Logcheck administration

Remark

This page is outdated. The current version of this page is at https://redmine.auroville.org.in/projects/public-pages/wiki/Logcheck

Versions

The information on this page is based on working with logcheck on Squeeze, Wheezy and Jessie: 1.3.13, 1.3.15 and 1.3.17.

References

1. Main documentation:
   Online: http://logcheck.org/docs/
   As installed: directories /usr/share/doc/logcheck and /usr/share/doc/logcheck-database.
   The .gz files may conveniently be read using zcat and less. For example:
   `zcat /usr/share/doc/logcheck-database/README.logcheck-database.gz | less`
2. logcheck man page (HTML format): http://linux.die.net/man/8/logcheck

How does logcheck work?

This description applies to a default installation on Debian. The default configuration may be different for other distros. The configuration may have been changed since installation.

Scheduled job and disabling

Logcheck is run via /etc/cron.d/logcheck. /etc/cron.d/logcheck runs logcheck at reboot and at 2 minutes past every hour.

Logcheck's scheduled job can be disabled, for example by:

```
mv /etc/cron.d/logcheck{,.disabled}
```

Selecting messages to mail

Logcheck reads each new line (message) from /var/log/syslog and /var/log/auth.log. For each message:

1. If the message matches a regular expression in one of the files in /etc/logcheck/cracking.d, it is selected for the ATTACK ALERT section of the email and processing continues with the next message.
2. If the message matches a regular expression in one of the files in /etc/logcheck/violations.d then:
   a. If it also matches a regular expression in one of the files in /etc/logcheck/violations.ignore.d it is ignored and processing continues with the next message.
   b. Otherwise the message is selected for the SECURITY EVENTS section of the email and processing continues with the next message
3. If the message matches a regular expression in one of the files in /etc/logcheck/ignore.d.server then:
   a. The message is ignored and processing continues with the next message.
   b. Otherwise the message is selected for the SYSTEM EVENTS section of the email.

Note: in the first two steps, a match selects the message. In the last step a match discards the message.

If logcheck doesn't select any messages for the email, the email is not sent.

Mailing

The subject of the mail is prefixed with "Reboot: " when logcheck is run at boot.

Mail is sent to logcheck.

The "from" address is "logcheck system account".

Terminology and naming

- **filters, patterns and rules** The logcheck documentation uses "filter", "pattern" and "rule" interchangeably, applying them to directories, files and individual regular expressions. On this WIKI page, only "filter" is used. It is used only to mean a single filter (a line in a file). On this page, a file of filters is called a "filter file".
- **server and workstation** are filtering levels, not computer roles. From README.logcheck:
  - "ignore.d.server; as the name implies, this is intended to cut out the routine messages ..."
  - "ignore.d.workstation. "... is only appropriate for relatively sheltered, non-critical machines"
- **Filter file names** The filter file for generic messages is called "logcheck". The rest of the filter files are named after a Debian package and contain filters for messages generated by software in the package.

logcheck emails

Unwanted messages under ATTACK ALERT

If the messages should be filtered out:

1. Enable /etc/logcheck/cracking.ignore.d by setting SUPPORT_CRACKING_IGNORE to 1 in logcheck.conf
2. Develop one or more filters for the messages
3. Put the filter(s) in a filter file in /etc/logcheck/cracking.ignore.d

Note: the messages are discarded and do not appear anywhere in the email.

Unwanted messages under SECURITY EVENTS

If the messages should be filtered out:

1. Develop one or more filters for the messages
2. Put the filter(s) in a filter file in /etc/logcheck/violations.ignore.d

Note: the messages are discarded and do not appear anywhere in the email.

Unwanted messages under SYSTEM EVENTS

This section applies in the most common case when unwanted messages appear in the SYSTEM EVENTS section of the logcheck emails.

The first task is to decide whether the unwanted messages should be filtered out:

- If a message shows there's an issue to be resolved, filtering the message out would hide essential information.
- If a message is generated rarely, it is not worth the work of filtering it out. Server boot messages are a good example.
- Often it is not clear from the message itself whether it shows there's an issue to be be resolved or not. Research work is required.

If the messages should be filtered out, the next task is to develop one or more regular expressions (filters) that match the messages.

For a single filter, when it works and assuming logcheck has the default configuration, the next step is to put it in a file in /etc/logcheck/ignore.d.server – either in an existing file or a new one.

Unwanted boot messages

Typically logcheck sends messages associated with booting the first time it runs after reboot.

This can be fixed by modifying /etc/cron.d/logcheck, inserting a sleep that delays logcheck until the boot sequence has finished:

```
@reboot logcheck if [ -x /usr/sbin/logcheck ]; then sleep 10; nice -n10 /usr/sbin/logcheck -R; fi
```

Explanation: during boot, logcheck runs when the cron daemon is started. This is normally part way through the boot sequence. Any messages already in the logs are sent in the mail with subject "Reboot: ...". This is done because it is not practicable to have logcheck filters for all the messages generated during boot. Any later messages are sent in the next mail, subject to the usual filtering.

Missing messages

This section applies when messages appear in the logs but do not appear in the emails.

Is the log with the message a log that logcheck is searching? logcheck searches the logs configured in /etc/logcheck/logcheck.logfiles (default /var/log/syslog and /var/log/auth.log) or whichever file is nominated by logcheck's -L option.

Is the message matched by a filter in an ATTACK ALERT or SECURITY EVENT ignore filter file?

If the log is being searched and the message is not matched by an ignore filter file, the message must be matched by a SYSTEM EVENTS filter. In a default installation these are in filter files in the /etc/logcheck/server.d directory, otherwise they are in the /etc/logcheck/<report level>.d directory where <report level> is the value of REPORTLEVEL in /etc/logcheck.conf.

Disabling all logcheck emails

Disable the cron job. For example:

```
     mv /etc/cron.d/logcheck{,.disabled}
```

Custom filters

"Custom filters" is used here to mean ones not installed with the logcheck package. They might come from third parties or later versions of logcheck. They might be developed locally.

Filters from third parties

Sometimes software includes filter files, for example snoopy, filters here.

There are some published filter file libraries to extend the logcheck standard filters, for example:

- Ties de Kock's at https://github.com/ties/logcheck-extrarules/tree/master/ignore.d.server
- sdeziel.info's at https://sdeziel.info/local-logcheck/ignore.d.server/
- Blue Light's available by running `git clone git://git.bluelightav.org/logcheck`

When considering third party filters, be aware that they may be designed to suit particular local conditions so not as widely suitable as filters from logcheck and other packages. For example, the Blue Light filter files are designed to suit low reliability Internet connections; in other locations, messages which we filter out as routine would indicate an unusual event requiring attention.

Filters from later versions of logcheck

Sometimes, especially when existing filters fail because a later version of a package produces differently formatted messages, a later version of logcheck has suitable filter files. The latest can be found in the logcheck git repository.

Developing a filter

Find a similar example

For example, suppose logcheck was mailing something like

```
May 23 10:08:33 LS1 nmbd[1199]: *****
```

The part in green is usually the name of a daemon or command. In some cases it is more. For example ovpn-client uses ovpn-client.<configuration file name>.

Find filter files for similar messages by, for example

```
cd /etc/logcheck/ignore.d.server && grep --files-with-matches ovpn-client *
```

Choose one of the filters in the file(s) listed as the basis for developing a new one. If no files are listed, choose any filter; almost all messages match the message format above up to the last ":" except for the part in green in the example.

Structure of a typical filter

A filter is an egrep regular expression, for example:

```
^\w{3} [ :[:digit:]]{11} [._[:alnum:]-]+ (openvpn|ovpn-[._[:alnum:]-]+)\[[[:digit:]]+\]:( ([-_.@[:alnum:]]+/)?[.[:
digit:]]{7,15}:[[:digit:]]{2,5})? Replay-window backtrack occurred \[[[:digit:]]+\]$
```

Notes:

1. To ensure a complete match, filters begin with ^ and end with $.
2. The grey part matches from start of the line through the time stamp and server name. It is common to most (all?) filters.
3. The green part identifies the daemon or command.
4. The blue part matches the process' PID shown in square brackets and followed by a :
5. The rest is specific to the particular message.

Design considerations

When developing a filter, you can usually use parts 1 to 4 above from a similar filter and develop part 5. It should be designed to match the messages you want to filter in or out – and only those messages.

If your regex is too general it may filter messages that show there's an issue to be resolved. For example ...

```
    ^\w{3} [ :[:digit:]]{11} [._[:alnum:]-]+ (openvpn|ovpn-[._[:alnum:]-]+)\[[[:digit:]]+\]:.*$
```

... would filter out all openvpn log messages.

If your regex is too specific you may need several filters to address closely related messages. For example ... [TODO: add example]

If you are not familiar with egrep regular expressions, it may be worth studying some existing logcheck filters alongside samples of the messages they filter to see how they work.

Regex snippets

- **Day of week (three letter, English)** `(Mon|Tue|Wed|Thu|Fri|Sat|Sun)`
- **Disk device name** (with or without /dev/ and partition number) `(/dev/)?(sd[[:lower:]]+[[:digit:]]*)|(dm-[[:digit:]]+)`
- **Domain name** [A-Za-z0-9.-]+\.[A-Za-z]{2,4}
- **Email address** [A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}
- **Hostname** (with or without domain name) `[-[:alnum:]]+(\.[-[:alnum:]]+)*`
  Rationale: adequate and simpler/faster than the accurate regex for a hostname with domain name ([a-zA-Z0-9]([a-zA-Z0-9\-]{0,61}[a-zA-Z0-9])?\.)+[a-zA-Z]{2,6}
- **hh:mm:ss** `[:[:digit:]]{8}`
  Rationale: adequate and simpler/faster than the accurate regex
- **IPv4 address** `[[:digit:]]+(.[[:digit:]]+){3}`
  Alternatively (commonly seen in as-installed filters): `[.0-9]{7,15}`
  Rationale: adequate and simpler/faster than the accurate regex \b(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b
- **IPv4 port number** `[[:digit:]]{1,5}`
  Rational: adequate and simpler/faster than matching 0 to 65535.
- **IPv6 address** `[[:xdigit:]]*:*[:[:xdigit:]]*:+[[:xdigit:]]+`
  Rationale: matches both full IPV6 address and the minimal loopback address ::1. The flexibility of valid IPV6 address representations makes a more precise regex impossible.
- **Month of year (three letter, English)** `(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)`
- **Network interface name** (eth|lo|tap|tun)[[:digit:].]+
- **Network interface chip name (not complete, simplistic)** (ATL1C|ATL1E|atl1c|atl2|bcm5700|bcmemac|bnx2|bnc2x|e100|e1000|e1000e|netxen_nic|uli526x)
- **User name** `[^ ]+`
  Rationale: although there are conventions for user names, they are not universally enforced and a wide variety of names is possible.
- **Time zone abbreviations (three letter)** `[[:upper:]]{3}`
  Rationale: adequate and simpler, faster and easier to maintain than matching valid time zone abbreviations.

Regex references

Extended regular expression references:

- Man pages: grep (includes egrep) and regex (man pages section 7, not 3). Online at http://linux.die.net/man/1/grep and http://linux.die.net/man/7/regex.
- Wikipedia: http://en.wikipedia.org/wiki/Regular_expression#POSIX_Extended_Regular_Expressions.

- The Linux Documentation Project: http://tldp.org/LDP/abs/html/x17046.html.

Testing and debugging

Filters are tested by seeing if they match sample message(s).

Finding sample messages

The first step is to find which log the sample message(s) are in. In a default logcheck installation this will be either /var/log/syslog or /var/log/auth.log. /var/log/syslog is assumed in the examples below.

If /var/log/syslog has been rotated since logcheck found the message(s), they will be in syslog.1 or syslog.*.gz. They can all be searched using:

```
zgrep '<your string>' /var/log/syslog*
```

Running tests

Having the filter(s) in a file in an editor is useful. Then it's easy to change, redo and undo between tests. Assuming the filter(s) to be tested are in /tmp/my_filter and the messages are in /var/log/syslog, they can be tested by:

```
sed -e 's/[[:space:]]*$//' /var/log/syslog | egrep --file /tmp/my_filter
```

Note: the `sed -e 's/[[:space:]]*$//'` emulates logcheck internals which strip trailing spaces from messages.

When the messages are in a compressed log:

```
zcat /var/log/syslog.3.gz | sed -e 's/[[:space:]]*$//' | egrep --file /tmp/my_filter
```

When the messages are in compressed and uncompressed logs:

```
zgrep --no-filename . <list of log files> | sed -e 's/[[:space:]]*$//' \
    | egrep --file /tmp/my_filter
```

If the filter does not work these can help:

1. Progressively right-truncate the regex until it does match. Then the last part discarded contains the error.
2. Use egrep's --only-matching (-o) option to see what the regex is actually matching.

Further information

For more tips on writing and testing filters:

```
zcat /usr/share/doc/logcheck-database/README.logcheck-database.gz | less
```

Notes on README.logcheck-database:

- If using sort to order filter files as suggested, sort's --version-sort (-V) option may be required.
- Multiple local-<package name> files have the advantage of being installable and removable with the associated package.

Installing filters in a filter file

The directory for the filter file is as listed in "Unwanted messages under ATTACK ALERT in the emails", "Unwanted messages under SECURITY EVENTS in the emails" or "Unwanted messages under SYSTEM EVENTS in the emails" above.

Filter file names must be made of upper and lower case letters, digits, underscores, and hyphens. This is because logcheck uses `run-parts --list` to select the filter files.

Choosing the file name

Custom filters can most easily be kept in local-<something> filter files. The alternatives are harder to administer:

- Changing the files installed by the logcheck package means those files will not be upgraded when logcheck is upgraded.
- A single file for all local filters does not allow installing a local filter file at the same time as a package is installed or upgraded (the format of the log messages may change with the upgrade).
- Having multiple local filter files is helpful when a filter has accidentally been added which matches too many messages; individual files can be disabled until the file causing the problem is identified (a powerful technique when the first step is to to disable half the files, and so on in a "binary search").
- If the filters are for messages generated by software not installed by a package (such as locally developed software or software installed from source), it is convenient to have a local-<something> file to install with the software in the same way that it might have files for /etc/cron.daily or /etc/logrotate.d. The local- prefix ensures such a file will not be clobbered by a package that happens to have the same name.

If the local filter extends the filters in an as-installed <package name> filter file, the relationship can be made clear by putting it in local-<same package name>.

If the package name is not obvious from the message, it can usually be found by finding which file corresponds to the message and which package the file comes from. For example, on Debian for message `Jul 12 09:19:55 LS1 named[1329]: listening on IPv4 interface tun0, 10.8.0.1 #53` the package is bind9:

```
    # type named
    named is /usr/sbin/named
    # dpkg -S /usr/sbin/named
    bind9: /usr/sbin/named
```

In case a standard set of local-* filter files are installed on multiple computers:

- The processing load can be reduced by installing them with extension .disabled and only removing it when needed.
- Host-specific filter files can have prefix local-local-

Backups

If updating an existing file, the original can be backed up with a name which is not entirely of upper and lower case letters, digits, underscores, and hyphens; for example local-foo.bak or local-foo~. These files will not be used by logcheck.

Filter file contents

Filter files may have comment lines (beginning with #) and empty lines (containing only none or more spaces and tabs). These are ignored by logcheck.

Filter files can be sorted to help identify duplicate and similar filters. If doing so, using sort's --version-sort (-V) option may avoid surprises.

Example session of developing a filter

The message was "Sep 16 17:33:31 rose gnome-keyring-prompt: could not grab keyboard: already grabbed"

Found the package:

```
root@rose:/etc/logcheck/ignore.d.server# dpkg -L gnome-keyring | grep gnome-keyring-prompt
/usr/share/applications/gnome-keyring-prompt.desktop
/usr/lib/gnome-keyring/gnome-keyring-prompt-3
/usr/lib/gnome-keyring/gnome-keyring-prompt
```

Found no existing filters:

```
root@rose:/etc/logcheck/ignore.d.server# grep gnome-keyring-prompt *
[no output]
```

Found the log with the message:

```
root@rose:/etc/logcheck/ignore.d.server# grep 'could not grab keyboard' /var/log/syslog
root@rose:/etc/logcheck/ignore.d.server# grep 'could not grab keyboard' /var/log/syslog.1
root@rose:/etc/logcheck/ignore.d.server# grep 'could not grab keyboard' /var/log/auth.log
Sep 16 17:33:31 rose gnome-keyring-prompt: could not grab keyboard: already grabbed
```

Developed a filter by copying an existing filter and modifying it. Tested it:

```
root@rose:/etc/logcheck/ignore.d.server# egrep  --file /tmp/my_filter /var/log/auth.log
Sep 16 17:33:31 rose gnome-keyring-prompt: could not grab keyboard: already grabbed
```

Added filter to existing local filter file:

```
root@rose:/etc/logcheck/ignore.d.server# cat /tmp/my_filter >> local-gnome-keyring
```

Gotchas

If a successfully tested filter does not work in production:

- Are there any trailing spaces on the message in the log? logcheck automatically removes these before looking for filter matches. This behaviour can be mimicked during testing by:

  ```
      sed -e 's/[[:space:]]*$//' /var/log/<log name> | ...
  ```

If a valid filter in a filter file in /etc/logcheck/ignore.d.server filter does not work:

- Does the message match a filter in a filter file in /etc/logcheck/cracking.d or /etc/logcheck/violations.d? Note: if this is the case, the unfiltered message would not be in the SYSTEM EVENTS section of the emails.