BUILD A SECURE SYSTEM WITH LIDS

Xie Huagang (xie@chinacluster.com, http://www.lids.org) Wed Oct 4 00:04:49 CST 2000

LIDS (Linux Intrusion Detection System) is a Linux kernel patch and admin tools(lidsadm) to enhance the the Linus kernel. It implements a security model – reference model – in the kernel and a Mandatory Access Control model in the kernel. In this article, we will talk about the LIDS including what it can do and how to use it to build a secure Linux system.

Contents

1 Why LIDS.

With increasing popularity of Linux on Internet , more and more security holes are found in the application software of the current GNU/Linux system. Many exploits are due to the careless of the programmers, such as Buffer Over Flow, Format String Attack. After the system is compromised by the exploit and hacker get the ROOT privilege, the whole system is controlled under the Intruders.

Thanks to the open source community, We can now get many Linux application source with our wish and modified them to fit our need. So the bugs can be found easily and can also be patched quickly. But when the hole is disclose to the public, and the administrator is too lazy to patch the hole. It is very easy to break into the un-patched system and it is worse that the hacker can get the root shell. With the current GNU/Linux system, he can do whatever he want. This is the problem that LIDS want to solve.

Let's see what's wrong with the current GNU/Linux system.

- File System is unprotected.

  There are many important files, such as /bin/login, in the system. if the hacker break in, he can upload a changed login program to replace /bin/login , so he can re-login without any login name of password. this is often call Trojan house.(?). But the files do not need to change frequently, unless you want to upgrade the system.

- Process is unprotected.

A process running on the system serve some function for the system, such as HTTPD is a web server to serve the web content to the remote client. As a web server system, it is very important to protect the process from illegal terminated. But we can not do anything is the intruder get the root privilege.

- System administration is unprotected.

  Many system administration, such as modules loading/unloading, route setting up, firewall rules, can be modified if the user id is 0. It is unsafe when the intruder getting root privilege.

- Superuser (root) may abuse the rights

  Being a root, he can do whatever he want. Even the capability existing in the current the system. As a root, he can easily change the capability.

At summary, we can see that the access control model in the current Linux system is not enough for building a secure Linux system. we must add a new model in the system to deal with these problems.

This is what LIDS do.

2 Features of LIDS.

The Linux Intrusion Detection System is `a patch and admin tools which enhances the kernel's security`. It implements a `reference monitor and Mandatory access control` in the Linux kernel. When it is in effect, chosen files access, every system/network administration operations, any capability use, raw device, mem and I/O access can be made impossible even for root. It uses and extends the system capabilities bounding set to control the whole system and adds some network and file-system security features in kernel to enhance the security. You can finely tune the security protections online, hide sensitive processes, receive security alerts through the network, and more.

In short, with the security model implementation in the kernel, LIDS provides `A Protection, Detection and Response` in the Linux system.

For more information about the secure model of LIDS, please refer to the `LIDS Hacking HOWTO`.

2.1 Protection.

LIDS provides the following protection,

- LIDS can protect important files and directories on your hard disk no matter what file-system type they reside on, anybody include root can not change the files.
- LIDS can also protect the important process from being killed.
- LIDS can prevent RAW IO operation from an unauthorized program. It can also protect your hard DISK,include MBR protection,etc.
- LIDS can protect your sensitive files on the system to prevent un-authorized users(including root) and unauthorized program to access them. And more.

2.2 Detection.

When someone scan your host, LIDS can detect it and inform the administrator. LIDS can also notice any activity on the system which violates the rules.

2.3 Response.

When someone violate the rules, LIDS can log the detail message about the violated action to the system log file which has been protected by LIDS. LIDS can also send the log message to your mailbox. In this case, LIDS can also shutdown the user's session at once.

3 Build a secure Linux system

With the LIDS features, let's go and see how to build a secure system with LIDS step by step.

3.1 Download LIDS patch and coresponsive official Linux kernel

You can download lids patch and admin tool from *LIDS Home* `<http://www.lids.org>` and *LIDS Ftp Home* `<ftp://ftp.lids.org>` and other mirrors of LIDS around the world, check *LIDS Mirror* `<http://www.lids.org/mirrors/>` for the nearby mirror site.

The patch name will be lids-x.xx-y.y.y.tar.gz, x.xx represents the lids version and the y.y.y represents the Linux kernel version. For example, lids-0.9.9-2.2.17.tar.gz represent lids version 0.9.9 and the coresponsive kernel version is 2.2.17.

You should download the coresponsive kernel version. For example, if you download the lids-0.9.9-2.2.17.tar.gz, you should download the Linux kernel 2.2.17 source code. You can download the kernel source from *Kernel FTP Site* `<ftp://ftp.linux.org>` or other mirror sites.

And then, you need to uncompress the kernel source and LIDS tar-ball.For example, you have get `lids-0.9.9-2.2.17.tar.gz` from www.lids.org and `linux-2.2.17.tar.bz2` from ftp.us.kernel.org.

```
1. uncompress the Linux kernel source code tree.
# cd linux_install_path/
# bzip2 -cd linux-2.2.17.tar.bz2 | tar -xvf -


2. uncompress the lids source code and install the lidsadm tool.
# cd lids_install_path
# tar -zxvf lids-0.9.8-2.2.17.tar.gz
```

3.2 Patch LIDS to official linux kernel

3. patch the lids to the Linux kernel source code.

```
# cd linux_install_path/linux
# patch -p1 </lids_install_path/lids-0.9pre4-2.2.14.patch
```

```
/* link the default source path to lids patched version
# rm -rf /usr/src/linux
# ln -s linux_install_patch/linux /usr/src/linux
```

3.3 configuring the Linux kernel

```
4. configure the Linux kernel
# cd linux
# make menuconfig or make xconfig
```

Now, it it time to configure the Linux kernel, turn the following option on,

```
[*] Prompt for development and/or incomplete code/drivers
[*] Sysctl support

After that, you will find that a new item appear in the bottom of the configuration menu  name "Linux Intrusion Detection System". Entering this menu, turn the

[*] Linux Intrusion Detection System support  (EXPERIMENTAL) (NEW).
```

After configure the LIDS kernel options. exit the configure interface, and compile the kernel.

```
# make dep
# make clean
# make bzImage
# make modules
# make modules_install
```

3.4 Install LIDS and admin tool into the linux system.

4. copy the bzImage to /boot/ and edit the /etc/lilo.conf

```
# cp arch/i386/boot/bzImage /boot/bzImage-lids-0.9.9-2.2.17

/* build admin tools */
# cd lids-0.9.8-2.2.17/lidsadm-0.9.8/
# make
# make install

# less /etc/lilo.conf
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux

image=/boot/vmlinuz-2.2.16-3
        label=linux
        read-only
        root=/dev/hda2
```

```
image=/boot/bzImage-lids-0.9.9-2.2.17
        label=dev
        read-only
        root=/dev/hda2
```

5 running /sbin/lilo to install the new kernel.

```
# /sbin/lilo
```

3.5 configure the LIDS system

Now before reboot, you must configure you lids system to meet you security need. You can define protected files, protected process,etc.

By default, the lidsadm will install the default configuration files in `/etc/lids/`. You must reconfig it with your need. First of all, you can update the inode/dev value of the default lids.conf by.

```
# /sbin/lidsadm -U
```

In the next chapter, we will show you the details about this topic.

3.6 Reboot the system

After you configure the Linux system, reboot your system. When lilo appear, select the lids enable kernel to load. After then, you will enter the wonderful world of LIDS.

3.7 Sealing the kernel.

After your system boots up, do not forget to seal the kernel with lidsadm, you can put the command in latest line of /etc/rc.local.

```
# /sbin/lidsadm -I
```

You can check the LIDS HOWTO to view the details meaning of each options of lidsadm.

3.8 Online administration

After you seal the kernel, your system is now protected by LIDS. You can do some test on it. If you want to change some configuration, such as modify the capability option, you can change your lids security level online by providing a password.

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

After you change you lids configuration value, such as lids.conf, lids.cap. You can also reload the configuration file into the kernel by

```
# /sbin/lidsadm -S -- +RELOAD_CONF
```

4 Configure the LIDS system

In this chapter, we will show you how to configure the lids.

4.1 LIDS configuration directory – "/etc/lids/"

After install lidsadm, you can find that it creates a lids configuration directory in `/etc/lids/`. When the kernel booting up, the configuration info will be read into the kernel to initial the LIDS system.

- lids.conf

  This is the file to store the LIDS ACLs info. It contain the ACLs to define the object's access type. the item can be added or deleted by lidsadm.

- lids.cap

  This file contain all the capability on the system, you can configure which capability you want to enable or disable on the system by edit this file. You can just set "+" in front of the capability name to enable the system or "-" to disable the capability. When install the system, the lids.cap came with a default value, you should change it with you need.

- lids.net

  This file used for configuring the sending alert mail through network. You can define the SMTP server, port, Message title,etc. This file is needed when you configure your kernel with

  ```
  [*]   Send security alerts through network (NEW)
  ```

- lids.pw

  This file used for storing the password generated by `"lidsadm -P"` and It is nessary when you configure the kernel with

  ```
  [*] Allow switching LIDS protections (NEW)
  ```

  Note: you should run "lidsadm -P" before rebooting the kernel if you want to switch the lids protection level.

4.2 Protect files and directories.

Firstly, you must determine which files you will protect. In most case, you may protect the system binary files and system configuration files, such as /usr/, /sbin/, /etc/, /var/log/.

Secondly, you must decide the way to protect the files. LIDS provide 4 protection type:

4.2.1 DENY access to any body.

The file or directory marked as DENY means that nobody can see or modify it. The most sensity file should be configurate as DENY. For example, we can make /etc/shadow as DENY access to anybody,

```
Usage
        lidsadm -A -o file_to_protected -j DENY

# lidsadm -A -o /etc/shadow -j DENY
After reboot or RELOAD the configurate files. you can see,
# ls /etc/shadow
ls: /etc/shadow: No such file or directory
```

After then, you can make some program can access to the files, for example, when login to the system, the /bin/login file need to read the password from the protected file /etc/shadow, but it has been labeled as DENY, so you can do,

```
Usage
        lidsadm -A -s SUBJECT_PROGRAM -o OBJECT_PROGRAM -j READ/WRITE/APPEND
# lidsadm -A -s /bin/login -o /etc/shadow -j READ
```

After the configurate is in effect * reboot the system of RELOAD the configure file) , you can login onto the system but you can not access the /etc/shadow using other command. It is an example of MAC(mandatory access control).

4.2.2 Read Only Files.

The files marked with Read Only means that nobody can change the files. We can think that the following files are in this catalog, /etc/passwd, /bin/passwd, etc.

```
USAGE:
lidsadm -A -o file_to_protect -j READ

example,

1. to protect the whole /sbin/ as read-only.

 # /sbin/lidsadm -A -o /sbin/ -j READ

2. to protect /etc/passwd as read-only

 # /sbin/lidsadm -A -o /etc/passwd -j READ
```

4.2.3 Append Only Files.

Most of the append only files are system log files, such as /var/log/message ,/var/log/secure. The files can only open with append mode and can not truncate or modify its previous contents.

```
USAGE:
lidsadm -A -o filename_to_protect -j APPEND

example,

1. to protect the system log files
```

```
# /sbin/lidsadm -A -o /var/log/message -j APPEND
# /sbin/lidsadm -A -o /var/log/secure -j APPEND

2. to protect the apache httpd log files

# /sbin/lidsadm -A -o /etc/httpd/logs/ -j APPEND
```

4.2.4 WRITE

This type used to define an object can be write to a subject.

4.2.5 Mandatory Access Control in file protection

You can define which subject (program) can access which object(files) with what kind of access mode - READ, APPEND, WRITE.

For example, you can define /home/httpd/ as DENY to anybody and let /usr/sbin/httpd can READ from that directory. In this case, the Web Server can serve as public WEB Server as normal but the contect and program under the /home/httpd/ can not be seen or modified. If the intruder can get the root shell by the security bugs in httpd, he can not even seen the files in the root shell. Even if he can insert some dangerous code in the httpd server by overwrite the stack, he can only read the file under /home/httpd but he can not modified it.

```
# lidsadm -A -o /home/httpd -j DENY
# lidsadm -A -s /usr/sbin/httpd -o /home/httpd -j READ
```

4.2.6 Example

Here is the example from LIDS HOWTO by Philippe Biond,

```
lidsadm -Z
lidsadm -A -o /boot -j READ
lidsadm -A -o /vmlinuz -j READ
lidsadm -A -o /lib -j READ
lidsadm -A -o /root -j READ
lidsadm -A -o /etc -j READ
lidsadm -A -o /sbin -j READ
lidsadm -A -o /usr/sbin -j READ
lidsadm -A -o /bin -j READ
lidsadm -A -o /usr/bin -j READ
lidsadm -A -o /usr/lib -j READ
lidsadm -A -o /var/log -j APPEND
```

An example lids.conf came with lidsadm package in /etc/lids/ after you install lidsadm, you must run "lidsadm -U" to update the inode/dev value and re-configrate it to fit you requirement.

4.3 Protect processes.

4.3.1 UN-killable process

LIDS can protect the process whose parent is init(pid=1), you must configurate the capability (/etc/lids/lids.cap) as below.

```
-29:CAP_INIT_KILL
```

4.3.2 hidden process

You label the process as HIDDEN, so when the process startup, it can not be found by anybody used "ps" or in "/proc".

```
example,
        lidsadm -A -s /usr/sbin/httpd -t -o CAP_HIDDEN -j INHERIT
```

4.4 Protect with capability.

Capabilities are like privileges you can give a process. A root process has all the capabilities. But there exists a capabilities bounding set. In a normal kernel, when you remove a capability from the bounding set, nobody can ever use it again, until next reboot. (see http://www.netcom.com/ spoon/lcap for the normal use).

LIDS modifies this behavior to enable you to switch theses on and off, whenever you want. An access to the `/proc/sys/kernel/cap_bset` is trapped and raise a security alert. lidsadm performs all the job.

You can list all capability in LIDS by running lidsadm, and you can see what the exactly meaning of each capability.

4.4.1 system capability configuration.

The system capability value is stored in `/etc/lids/lids.cap`. You must edit the file to fit your requirement.

We here discuss two of them,

- CAP_SYS_RAWIO

  With this capability on, we can

  ```
  allow ioperm/iopl and /dev/port access,
  allow /dev/mem and /dev/kmem access and
  allow raw block devices (/dev/[sh]d??) access
  ```

  When we disable this capability, we can make all process on the system has no any right to the raw device, such as running lilo. But some process may want this capability to run, such as XF86_SVGA. In this case, we can let the program in the exception list when compile the kernel.

- CAP_NET_ADMIN

  This capability get the following ability,

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

For the security reason, we should disable this to disallow changing network configuration. When it disallow, the firewall rules will not allow to change.

4.4.2 Configurate the lids.cap.

we can enable a capability to set "+" in front of the capability name or "-" to disable the capability.

```
### 0: In a system with the _POSIX_CHOWN_RESTRICTED option defined, this overrides the restriction
### 0: of changing file ownership and group ownership.
#
-0:CAP_CHOWN

### 1: Override all DAC access, including ACL execute access if _POSIX_ACL is defined. Excluding
### 1: DAC access covered by CAP_LINUX_IMMUTABLE.
#
+1:CAP_DAC_OVERRIDE
```

The example above is to set CAP_CHOWN to disable(-) and enable the CAP_DAC_OVERRIDE(+). You should check the file (lids.cap) carefully to determine which capability you will use and which you will not use.

4.4.3 Labeling individual process with capabilities.

You can labeling process with capability to make the process capable to do something that the system capability has been disable.

For example, you can make CAP_SYS_RAWIO as disable(-) in /etc/lids/lids.cap. But you still need to use X Server which must use this capability to start it. so you can do

```
# lidsadm -A -s /usr/X11R6/bin/XF86_SVGA -t -o CAP_SYS_RAWIO -j INHERIT
```

to make XF86_SVGA has the capability of CA_SYS_RAWIO while other program is unable to obtain the CAP_SYS_RAWIO.

4.4.4 Sealing the kernel

After booting the kernel, the system capability is not in effect until you seal the kernel. you `*MUST*` add the following line in /etc/rc.d/rc.local in RedHat System or other bootup initial file in other system,

```
#/sbin/lidsadm -I
```

4.5 Network Security.

LIDS provide some network security enhancement.

4.5.1 network security with capability

With the capability, we can enhance the network security. such as anti snifferring, can not bind to the port lower than 1024, can not change the firewall and routing rules.

so, what I suggest is to view each capability meaning carefully.

4.5.2 Scanner detector in kernel

LIDS provide a scanner detector in kernel in order to detect who had scan your system. The scanner can detect half-open scan, SYN stealth port scan, Stealth FIN, Xmas, or Null scan,etc. Tools like nmap, satan can be detected by the detector.

It is useful when raw socket is disable. In this case, some user space detector based on snifferring will not work. And the detector does not use any socket, it will be more secure than a user space detector.

If you want this feature, you should select it on when compile the kernel.

4.6 Intrusion Responsive system.

When LIDS detect some thing violate the rules defined, it can response to the action by following method.

4.6.1 Logging the message

When someone violate rules, lids_security_log will log a message the klogd, the logging also have the ability to anti_logging_flood. You can set it when compile the kernel.

4.6.2 Logging the message via mail server

Now, LIDS has a new feature to mail the message to your mail account. you can define the mail server IP, the out-coming mail address,etc, in the /etc/lids/lids.net, more detail please refer to that file.

Here is an example

```
MAIL_SWITCH=1

# MAIL_RELAY=hex IP:port
# IP11.1 of the machine that will be directly connected by LIDS
# for relaying its mails. Port is usually 25, but who knows...
MAIL_RELAY=210.73.88.149:25

# MAIL_SOURCE=source machine :
# Name of the source machine, used for the ehlo identification.
```

```
# Note that a bad name here could make the mail relay refuse your
# mails.
MAIL_SOURCE=lids.chinacluster.com
```

In this example, the SMTP server to relay the server is 210.73.88.149, port 25. The mail source is used for EHLO identification.

4.6.3 Handup the console

When user violate the rules, the console will shutdown the user's console.

5 Thanks.

First of all, I want thank my dear kate , who always encourage me to write document of that, this document is dedicated to her.

I also want to thank Philippe Biond and Christophe Long who give many helps to the project. Without them, the project can not even develop to this step.

Many thanks must go to all the LIDS users, without their contribution and discussion, LIDS can not has so many great ideas.

More thanks will go here if you sent your contribution to the LIDS project, especially this documentation error corrections. Any suggestion is welcome!