Protect Your Logs from Tampering

**Use file attributes to prevent intruders from removing traces of their break-in**.

In the course of an intrusion, an attacker will more than likely leave telltale signs of his actions in various system logs. This is a valuable audit trail that should be well protected. Without reliable logs, it can be very difficult to figure out how the attacker got in, or where the attack came from. This information is crucial in analyzing the incident and then responding to it by contacting the appropriate parties involved [**[Hack #100]**](https://www.oreilly.com/library/view/network-security-hacks/0596006438/ch08s06.html). However, if the break-in attempt is successful and the intruder gains root privileges, what’s to stop him from removing the traces of his misbehavior?

This is where file attributes come in to save the day (or at least make it a little better). Both Linux and the BSDs have the ability to assign extra attributes to files and directories. This is different from the standard Unix permissions scheme in that the attributes set on a file apply universally to all users of the system, and they affect file accesses at a much deeper level than file permissions or ACLs [**[Hack #4]**](https://www.oreilly.com/library/view/network-security-hacks/0596006438/ch01s05.html). In Linux you can see and modify the attributes that are set for a given file by using the lsattr and chattr commands, respectively. Under the BSDs, ls -lo can be used to view the attributes, and chflags can be used to modify them. At the time of this writing, file attributes in Linux are available only when using the ext2 and ext3 filesystems. There are also kernel patches available for attribute support in XFS and reiserfs.

One useful attribute for protecting log files is append-only. When this attribute is set, the file cannot be deleted, and writes are only allowed to append to the end of the file.

To set the append-only flag under Linux, run this command:

# **chattr +a**

filename

Under the BSDs, use this:

# **chflags sappnd**

filename

See how the +a attribute works by creating a file and setting its append-only attribute:

# **touch /var/log/logfile**

# **echo "append-only not set" > /var/log/logfile**

# **chattr +a /var/log/logfile**

# **echo "append-only set" > /var/log/logfile**

bash: /var/log/logfile: Operation not permitted

The second write attempt failed, since it would overwrite the file. However, appending to the end of the file is still permitted:

# **echo "appending to file" >> /var/log/logfile**

# **cat /var/log/logfile**

append-only not set

appending to file

Obviously, an intruder who has gained root privileges could realize that file attributes are being used and just remove the append-only flag from our logs by running chattr -a. To prevent this, we need to disable the ability to remove the append-only attribute. To accomplish this under Linux, use its capabilities mechanism. Under the BSDs, use its securelevel facility.

The Linux capabilities model divides up the privileges given to the all-powerful root account and allows you to selectively disable them. In order to prevent a user from removing the append-only attribute from a file, we need to remove the CAP\_LINUX\_IMMUTABLE capability. When present in the running system, this capability allows the append-only attribute to be modified. To modify the set of capabilities available to the system, we will use a simple utility called lcap (<http://packetstormsecurity.org/linux/admin/lcap-0.0.3.tar.bz2>).

To unpack and compile the tool, run this command:

# **tar xvfj lcap-0.0.3.tar.bz2 && cd lcap-0.0.3 && make**

Then, to disallow modification of the append-only flag, run:

# **./lcap CAP\_LINUX\_IMMUTABLE**

# **./lcap CAP\_SYS\_RAWIO**

The first command removes the ability to change the append-only flag, and the second command removes the ability to do raw I/O. This is needed so that the protected files cannot be modified by accessing the block device they reside on. It also prevents access to /dev/mem and /dev/kmem, which would provide a loophole for an intruder to reinstate the CAP\_LINUX\_IMMUTABLE capability. To remove these capabilities at boot, add the previous two commands to your system startup scripts (e.g., /etc/rc.local). You should ensure that capabilities are removed late in the boot order, to prevent problems with other startup scripts. Once lcap has removed kernel capabilities, they can be reinstated only by rebooting the system.

The BSDs accomplish the same thing through the use of securelevels. The securelevel is a kernel variable that can be set to disallow certain functionality. Raising the securelevel to 1 is functionally the same as removing the two previously discussed Linux capabilities. Once the securelevel has been set to a value greater than 0, it cannot be lowered. By default, OpenBSD will raise the securelevel to 1 when in multiuser mode. In FreeBSD, the securelevel is -1 by default.

To change this behavior, add the following line to /etc/sysctl.conf:

kern.securelevel=1

Before doing this, you should be aware that adding append-only flags to your log files will most likely cause log rotation scripts to fail. However, doing this will greatly enhance the security of your audit trail, which will prove invaluable in the event of an incident.