Update #17 - Installing Root Kit Detection on Virtual Private Server

Date: May 30, 2025
Category: Security / System Monitoring

I am going to install rootkit detection on my VPS. This is good practice, although not many Linux servers are attacked in this way.

In this guide, I'll be installing and using `chkrootkit` and `rkhunter`.

Update and Upgrade:

```
sudo apt update && sudo apt upgrade -y
```

Install `chkrootkit` :

```
sudo apt install chkrootkit -y
```

```
[redacted]:~$ sudo apt install chkrootkit -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  chkrootkit
0 upgraded, 1 newly installed, 0 to remove and 13 not upgraded.
Need to get 352 kB of archives.
After this operation, 1079 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 chkrootkit amd64 0.55-4 [352 kB]
Fetched 352 kB in 0s (1996 kB/s)
Selecting previously unselected package chkrootkit.
(Reading database ... 104015 files and directories currently installed.)
Preparing to unpack .../chkrootkit_0.55-4_amd64.deb ...
Unpacking chkrootkit (0.55-4) ...
Setting up chkrootkit (0.55-4) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.
```

Run the scan:

```
sudo chkrookit
```

```
Searching for suspicious files and dirs, it may take a while... The following suspicious files and directories were foun
d:
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/basic/file/.htpasswd
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/basic/file/.htaccess
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/basic/authz_owner/.htpasswd
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/basic/authz_owner/.htaccess
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/noentry/.htaccess
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/digest_wrongrelm/.htpasswd
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/digest_wrongrelm/.htaccess
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/digest_anon/.htpasswd
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/digest_anon/.htaccess
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/digest_time/.htpasswd
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/digest_time/.htaccess
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/digest/.htpasswd
/usr/lib/python3/dist-packages/fail2ban/tests/files/config/apache-auth/digest/.htaccess
/usr/lib/modules/5.15.0-140-generic/vdso/.build-id
/usr/lib/modules/5.15.0-139-generic/vdso/.build-id
```

There was nothing found in most cases, but since I use `Fail2Ban` , these files are from its own test files, which include `.htpasswd` and `.htaccess` examples. These are **not actually security threats and are normal and expected.**

`chrootkit` **flags these because** `.htaccess` **and** `.htpasswd` **can sometimes hide malicious behavior - but in this context, they're safe.**

The two `.build-id` are also **normal and expected.** They are part of the Linux Kernel's `vdso` (Virtually Dynamically-linked Shared Object), which helps with performance for certain syscalls.

So this is clean, `chrootkit` is doing its job of alerting me of **potentially dangerous file types**, not actual infections.

>

> In this case, all flagged items are **false positives**.

Now I'm going to move on and install `RKHunter`:

1.) Install RKHunter

```
sudo apt update
sudo apt install rkhunter -y
```

2.) Open RKHunter's Config File:

```
sudo nano /etc/rkhunter.conf
```

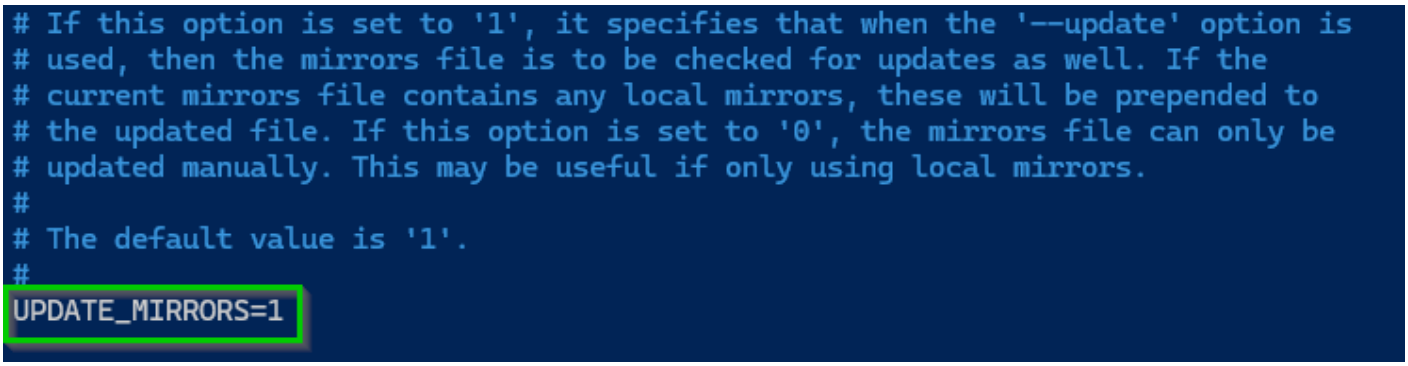
Modify the following

Comment this out: use Ctrl + W to search the file and then type WEB_CMD

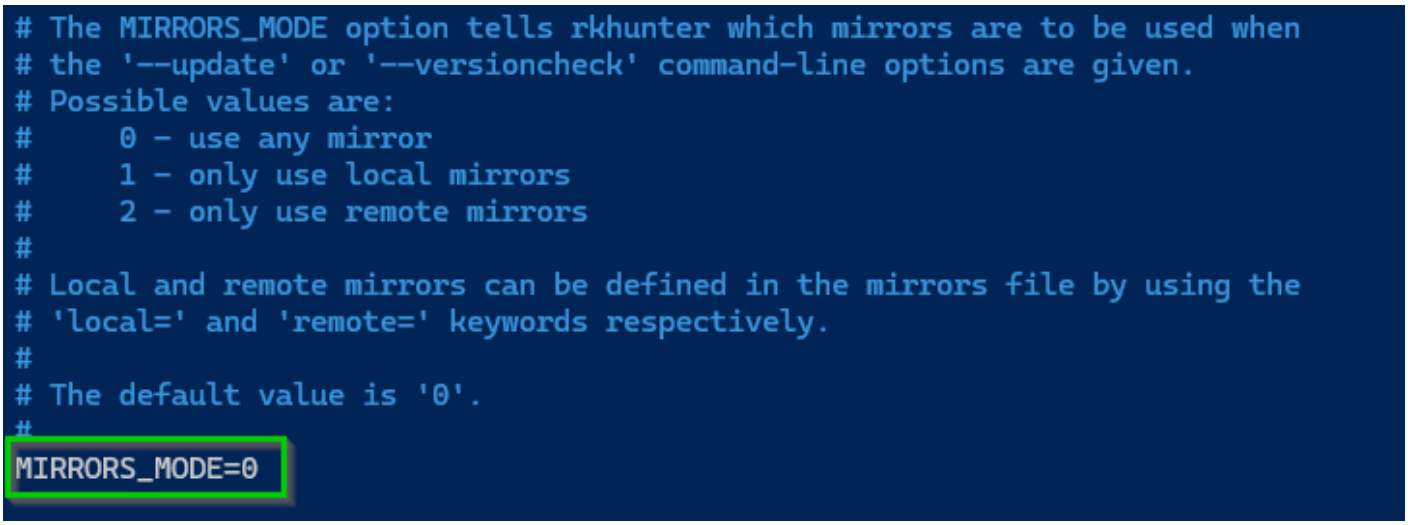
```
#
# *BSD users may want to use the 'ftp' command, provided that it supports the
# HTTP protocol:
#
#     WEB_CMD="ftp -o -"
#
# This option has no default value.
#
WEB_CMD="/bin/false"

#
# Set the following option to '1' if locking is to be used when rkhunter runs.
# The lock is set just before logging starts, and is removed when the program
# ends. It is used to prevent items such as the log file, and the file
# properties file, from becoming corrupted if rkhunter is running more than
# once. The mechanism used is to simply create a lock file in the LOCKDIR
# directory. If the lock file already exists, because rkhunter is already
```

Use CTRL+W to search for UPDATE_MIRRORS and change it to 1



Do the same for MIRRORS_MODE and set that to 0



3.) Update RKHunter's Data Files

```
sudo rkhunter --update
```

> This pulls the latest rootkit definitions and mirror lists.

This is what you want it to look like, since it was freshly installed, nothing new was added:

```
[redacted]:~$ sudo rkhunter --update
[ Rootkit Hunter version 1.4.6 ]

Checking rkhunter data files...
  Checking file mirrors.dat                                  [ No update ]
  Checking file programs_bad.dat                             [ No update ]
  Checking file backdoorports.dat                            [ No update ]
  Checking file suspscan.dat                                 [ No update ]
  Checking file i18n/cn                                      [ Skipped ]
  Checking file i18n/de                                      [ Skipped ]
  Checking file i18n/en                                      [ No update ]
  Checking file i18n/tr                                      [ Skipped ]
  Checking file i18n/tr.utf8                                 [ Skipped ]
  Checking file i18n/zh                                      [ Skipped ]
  Checking file i18n/zh.utf8                                 [ Skipped ]
  Checking file i18n/ja                                      [ Skipped ]
[redacted]:~$
```

Now lets run it:

```
sudo rkhunter --check
```

It will check for possible root kits and the output should look like this:

```
System checks summary
=====================

File properties checks...
    Files checked: 142
    Suspect files: 0

Rootkit checks...
    Rootkits checked : 498
    Possible rootkits: 0

Applications checks...
    All checks skipped

The system checks took: 2 minutes and 49 seconds

All results have been written to the log file: /var/log/rkhunter.log

No warnings were found while checking the system.
```

Now you will have to make key presses by hitting ENTER to continue, in order to automate this we can run:

```
sudo rkhunter --check --sk
```

or the full command:

```
sudo rkhunter --check --skip-keypress
```

This will be useful when saving it to a logfile or scheduling it as a cron job.

It runs the scan **fully unattended**.
You can save the output.
Allows for scripts or email alerts!

Revision #2
Created 30 May 2025 20:53:36 by Nate Nash
Updated 7 June 2025 00:14:24 by Nate Nash