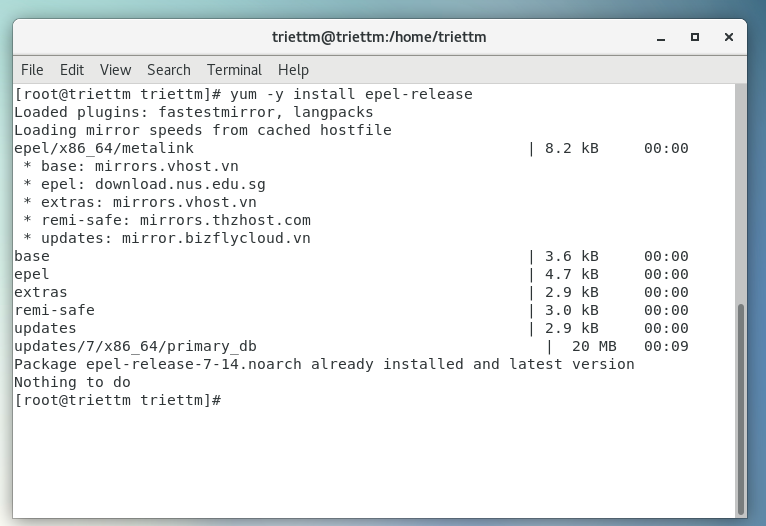
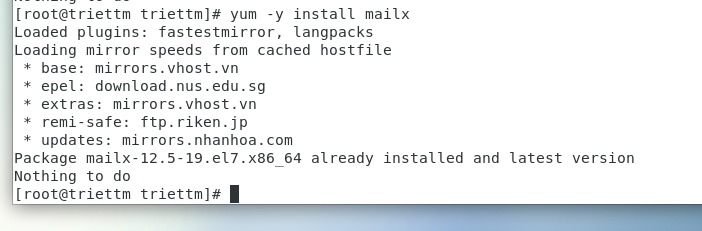
**Install Clamav and maldet**

**Step 1 - Install Epel repository and Mailx**

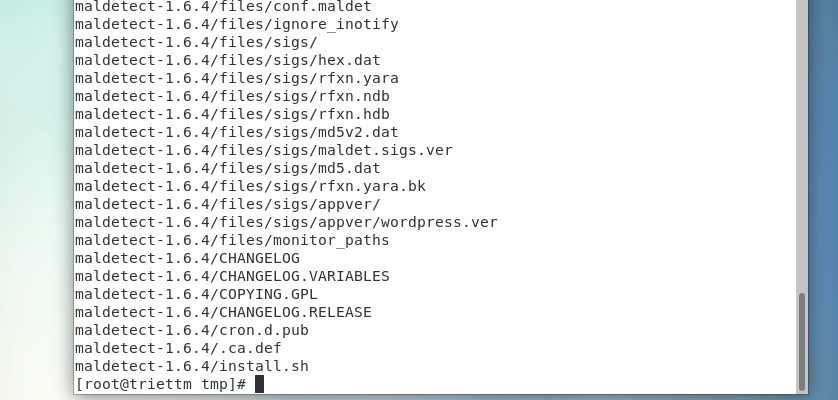
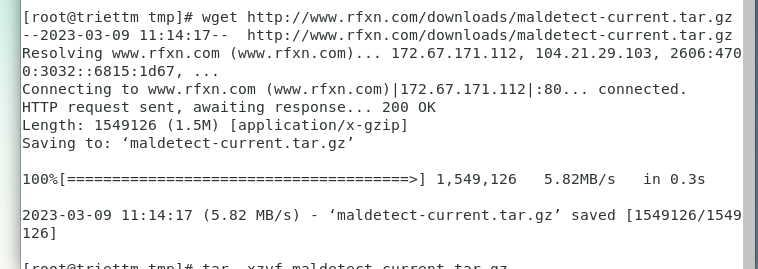
Install the Epel (Extra Packages for Enterprise Linux) repository and the mailx command with yum. We need mailx installed on the system so that LMD can send the scan reports to your email address. 

Install mailx so we can use the mail command on CentOS 7:

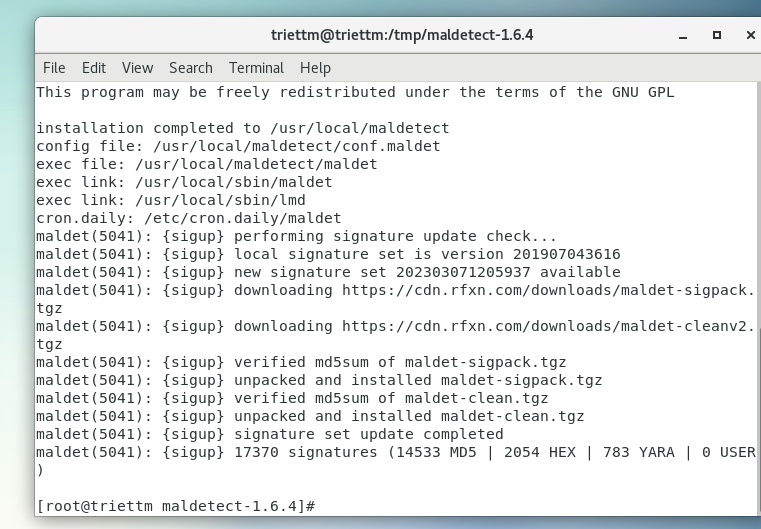
****

**Step 2 - Install Linux Malware Detect (LMD)**

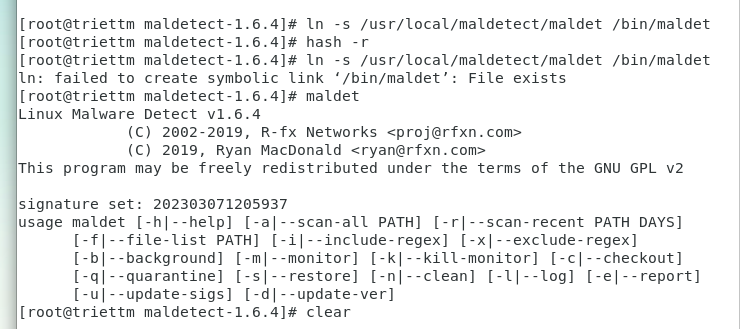
Linux Malware Detect is not available in CentOS or Epel repository, we need to install it manually from source.

****

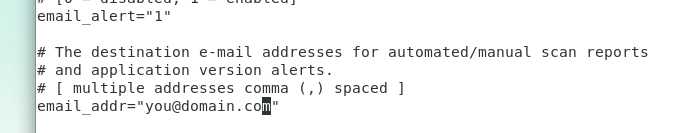
Go to the maldetect directory and run the installer script 'install.sh' as root:

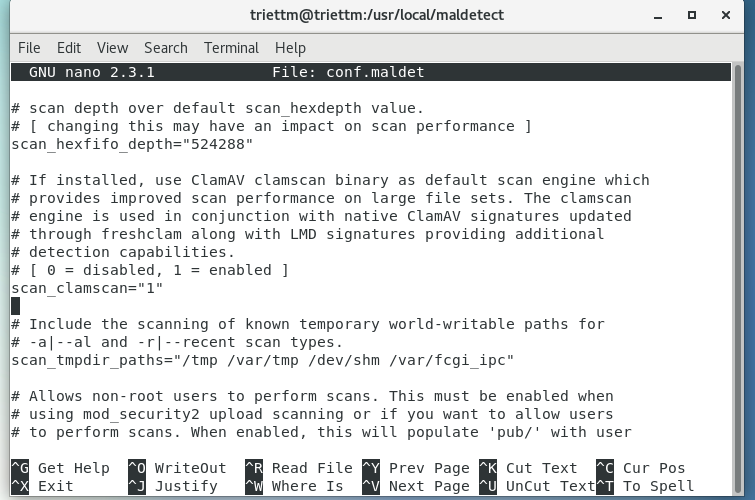
****

Next, make a symlink to the maldet command in the /bin/ directory:

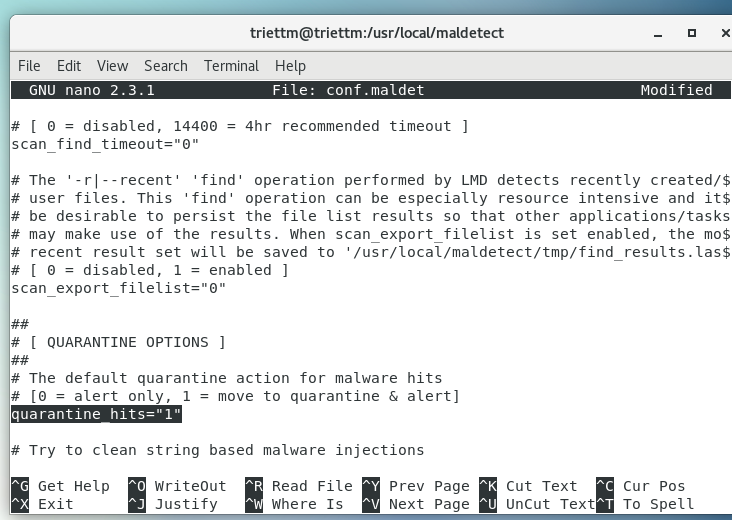
****

**Step 3 - Configure Linux Malware Detect (LMD)**

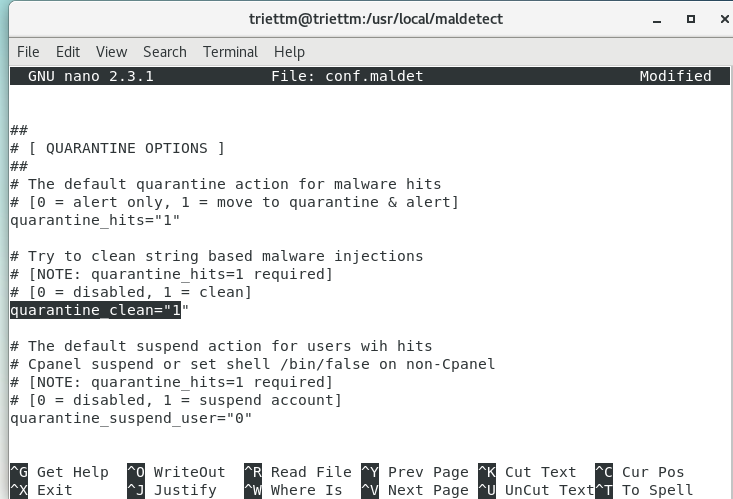
****

****

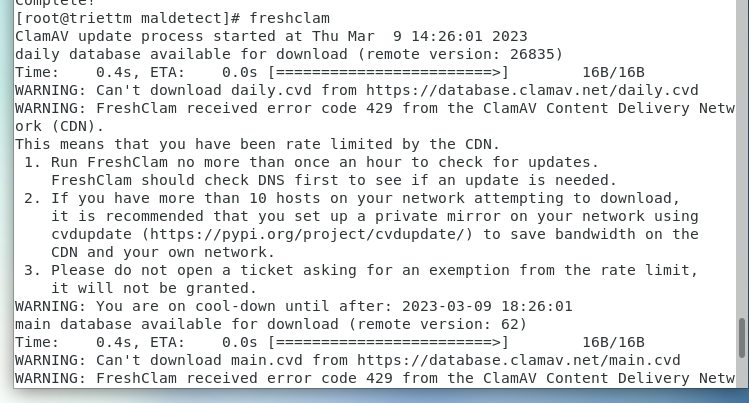
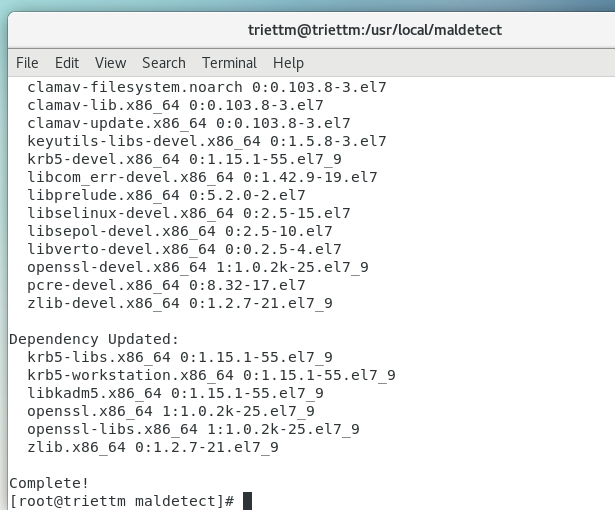
Next, enable quarantining to move malware to the quarantine automatically during the scan process.

****

Enable clean based malware injections.

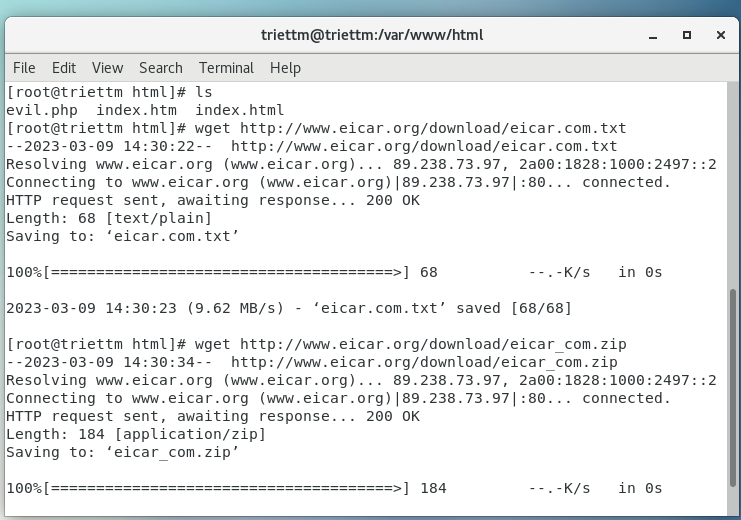
****

**Step 4 - Install ClamAV**

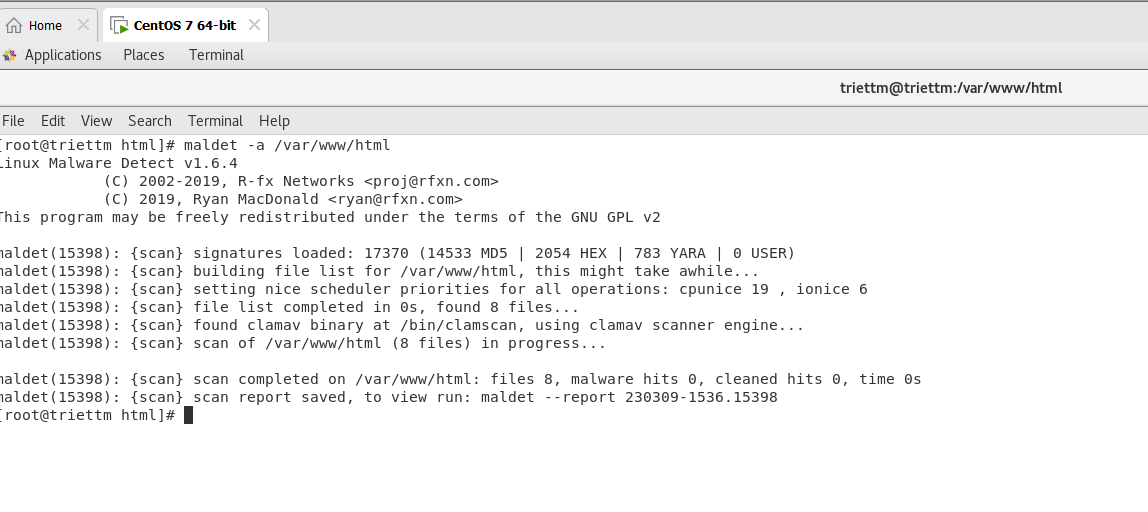
****

**Step 5 - Testing LMD and ClamAV**

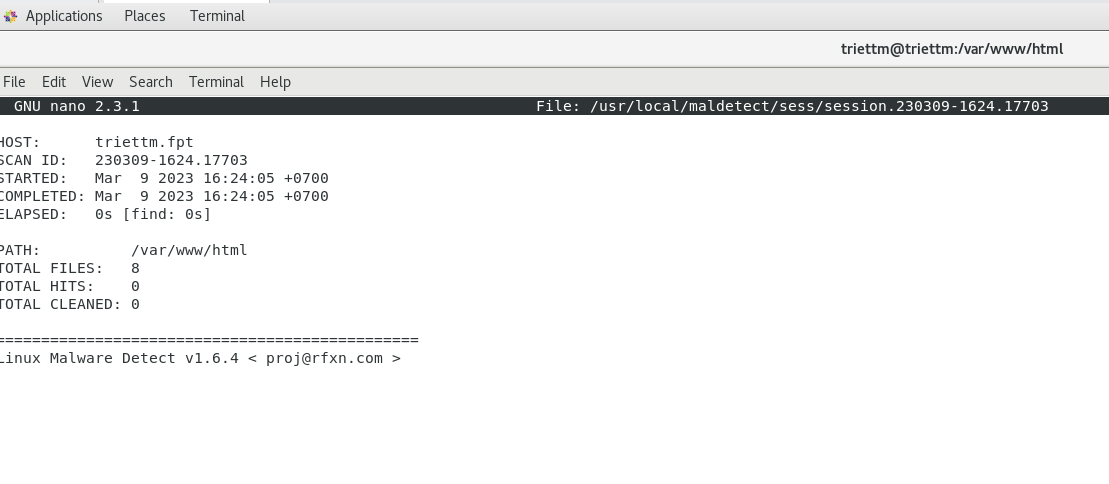
**First we will download some malware for testing purpose**

****

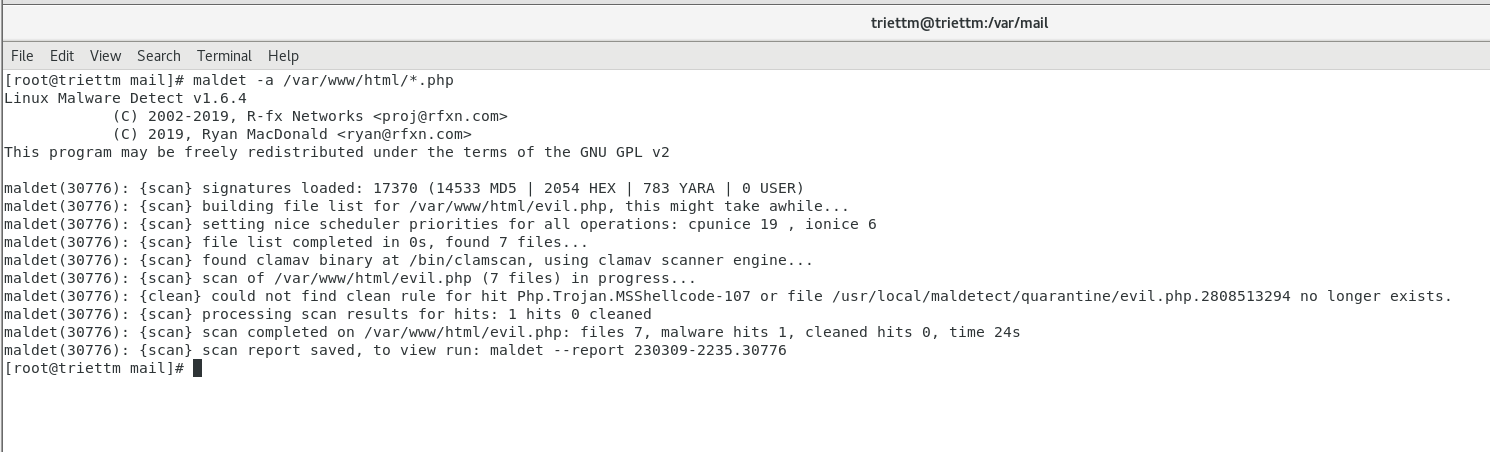
Next, scan the web root directory with the maldet command below:

****

As we can see maldet does not detect any malware file in the folder

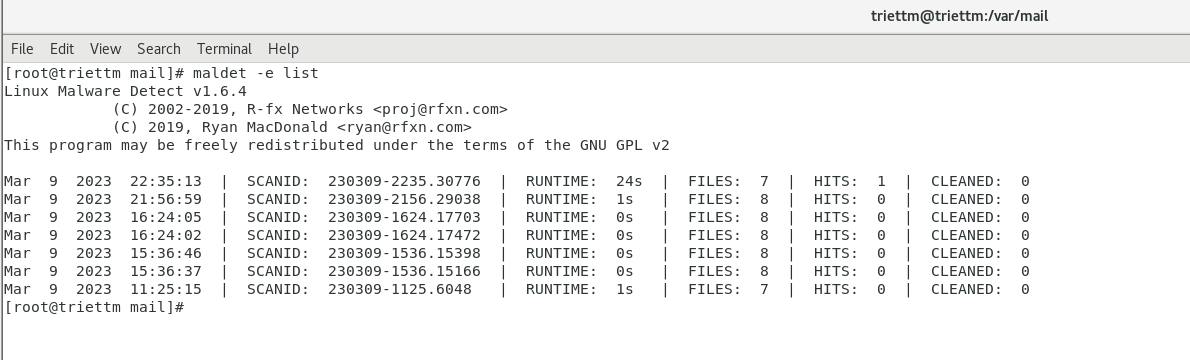


**Step 6 - Other LMD Commands**

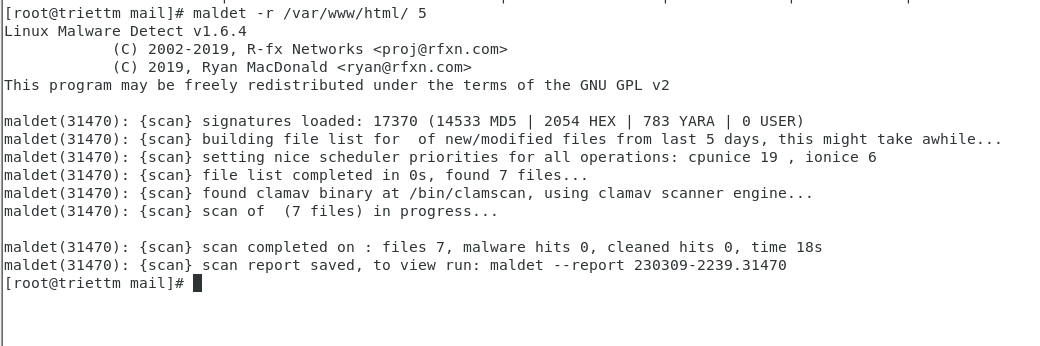


If I specify to detect the php malware file, maldet can detect one of them.

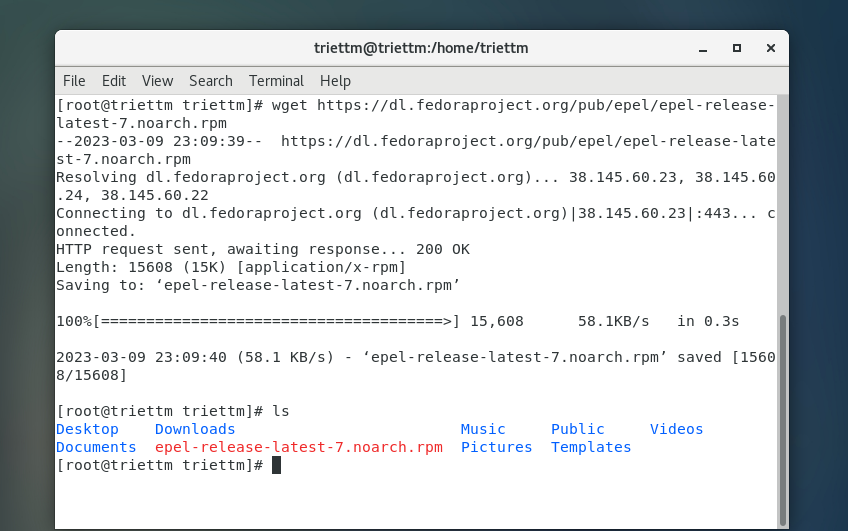
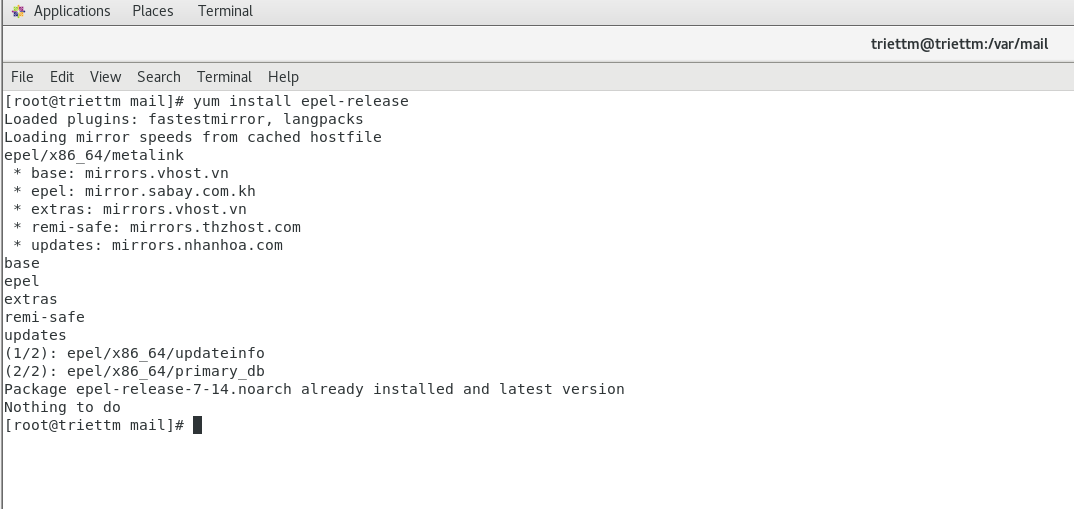
Get a list of all reports:

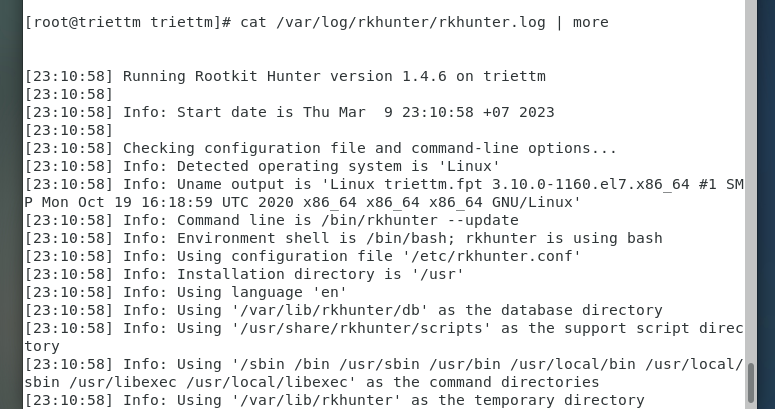
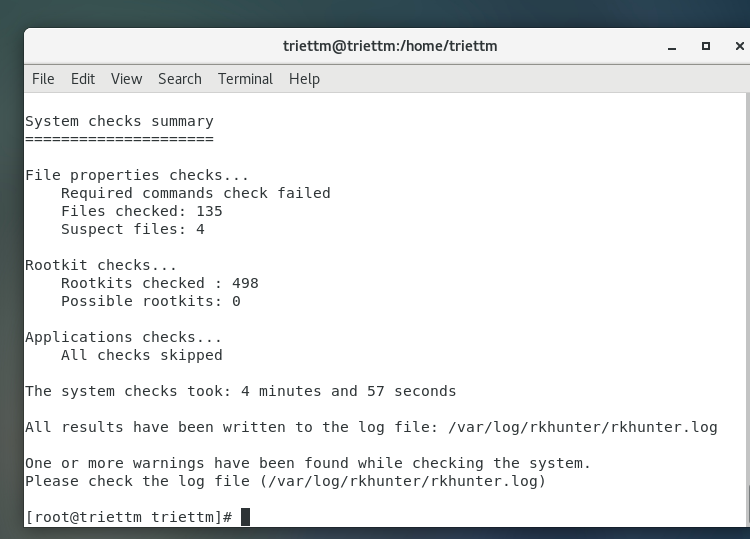
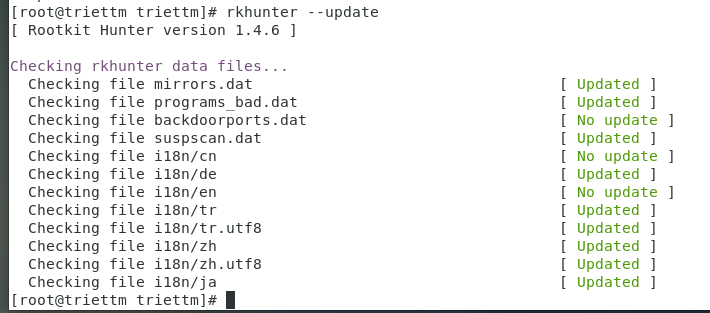
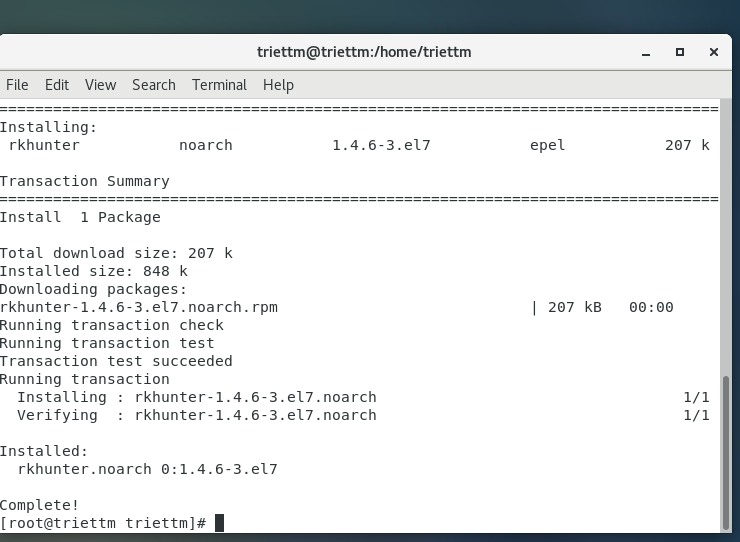
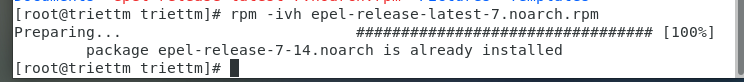


Scan files that have been created/modified in the last X days.



**Install rkhunter**



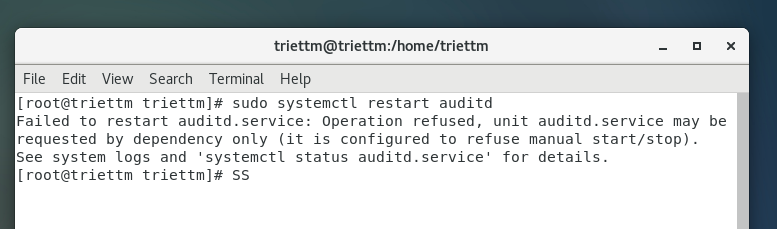


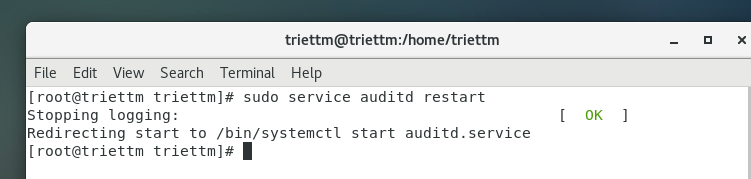
**Controlling the auditd daemon**

On CentOS 7, for some reason that I don't understand, the normal systemctl commands

don't work with auditd. (For all other daemons, they do.) So, on your CentOS 7 machine,

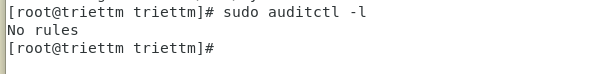
you'll restart the auditd daemon with the old-fashioned service command, like so:



**Creating audit rules**

Okay, let's start with something simple and work our way up to something awesome. First,

let's check to see whether any audit rules are in effect:



**Auditing a file for changes**



As the /etc/passwd have not change anything so the screen print nothing

Here's the breakdown:

-w: This stands for where, and it points to the object that we want to monitor. In

this case, it's /etc/passwd.

-p: This indicates the object's permissions that we want to monitor. In this case,

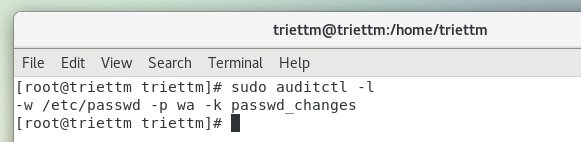
we're monitoring to see when anyone either tries to (w)rite to the file, or tries to

make (a)ttribute changes. (The other two permissions that we can audit are (r)ead

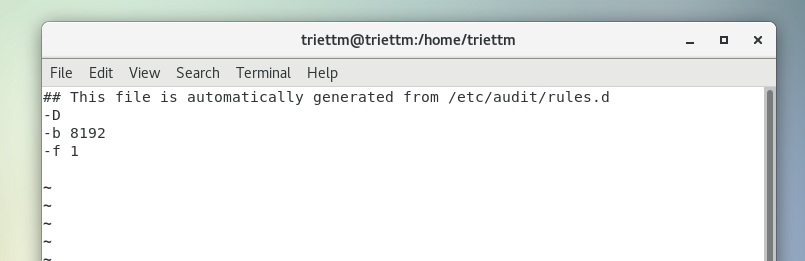
and e(x)ecute.)

-k: The k stands for key, which is just auditd's way of assigning a name to a rule.

So, passwd\_changes is the key, or the name, of the rule that we're creating.



The auditctl -l command shows us that the rule is indeed there.



Here's the breakdown for this file:

-D: This will cause all rules and watches that are currently in effect to be deleted,

so that we can start from a clean slate. So, if I were to restart the auditd daemon

right now, it would read this audit.rules file, which would delete the rule that

I just now created.

-b 8192: This sets the number of outstanding audit buffers that we can have

going at one time. If all of the buffers get full, the system can't generate any more

audit messages.

-f 1: This sets the failure mode for critical errors, and the value can be either 0, 1,

or 2. A -f 0 would set the mode to silent, meaning that auditd wouldn't do

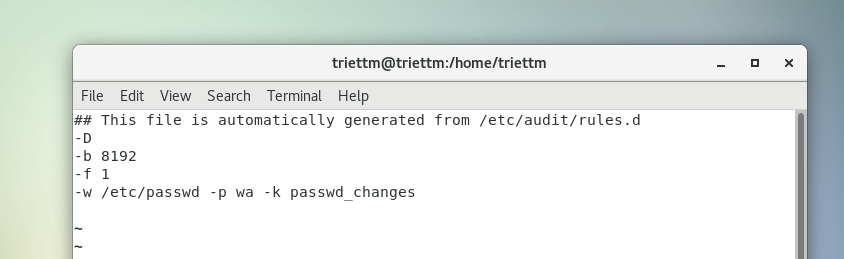
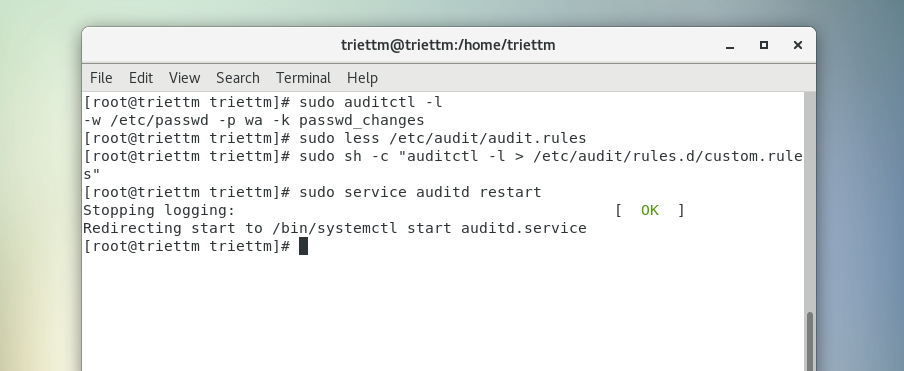
anything about critical errors. A -f 1, as we see here, tells auditd to only report

the critical errors, and a -f 2 would cause the Linux kernel to go into panic

mode. According to the auditctl man page, anyone in a high-security

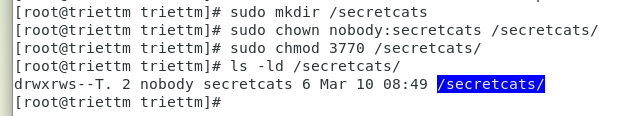
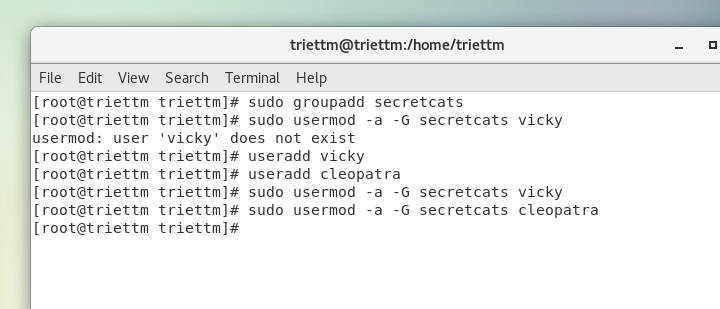
environment would likely want to change this to -f 2. For our purposes though,

-f1 works.



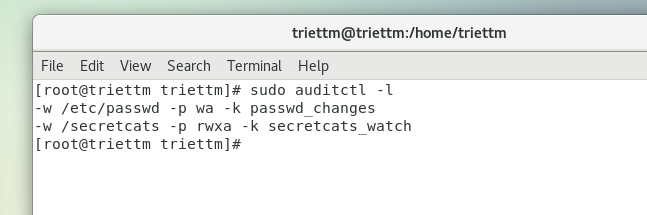
We add new rule to the file

**Auditing a directory**



Vicky and Cleopatra want to be absolutely sure that nobody gets into their stuff, so they

requested that I set up an auditing rule for their directory:

As before, the -w denotes what we want to monitor, and the -k denotes the name of the

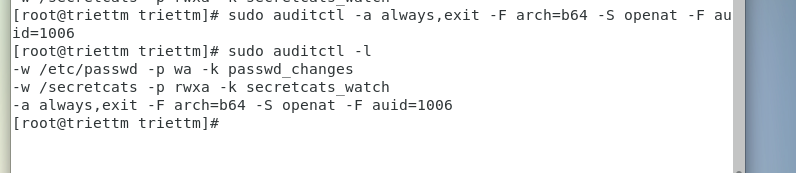
audit rule. This time, I left out the -p option because I want to monitor for every type of

access. In other words, I want to monitor for any read, write, attribute change, or execute

actions. (Because this is a directory, the execute action happens when somebody tries to cd

into the directory.)

**Auditing system calls**



Here's the breakdown:

-a always,exit: Here, we have the action and the list. The exit part means

that this rule will be added to the system call exit list. Whenever the operating

system exits from a system call, the exit list will be used to determine if an audit

event needs to be generated. The always part is the action, which means that an

audit record for this rule will always be created on exit from the specified system

call. Note that the action and list parameters have to be separated by a comma.

-F arch=b64: The -F option is used to build a rule field, and we see two rule

fields in this command. This first rule field specifies the machine's CPU

architecture. The b64 means that the computer is running with an x86\_64 CPU.

(Whether it's Intel or AMD doesn't matter.) Considering that 32-bit machines are

dying off and that Sun SPARC and PowerPC machines aren't all that common,

b64 is what you'll now mostly see.

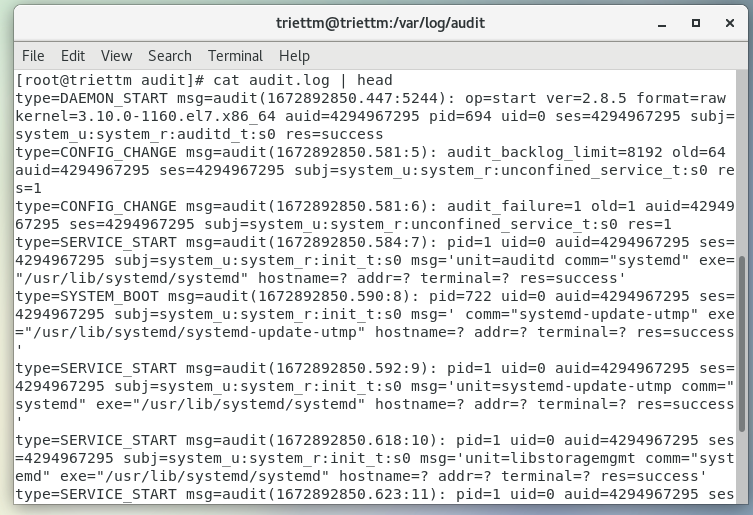
-S openat: The -S option specifies the system call that we want to monitor.

openat is the system call that either opens or creates a file.

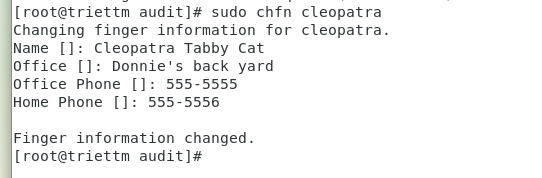
-F auid=1006: This second audit field specifies the user ID number of the user

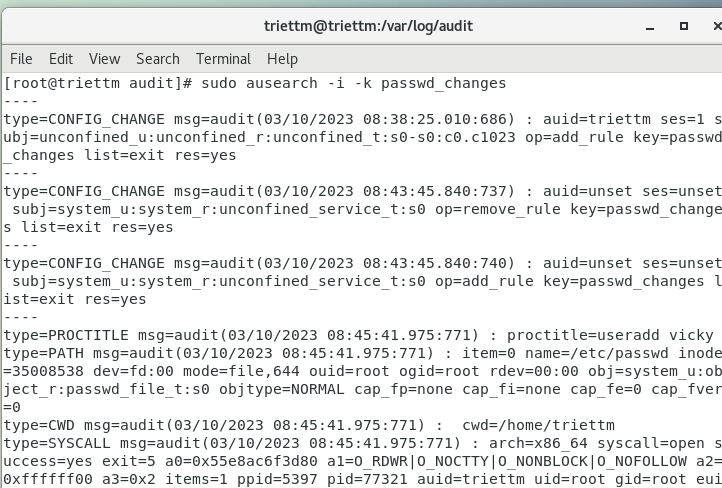
that we want to monitor. (Charlie's user ID number is 1006.)

**Using ausearch and aureport**



**Searching for file change alerts**

****

I'll now use ausearch to look for any audit messages that this event may have generatedHere's the breakdown:

-i: This takes any numeric data and, whenever possible, converts it into text. In

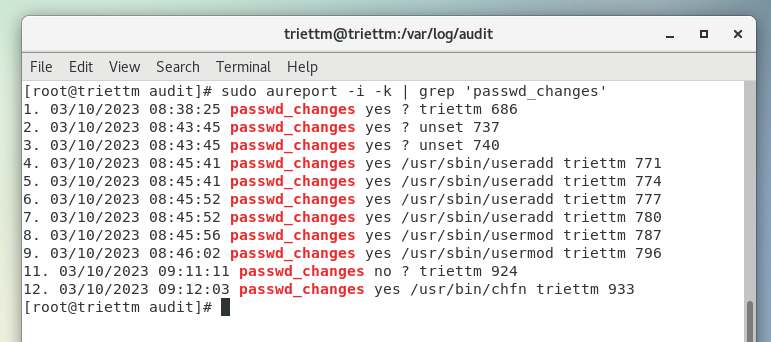
this case, it takes user ID numbers and converts them to the actual username,

which shows up here as auid=donnie. If I were to leave the -i out, the user

information would instead show up as auid=1000, which is my user ID number.

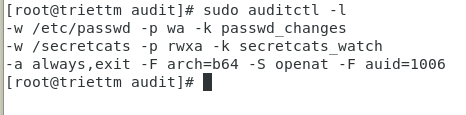
-k passwd\_changes: This specifies the key, or the name, of the audit rule for

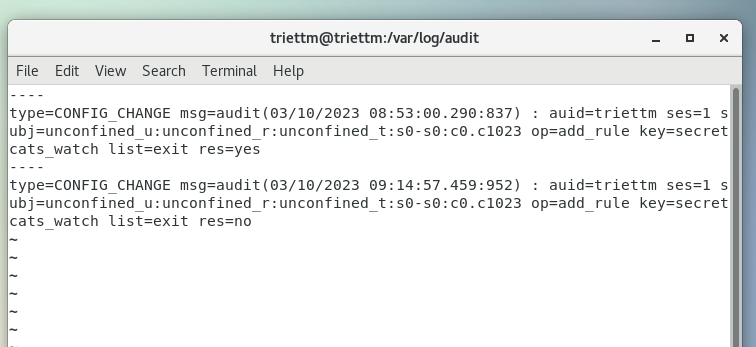
which we want to see audit messages.

****

**Searching for directory access rule violations**

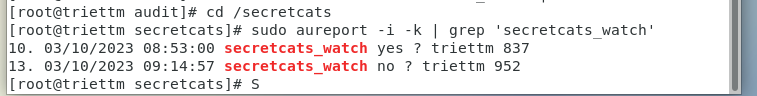
In our next scenario, we created a shared directory for Vicky and Cleopatra and created an

audit rule for it that looks like this

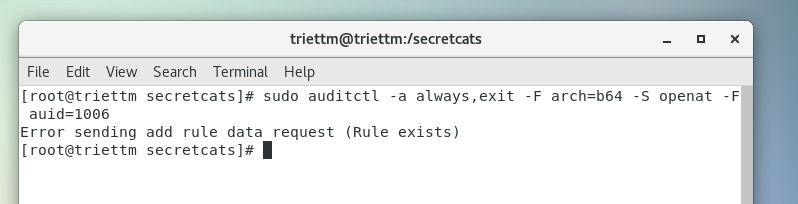
****

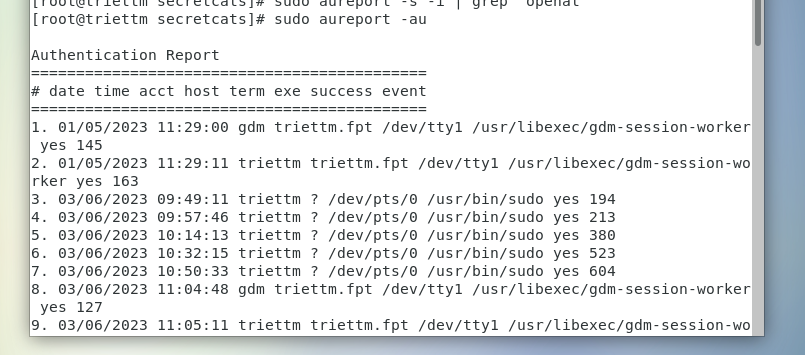
Next, let's say that that sneaky Charlie guy logs in and tries to get into the

/secretcats directory:

****

**Searching for system call rule violations**

****

****