forged UID when passing socket credentials via Unix domain sockets.

`CAP_SYS_ADMIN`

Permit a range of system administration operations including: `quotactl(2)`, `mount(2)`, `umount(2)`, `swapon(2)`, `swapoff(2)`, `sethostname(2)`, `setdomainname(2)`, `IPC_SET` and `IPC_RMID` operations on arbitrary System V IPC objects; perform operations on trusted and security Extended Attributes (see `attr(5)`); call `lookup_dcookie(2)`; perform `keyctl(2)` `KEYCTL_CHOWN` and `KEYCTL_SETPERM` operations; allow forged UID when passing socket credentials; exceed `/proc/sys/fs/file-max`, the system-wide limit on the number of open files, in system calls that open files (e.g., `accept(2)`, `execve(2)`, `open(2)`, `pipe(2)`; without this capability these system calls will fail with the error `ENFILE` if this limit is encountered).

`CAP_SYS_BOOT`

Permits calls to `reboot(2)` and `kexec_load(2)`.

`CAP_SYS_CHROOT`

Permits calls to `chroot(2)`

`CAP_SYS_MODULE`

Allow loading and unloading of kernel modules; allow modifications to capability bounding set (see `init_module(2)` and `delete_module(2)`).

`CAP_SYS_NICE`

Allow raising process nice value (`nice(2)`, `setpriority(2)`) and changing of the nice value for arbitrary processes; allow setting of real-time scheduling policies for calling process, and setting scheduling policies and priorities for arbitrary processes (`sched_setscheduler(2)`, `sched_setparam(2)`); set CPU affinity for arbitrary processes (`sched_setaffinity()`); use the `MPOL_MF_MOVE_ALL` with `mbind(2)`.

CAP_SYS_PACCT

Permit calls to acct(2).

CAP_SYS_PTRACE

Allow arbitrary processes to be traced using ptrace(2)

CAP_SYS_RAWIO

Permit I/O port operations (ioperm(2) and iopl(2)

CAP_SYS_RESOURCE

Permit: use of reserved space on Ext2 file systems; ioctl(2) calls controlling Ext3 journaling; disk quota limits to be overridden; resource limits to be increased (see setrlimit(2)); RLIMIT_NPROC resource limit to be overridden; msg_qbytes limit for a message queue to be raised above the limit in /proc/sys/kernel/msgmnb (see msgop(2) and msgctl(2).

CAP_SYS_TIME

Allow modification of system clock (settimeofday(2), adjtimex(2)); allow modification of real-time (hardware) clock.

CAP_SYS_TTY_CONFIG

Permit calls to vhangup(2).

Question: lids_acl.c refers to CAP_KILL_PROTECTED but this is certainly not available as a Capability (try it with lidsconf). Is this dead code?

19. LIDS Man Pages: lidsconf

This man *page is an updated/corrected version of that which comes with* lidstools *v2.2.7.*

```
NAME
       lidsconf - configuration tool for the Linux Intrusion Detection System


SYNOPSIS

       lidsconf -A [acl_type] [-s subject] [-R] -o object [-d] [-i level] -j ACTION
       lidsconf -C
       lidsconf -D [acl_type] [-s file] [-o file]
       lidsconf -Z [acl_type]
       lidsconf -U
       lidsconf -L [acl_type] [-e]
       lidsconf -P
       lidsconf -S [acl_type]
       lidsconf -v
       lidsconf [-h|H]
```

```
DESCRIPTION

       lidsconf is a configuration tool for  the  Linux  Intrusion  Detection  System
       (LIDS).


       LIDS is a kernel patch to enhance the current Linux kernel. With LIDS, you can
       protect important files, directories, and devices. You can  also  define  ACLs
       that  restrict  the  access control on the entire system. For more information
       about LIDS, please go to http://www.lids.org.


       lidsconf is used to configure the access restriction information for LIDS. All
       of           the           information          is          stored          in
       "/etc/lids/lids.conf","/etc/lids/lids.boot.conf","/etc/lids/lids.post-
       boot.conf","/etc/lids/lids.shutdown.conf" based on the ACL type.


OPTIONS (ACL's)

       ACL  is short for "Access Control List". The ACL in LIDS defines how a subject
       can access an object. The subject can be any program file on the  system.  The
       object  can  be  a  file, directory, or a special option (MEM devices, RAW IO,
       etc). The target defines the access type that the subject  has  on the object.


       The synopsis of the ACL is
              [-s subject] [-i TTL] -o object -j TARGET


       When  a subject is not specified, the ACL defines the object's default access.


       acl_type
              acl_type  can be "BOOT","POSTBOOT", "SHUTDOWN" or blank which refers to
              different acl states, if you do not provide  an  acl_type  the  default
              value is "GLOBAL" which will apply for all states. For more information
              on the LIDS STATEFUL ACL, please check the FAQ under the doc directory.


       -s subject
              A subject can be any program on the system, such as "/bin/login".


       -o object [portscale]
              An object can be a file, directory, or a special option (CAP_SYS_RAWIO,
              CAP_INIT_KILL,  etc).  If  the  object  is  CAP_NET_BIND_SERVICE,  you
              must specify the port range. For example, "20-299,400-1002".
```

```
-i <inheritance level>
       This specifies that the ACL is inheritable by the  subject's  children.
       The  inheritance  level  affects  how  far  the  ACL  is inherited. An
       inheritance level of "-1" means unlimited inheritance.  An  inheritance
       level  of  1  means that a child process spawned by the parent which is
       not the same program as the parent will inherit the ACL,  but  a  child
       process  spawned  from the child (i.e. a grandchild of the orignal pro-
       cess) won't.  The Inheritance level will only affect the children which
       are  not  the same program as its parent. If the child is the same pro-
       gram as the parent, it will gain all the permission from its parent.


-j target
       The target can be DENY, READ, APPEND, WRITE, or IGNORE for  nor-
       mal  file access ACLs. For a special object, the target can only
       be GRANT.


COMMANDS

       These options specify the action to perform. Only one  command  can  be
       given on the commandline unless otherwise specified.


-A, --add [acl_type]
       Add one or more rules to the end of the selected acl_type chain.


-C, --check
       Check your LIDS rules and have them compiled. The output of this
       command  can  help  in  making tighter rules or showing problems
       with your current rulebase.


-D, --delete [acl_type]
       Delete one or more rules from the selected acl_type.


-Z, --zero [acl_type]
       Delete all acl's from the selected acl_type. If no  acl_type  is
       given then the rules from the GLOBAL acl_type are deleted.


-U, --update
       Update  your  acl's.  If you change or move a file or directory,
       it's inode will change. You the need to update your lids  config
       with this command


-L, --list [acl_type]
       List the acl's in the selected acl_type.


-P, --passwd
       Set a new LIDS password.
```

```
-S, --script
       Write out a script to set your acl's.


-v, --version
       Show the lidsconf version.


-h, --help
       Show the lidsconf help.


-H, --morehelp
       Show more help options.
```

```
AVAILABLE CAPABILITIES

        The  capabilities  used in LIDS are shown below. You can use the
        name to enable  or  disable  the  capability  when  sealing  and
        switching.  You  can also grant the capability to a program even
        if the capability is disabled globally on the system.
        .
        .
```

For a list of AVAILABLE CAPABILITIES*, see Capabilities [Page 26], above.*

```
EXAMPLES
        .
        .
```

For a list of EXAMPLES*, see Command-Line Tools [Page 17], above.*

```
OTHER SOURCES OF INFORMATION.

       Mailing List
              To  subscribe,  unsubscribe,  go  to:  http://lists.source-
              forge.net/lists/listinfo/lids-user
              To post a  message  to  the  list,  send  an  e-mail  to:  lids-
              user@lists.sourceforge.net
              Current      LIDS      archive      can      be      found      at:
              http://www.geocrawler.com/redir-sf.php3?list=lids-user
              An  outdated  searchable  archive  can  be  found  at:
              http://groups.yahoo.com/group/lids


       LIDS FAQ
              The LIDS FAQ is located at:
              http://www.lids.org/lids-faq/lids-faq.html
              or
              http://www.roedie.nl/lids-faq
```

```
BUGS

       Any  bugs  found  with  LIDS itself should be sent to Xie, Phil, or the
       mailing list (lids-user@lists.sourceforge.net).   Please  include  your
       .config  file  used  to  compile  your  kernel,  and  the lids.conf and
       lids.cap files located in /etc/lids directory.   Any  errors  found  in
       this man page should be sent to Sander Klein.


FILES

       /etc/lids/lids.ini - LIDS Initial file.
       /etc/lids/lids.cap - Defines the global capabilities.
       /etc/lids/lids.boot.cap - Defines the BOOT capabilities.
       /etc/lids/lids.postboot.cap - Defines the POSTBOOT capabilities.
       /etc/lids/lids.shutdown.cap - Defines the SHUTDOWN capabilities.
       /etc/lids/lids.pw - Contains the encrypted LIDS password.


SEE ALSO

       lidsadm(8)


AUTHORS

       Huagang Xie <xie@lids.org>

       Philippe Biondi <biondi@cartel-securite.fr>

       Manpage written by Sander Klein <roedie@roedie.nl>


DISTRIBUTION

       The newest version of LIDS can be obtained from http://www.lids.org/ or
       one  of  it's  mirrors.   LIDS  is  (C)  1999-2004  by  Huagang
       Xie(xie@lids.org).
```

20. LIDS Man Pages: `lidsadm`

This `man` *page is an updated/corrected version of that which comes with* `lidstools` *v2.2.7.*

```
NAME
       lidsadm - administration tool for the Linux Intrusion Detection System


SYNOPSIS
       lidsadm -[S|I] -- [+|-][LIDS_FLAG] [...]
       lidsadm -V
       lidsadm -h
```

```
DESCRIPTION

       lidsadm  is  an  adminstration  tool  for the Linux Intrusion Detection System
       (LIDS).

       LIDS is a kernel patch to enhance the current Linux kernel. With LIDS, you can
       protect  important  files,  directories, and devices. You can also define ACLs
       that restrict the access control on the entire system.  For  more  information
       about LIDS, please go to http://www.lids.org.

       lidsadm is used to define ACLs and administer the LIDS protections online.


COMMANDS

       Commands  define the individual functions of the lidsadm utility.  They cannot
       be combined.


       -S     Change LIDS protections (requires your LIDS password).


       -I     Changes LIDS protections once without a password. This is used to "seal
              the kernel" and to switch from the BOOT to the POSTBOOT acl_type.


       -V     Lets  you view the current state of your LIDS system. (this needs to be
              built in during compile time)


       -v     Shows the version of the lidsadm tool.


       -h     List the help.


LIDS_FLAG's

       There are many flags you can set. They can be used to set or  unset  capabili-
       ties  but  they  can also switch your LIDS system on or off, or to switch into
       different states.


AVAILABLE CAPABILITIES

       The capabilities used in LIDS are shown below. You can use the name to  enable
       or  disable  the capability when sealing and switching. You can also grant the
       capability to a program even if the capability is  disabled  globally  on  the
       system.
       .
       .
```

For a list of AVAILABLE CAPABILITIES*, see Capabilities [Page 26], above.*

```
AVAILABLE FLAGS

       These flags are used with the ADMIN option "-S".


       LIDS_GLOBAL
              Enable/disable LIDS system-wide.


       RELOAD_CONF
              Reload config files and inode/dev numbers of special programs.


       LIDS   Enable/disable  LIDS  locally  (the shell & childs). This is known as a
              LIDS free session (LFS).


       ACL_DISCOVERY
              Enable/disable LIDS ACL Discovery Mode. When this mode is turned on, if
              something  violates  the  rules,  LIDS  will not prevent the action and
              print out a rule that you can use in future ACLs.  SHUTDOWN  Switch  to
              "SHUTDOWN" State.


EXAMPLES

       Here are some examples of using lidsadm.
       .
       .
```

For a list of EXAMPLES*, see Command-Lind Tools [Page 19], above.*

```
OTHER SOURCES OF INFORMATION.

       Mailing List
              To    subscribe,    unsubscribe,    go     to:     http://lists.source-
              forge.net/lists/listinfo/lids-user
              To   post   a   message   to   the   list,   send   an   e-mail   to:   lids-
              user@lists.sourceforge.net
              Current LIDS archive can be found at:  http://www.geocrawler.com/redir-
              sf.php3?list=lids-user
              An     outdated     searchable     archive     can     be     found     at:
              http://groups.yahoo.com/group/lids


       LIDS FAQ
              The LIDS FAQ is located at:
              http://www.lids.org/lids-faq.lids-faq.html
              or
              http://www.roedie.nl/lids-faq/
```

```
BUGS

       Any bugs found with LIDS itself should be sent to Xie, Phil, or the mailing
       list (lids-user@lists.sourceforge.net). Please include your .config file used
       to compile your kernel, and the lids.conf and lids.cap files located in
       /etc/lids directory. Any errors found in this man page should be sent to
       Sander Klein.


FILES

       /etc/lids/lids.conf - LIDS configuration file.
       /etc/lids/lids.cap - Defines the global capabilities.
       /etc/lids/lids.net - Configuration file for e-mail alerts.
       /etc/lids/lids.pw - Contains the encrypted LIDS password.


SEE ALSO

       lidsconf(8)


AUTHORS

       Huagang Xie <xie@lids.org>

       Philippe Biondi <biondi@cartel-securite.fr>

       Manpage written by Sander Klein <roedie@roedie.nl>


DISTRIBUTION

       The newest version of LIDS can be obtained from http://www.lids.org/ or one of
       it's mirrors. LIDS is (C) 1999-2004 by Huagang Xie(xie@lids.org).
```

21. Booting Into LIDS and Switching to POSTBOOT Mode

21.1. Booting Into LIDS: BOOT Mode

Booting into your LIDS-enabled kernel you should see some LIDS-related messages in `/var/log/messages` (or similar, depending on your `syslog` configuration)...

```
Apr 18 16:11:00 pinback kernel: LIDS: Initializing sysctl
Apr 18 16:11:00 pinback kernel: LIDS: Initializing LIDS ACLs
Apr 18 16:11:00 pinback kernel: LIDS: user space is 32 bit
Apr 18 16:11:00 pinback kernel: LIDS: lidsadm inode 0x9fb7 dev 0x3:a
Apr 18 16:11:00 pinback kernel: LIDS: ACL Discovery: OFF, Effective \
    Capability: 7fffffff, Total ACLs Count: 15
Apr 18 16:11:00 pinback kernel: LIDS: GLOBAL and BOOT state configuration \
    files loaded
Apr 18 16:11:00 pinback kernel: LIDS: Entering BOOT state
Apr 18 16:11:00 pinback kernel: LIDS: Linux Intrusion Detection System \
    2.2.2 started
```

It's worth examining these messages line by line to see what's going on.

```
Initializing sysctl


Initializing LIDS ACLS


user space is 32 bit


lidsadm inode...


ACL Discovery:  OFF


GLOBAL and BOOT state configured


Entering BOOT state


Linux Intrusion Detection System Started
```

21.2. Switching to POSTBOOT Mode

Immediately after booting the LIDS-enabled kernel is not yet fully functional: while LIDS File ACLs are enforced, LIDS Capability ACLs are not and the kernel is not "sealed" — modules may still be loaded or unloaded. To seal the kernel and enforce LIDS Capability ACLs issue the command

```
lidsadm -I
```

In `/var/log/messages` (or similar, depending on your `syslog` configuration) you will see

```
Apr 18 16:23:36 pinback kernel: LIDS: Initializing LIDS ACLs
Apr 18 16:23:36 pinback kernel: LIDS: user space is 32 bit
Apr 18 16:23:36 pinback kernel: LIDS: ACL Discovery: \
    OFF, Effective Capability: 3684ce7f, Total ACLs Count: 14
Apr 18 16:23:36 pinback kernel: LIDS: Attaching ACLs to Processes
Apr 18 16:23:36 pinback kernel: LIDS: GLOBAL and POSTBOOT state Config. \
    files loaded
Apr 18 16:23:36 pinback kernel: LIDS: Switching to POSTBOOT state
```

It's again worth examining these messages line by line to see what's going on — we omit those we have seen before (above).

```
GLOBAL and POSTBOOT state Config. files loaded


Switching to POSTBOOT state
```

22. LIDS States

Introductory material on LIDS *states* and ACL *types* can be found in the Quick Start [Page 10].

It is a good idea to switch states automatically after the boot sequence finishes, and before shutting down the system, byt creating a suitable `init` script

```
#!/bin/sh

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

and sym-linking this appropriately, e.g.,

```
/etc/rc2.d/S99lidsinit
```

for Debian, or

```
/etc/rc3.d/S99lidsinit
/etc/rc5.d/S99lidsinit
```

for RedHat and (for all)

```
/etc/rc0.d/K1lidsinit
```

23. LIDS Files

23.1. Configuration Files

When `lidsconf` is used to add ACLs (`lidsconf -A...`) they are stored in the `*.conf` files

```
/etc/lids/lids.conf
         /lids.boot.conf
         /lids.postboot.conf
         /lids.shutdown.conf
```

if `acl_type` is specified [Page 29], the rule is added to the corresponding `.conf` file, otherwise the rule is added to `lids.conf` and is considered `GLOBAL`, i.e, applies across all states. *These files should NOT normally be manually edited.* The following illustrates the contents of a `.conf` file:

```
subject   subj.    subject     RWDAG    inherit   object  object   object          ??
inode     device   path/name                      inode   device   path/name
.
0       : 0      :            : 3    : 0       : 15937  : 780    : /var/log      : 0-0
0       : 0      :            : 7    : 0       : 15939  : 780    : /var/log/wtmp : 0-0
179884  : 778    : /bin/login : 1    : 0       : 33191  : 778    : /etc/shadow   : 0-0
179885  : 778    : /bin/su    : 1    : 0       : 33191  : 778    : /etc/shadow   : 0-0
```

(subject inode and/or device equal to zero means “any file”) and

```
subject   subj.    subject           RWDAG   inh.   obj.    cap.   cap.            ??
inode     device   path/name                        inode   num.
.
.
179884  : 778    : /bin/login      : 16    :  0   : -1    :  7   : CAP_SETUID    : 0-0
179885  : 778    : /bin/su         : 16    :  0   : -1    :  7   : CAP_SETUID    : 0-0
31917   : 779    : /usr/sbin/exim4 : 16    : -1   : -1    : 31   : CAP_PROTECTED : 0-0
```

(capability-associated ACLS are given an object inode equal to `-1`).

The `*.cap` files

```
/etc/lids/lids.cap
         /lids.boot.cap
         /lids.postboot.cap
         /lids.shutdown.cap
```

specify whether each capability is switched off or on by default. Capability settings for a particular state override, i.e., those specified by `lids.*.cap` override global settings (in `lids.cap`).

When `lidsconf` is used to check and compile added ACLs (`lidsconf -C`) updated `*.acl` files are created from the `.cap` and `.conf` files.

```
/etc/lids/lids.boot.acl
/etc/lids/lids.postboot.acl
/etc/lids/lids.shutdown.acl
```

These files are read when the command `lidsadm -S -- +RELOAD_CONF` is issued

Some initial values for LIDS are stored in

```
/etc/lids.ini
```

Finally, an encrypted version of the LIDS password is stored in

```
/etc/lids.pw
```

23.2. Lids Tools

The LIDS Tools are installed, by default in `/sbin`:

```
/sbin/lidsconf
     /lidsadm
```

23.3. `man` Pages

The `man` pages install, by default, in `/usr/local/share/`:

```
/usr/local/share/man/man8/lidsadm.8
                          lidsconf.8
```

If necessary adjust your `MANPATH` environment variable to include this path, e.g.,: `export MANPATH=$MANPATH:/usr/local/share/man`.

23.4. Source Files

You should start with a "vanilla" source from `www.kernel.org`, rather than a tree from your distro, which traditionally unpacked in `/usr/local`

```
/usr/local/src/linux-2.xy.pq/
```

and the corresponding LIDS source

```
/usr/local/src/lids-2.2.2-2.xy.pq/
             /lidstools-2.2.7
```

23.5. Boot Files

A minimum of your new kernel, and the corresponding `System.map`

```
/boot/vmlinuz-<version>
      System.map-<version>
```

optionally the corresponding `config` file for documentation purposes

```
      config-<version>
```

and possibly, depending on your kernel configuration — is it modular, does it require extra drivers, e.g., `scsi.o`> — an `initrd` image

```
     initrd.img-<version>
```

and some modules

```
/lib/modules/<version>//
```

Finally, so you can boot your LIDS-enabled kernel, a GRUB entry,

```
/boot/grub/menu.lst
```

About this page:

Produced from the SGML: `/home/mc/public_html/_unix_security/_reml_grp/unix_sec_kernel_lids.reml`

On: 17/5/2006 at 15:1:18

Options: `reml2 -l nolong -o tex -p single`