THE LOGGING SYSTEM

For any Linux user, it's crucial to be knowledgeable in the use of the log files. Log files store information about events that occur when the operating system and applications are run, including any errors and security alerts.

THE RSYSLOG LOGGING SYSTEM:

Linux uses a daemon called syslogd to automatically log events on your computer. Several variations of syslog, including rsyslog and syslog-ng, are used on different distributions of Linux, and even though they operate very similarly, some minor differences exist. Since Kali Linux is built on Debian, and Debian comes with rsyslog by default.

Let's take a look at rsyslog on your system. We'll search for all files related to rsyslog. First, open a terminal in Kali and enter the following:

```
Kali> locate rsylog
```

```
/etc/rsyslog.conf
/etc/rsyslog.d
/etc/default/rsyslog
/etc/init.d/rsyslog
/etc/logcheck/ignore.d.server/rsyslog
/etc/logrotate.d/rsyslog
/etc/rc0.d/K04rsyslog
--snip—
```

Numerous files contain the keyword rsyslog—some of which are more useful than others. The one we want to examine is the configuration file rsyslog.conf.

THE RSYLOG CONFIGURATION FILE:

Like nearly every application in Linux, rsyslog is managed and configured by a plaintext configuration file located, as is generally the case in Linux, in the /etc directory. In the case of rsyslog, the configuration file is located at /etc/rsyslog.conf.

```
Kali> nano /etc/rsylog.conf
```

The rsyslog.conf file comes well documented with numerous comments explaining its use. Much of this information will not be useful to you at this moment, but if we navigate down to below line 50, we'll find the Rules section. This is where we can set the rules for what our Linux system will automatically log for us.

THE RSYSLOG LOGGING SYSTEM:

The rsyslog rules determine what kind of information is logged, what programs have their messages logged, and where that log is stored. Scroll to line 50 and you should see something like:

##############
####RULES####
##############
--snip—

Each line is a separate logging rule that says what messages are logged and where they're logged to. The basic format for these rules is as follows:
Syntax> facility.priority action

facility -- the program, such as mail, kernel, or lpr, whose messages are being logged.
priority -- what kind of messages to log for that program.
action -- on the far right, references the location where the log will be sent.

Let's look at each section more closely, beginning with the facility keyword, which refers to whatever software is generating the log, whether that's the kernel, the mail system, or the user.

The following is a list of valid codes that can be used in place of the facility keyword in our configuration file rules:

auth/authpriv	Security/authorization messages
cron	Clock daemons
daemon	Other daemons
kern	Kernel messages
lpr	Printing system
mail	Mail system
user	Generic userlevel messages

An asterisk wildcard (*) in place of a word refers to all facilities. You can select more than one facility by listing them separated by a comma.

The priority tells the system what kinds of messages to log. Codes are listed from lowest priority, starting at debug, to highest priority, ending at panic. If the priority is *, messages of all priorities are logged. When you specify a priority, messages of that priority and higher are logged. For instance, if you specify a priority code of alert, the system will log messages classified as alert and higher priority, but it won't log messages marked as crit or any priority lower than alert.
Here's the full list of valid codes for priority:

- debug
- info
- notice
- warning -- deprecated and we should not use
- warn -- deprecated and we should not use
- error -- deprecated and we should not use
- err -- deprecated and we should not use
- crit
- alert
- emerg -- deprecated and we should not use
- panic -- deprecated and we should not use

The action is usually a filename and location where the logs should be sent. Note that generally, log files are sent to the /var/log directory with a filename that describes the facility that generated them,

such as auth. This means, for example, that logs generated by the auth facility would be sent to /var/log.auth.log.

Example:

1. mail.* /var/log/mail
   This command will log mail events of all priorities to /var/log/mail

2. kern.crit /var/log/kernel
   log kernel events of critical (crit) priority or higher to /var/log/kernel.
3. *.emerge *
   log all events of the emergency (emerg) priority to all logged on users. With these rules, the hacker can determine where the log files are located, change the priorities, or even disable specific logging rules.

AUTOMATICALLY CLEANING UP LOGS WITH LOGROTATE:

Log files take up space, so if you don't delete them periodically, they will eventually fill your entire hard drive. On the other hand, if you delete your log files too frequently, you won't have logs to investigate at some future point in time. You can use logrotate to determine the balance between these opposing requirements by rotating your logs. ***Log rotation is the process of regularly archiving log files by moving them to some other location, leaving you with a fresh log file.*** That archived location will then get cleaned up after a specified period of time. Your system is already rotating log files using a cron job that employs the logrotate utility. You can configure the logrotate utility to choose the regularity of your log rotation with the /etc/logrotate.conf text file.

Kali> nano /etc/logratae.conf

```
# see "man logrotate" for details
# rotate log files weekly
❶ weekly

# keep 4 weeks worth of backlogs
❷ rotate 4

❸ # create new (empty) log files after rotating old ones
 create
❹ # uncomment this if you want your log files compressed
#compress
# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp  we'll rotate them here
/var/log/wtmp {
missingok
monthly
create 0664 root utmp
rotate 1
}
```

First, you can set the unit of time your rotate numbers refer to ❶. The default here is weekly, meaning any number after the rotate keyword always refers to weeks.

Further down, you can see the setting for how often to rotate logs—the default setting is to rotate logs every four weeks ❷. This default configuration will work for most people, but if you want to keep your logs longer for investigative purposes or shorter to clear them out quicker, this is the setting you should change. For instance, if you check your log files every week and want to save storage space, you could change this setting to rotate1. If you have plenty of storage for your logs and want to keep a semipermanent record for forensic analysis later, you could change this setting to rotate 26 to keep your logs for six months or rotate 52 to keep them for one year. By default, a new empty log file is created when old ones are rotated out ❸. As the comments in the configuration file advise, you can also choose to compress your rotated log files ❹.

At the end of each rotation period, the log files are renamed and pushed toward the end of the chain of logs as a new log file is created, replacing the current log file. For instance, /var/log.auth will become /var/log.auth.1, then /var/log.auth.2, and so on. If you rotate logs every four weeks and keep four set of backups, you will have /var/log.auth.4, but no /var/log.auth.5, meaning that /var/log.auth.4 will be deleted rather than being pushed to /var/log/auth.5. You can see this by using the locate command to find /var/log/auth.log log files with a wildcard, as shown here:

```
Kali> locate  /var/log/auth.log.*
```

RMAINING STEALTHY:

Once you've compromised a Linux system, it's useful to disable logging and remove any evidence of your intrusion in the log files to reduce the chances of detection. There are many ways to do this, and each carries its own risks and level of reliability.

REMOVING EVIDENCE:

First, you'll want to remove any logs of your activity. You could simply open the log files and precisely remove any logs detailing your activity, line by line. However, this could be time consuming and leave time gaps in the log files, which would look suspicious. Also, deleted files can generally be recovered by a skilled forensic investigator.

A better and more secure solution is to shred the log files. With other file deletion systems, a skilled investigator is still able to recover the deleted files, but suppose there was a way to delete the file and overwrite it several times, making it much harder to recover. Lucky for us, Linux has a built-in command, appropriately named shred, for just this purpose.
To understand how the shred command works, take a quick look at the help screen by entering the following command:

```
Kali> shred --help
Kali> shred <file>
```

On its own, shred will delete the file and overwrite it several times—by default, shred overwrites four times. Generally, the more times the file is overwritten, the harder it is to recover, but keep in mind that each overwrite takes time, so for very large files, shredding may become time consuming.

Two useful options to include are the -f option, which changes the permissions on the files to allow overwriting if a permission change is necessary, and the –n option, which lets you choose how many

times to overwrite the files. As an example, we'll shred the log files in /var/log/auth.log 10 times using the following command:

```
Kali> shred -f -n 10 /var/log/auth.log.*
```

```
–f      to give us permission to shred auth files
–n      option with the desired number of times to overwrite
```

DISABLING LOGGING:

Another option for covering your tracks is to simply disable logging. When a hacker takes control of a system, they could immediately disable logging to prevent the system from keeping track of their activities. This, of course, requires root privileges. To disable all logging, the hacker could simply stop the rsyslog daemon.

```
Syntax: service service_name  start|stop|restart
Kali> service rsylog stop
```

Now Linux will stop generating any log files until the service is restarted, enabling you to operate without leaving behind any evidence in the log files!