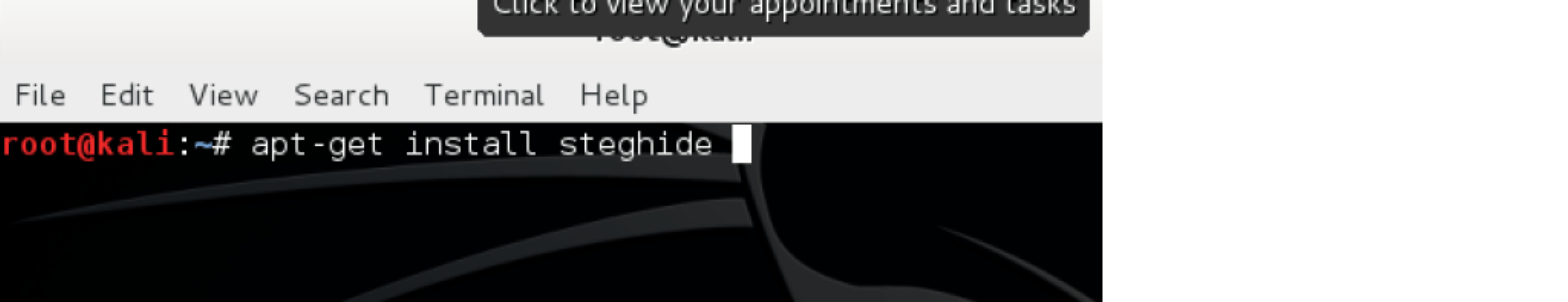
**HIDING SECRET MESSAGES IN IMAGES WITH STEGANOGRAPHY AND METADATA**



Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. The word *steganography* combines the Greek words *steganos*, meaning ‘covered, concealed, or protected’, and *graphein,* meaning ‘writing’. The first recorded use of the term was in 1499 by Johannes Trithemius in *Steganographia*, a treatise on cryptography and steganography which was disguised as a book about magic (Wikipedia).

The general concept is hiding messages that overtly appear to be (or be part of) something innocuous such as images, articles, shopping lists, or a *cover text*. For example, the hidden message may be in invisible

ink between the visible lines of a private letter. Some implementations of steganography lack a shared

secret key and are forms of security through obscurity, whereas key-dependent steganographic schemes adhere to Kerckhoffs’s principle. For more information about this please

see[https://en.wikipedia.org/wiki/Kerckhoffs%27s\_principle.](https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle)

The advantage of steganography over cryptography is that the secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter how unbreakable, arouse interest and

may be incriminating in countries where encryption is illegal or encryption keys are required to be accessible

([https://en.wikipedia.org/wiki/Key\_disclosure\_law).](https://en.wikipedia.org/wiki/Key_disclosure_law)

Thus cryptography is the practice of protecting the contents of a message, whereas steganography is concerned with concealing the fact that a secret message is located within the message content.

Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding within the transported medium, such as a document file, an image file, a program, or a protocol. Media files are ideal for steganographic transmission because of the potentially large size, which makes detection difficult. For example, a sender might start with an innocuous image file and adjust the color of every 100th pixel to correspond to a letter in the alphabet. This is

a technique that is so subtle it would be almost impossible to notice unless looking directly for this adjustment.

Part II: Steganography in Action Using Kali Linux

**NOTE: steghide only works with AU, BMP, JPEG, or WAV files.** If you want to use a different file format, you will need to use a different tool.

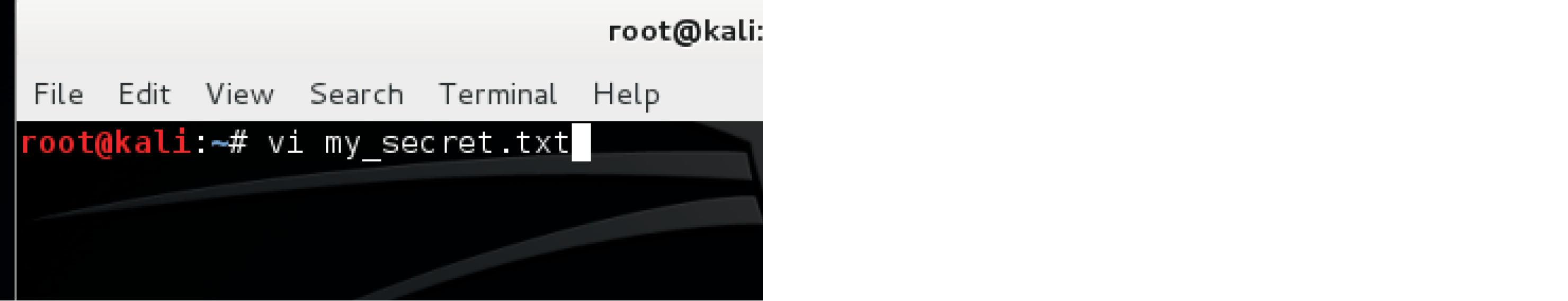
To hide your secret messages inside images, you will need to download some tools. A popular tool I like to use is *Steghide* which you can install on Kali Linux. Here are the steps to using the tool:

**Step 1**: Installing Steghide

You will need to install a tool that lets you hide messages. You only have to install the tool one time and it is not provided as an included module with Kali Linux.

**apt-get install steghide**

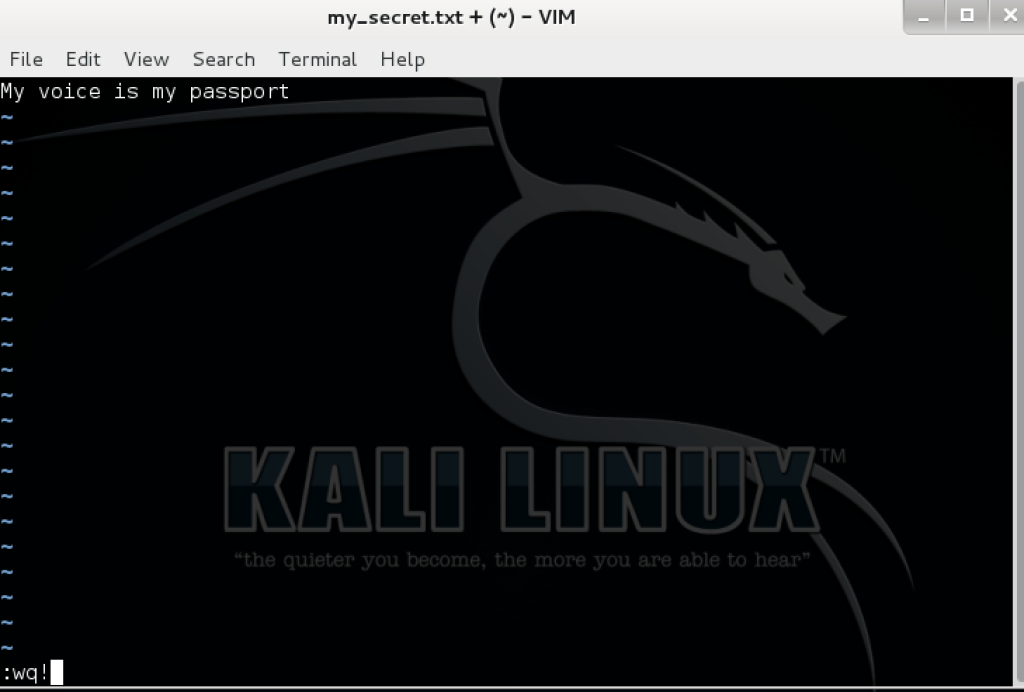
**Step 2:** Create a text file with your hidden message



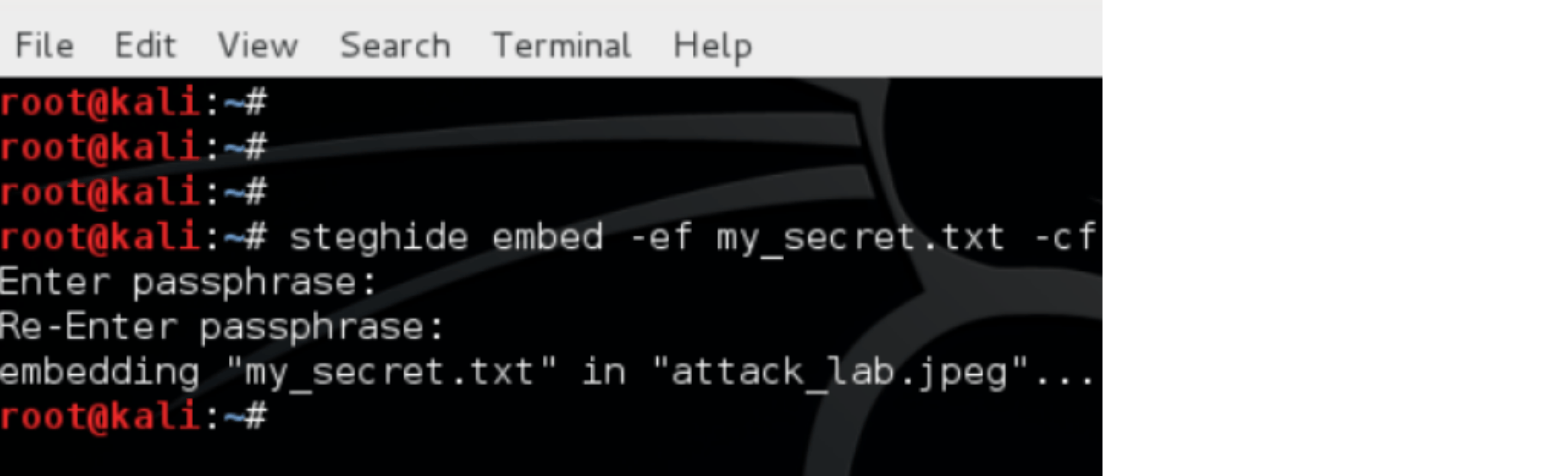
After you get comfortable with Steghide you can use more advanced command line features, but for now let’s

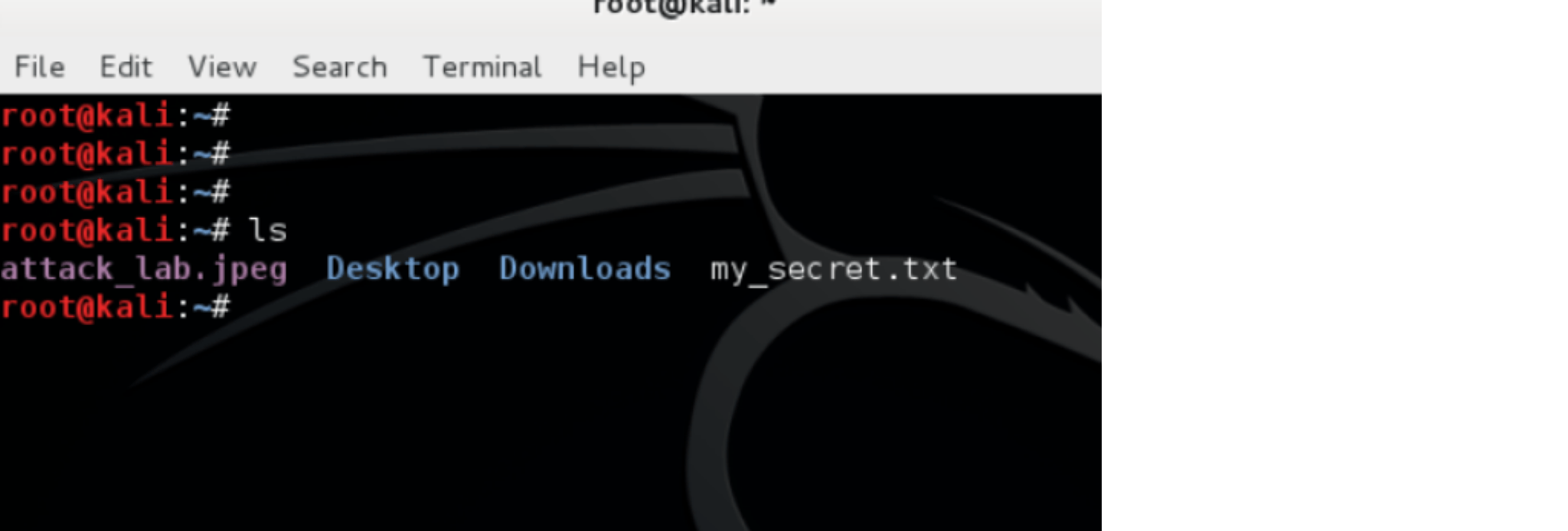
simply create a text file with our hidden message. Feel free to use a text editor of your choice. I simply used **vi** to create a text file called *my\_secret.txt* with the message, “My voice is my passport”.

Don’t forget to save your file!



My file is now saved along with the image file I am going to use.





Next we will use **steghide** to embed the hidden message into the graphic file. The graphic file I used was a simple drawing of my home lab network. As you can see it is an ordinary .jpg file.

**Step 3: Using Steghide to hide your message**

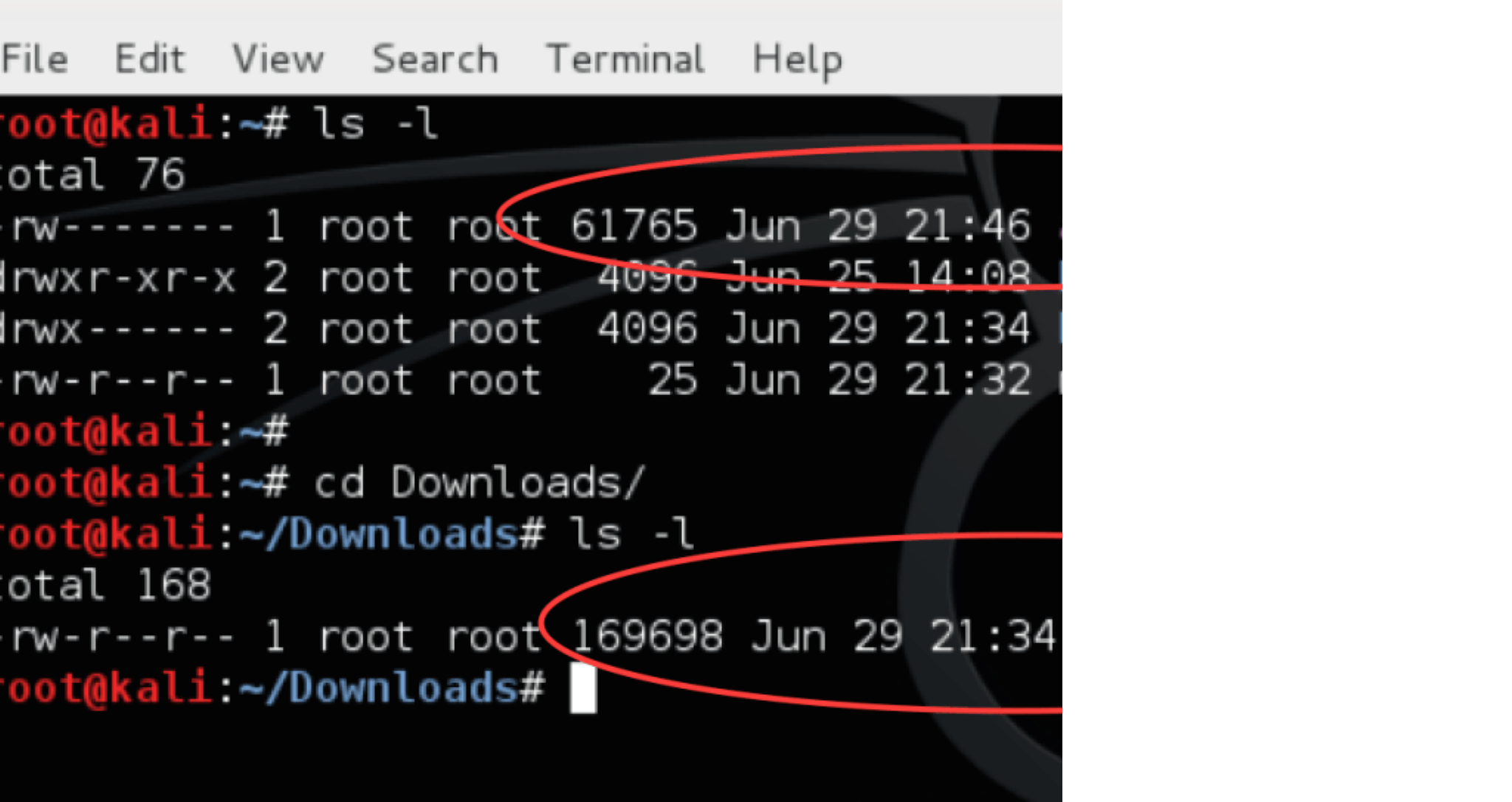
We will use a couple of basic options with Steghide:

1. **embed –ef** is option to embed the secret message *txt*

2. **-cf** is image file we will use

3. jpg is a pic file which obfuscates the message

You’ll see a line to input a password or passphrase within the terminal screen. Create one you can remember or do the unthinkable and write it down. We then successfully embed the message into our image, thus hiding secret the message.



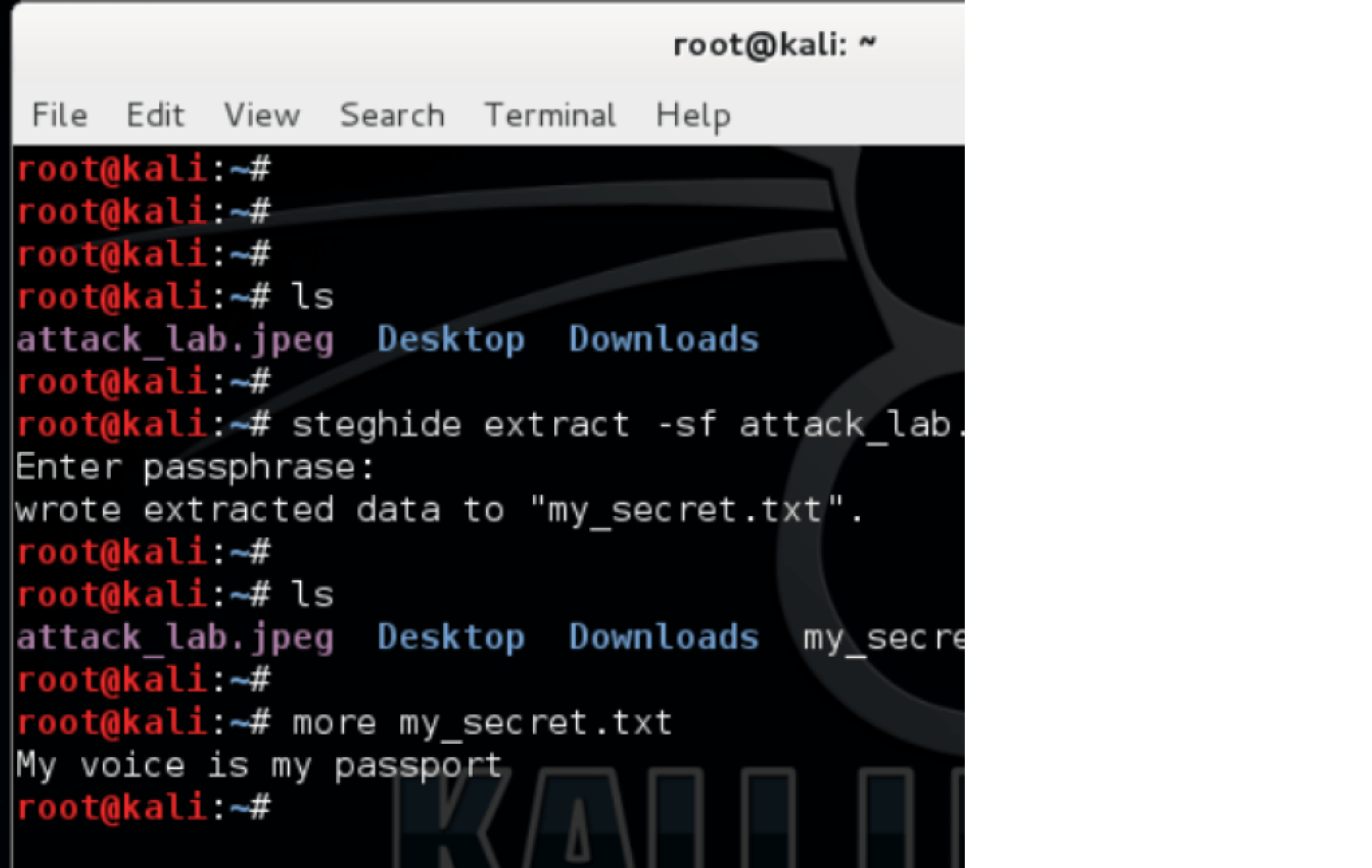
You can also compare the original file to the stenographic file, which is now considerably larger than the original. The hash values of the modified file and the original file are now different as well:

**Step 4: Reading the embedded secret message**

Okay, now that we know how to hide a message, how do we read the message from the image file? We will

use the **extract –sf** command:

**steghide extract –sf attack\_lab.jpeg**

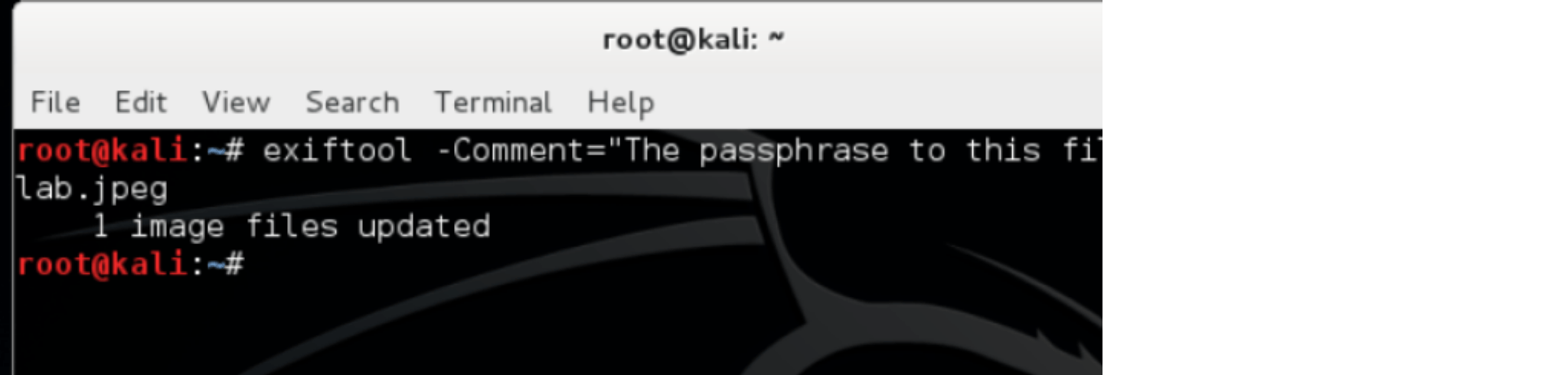


Congratulations**,** you are well on your way on becoming a secret agent! Part III: Exif Metadata

In addition to hiding secret messages in images, the file’s metadata can also contain some very interesting and useful information. Attackers sometime hide information in the metadata as well. ExifTool is a Kali Linux application that allows a user to view and manipulate the image’s metadata.

Let’s go back to our image file (attack\_lab.jpeg) and view the metadata using ExifTool:

**exiftool attack\_lab.jpeg**



You can see there is a variety of interesting information about this image file. If the file has the necessary metadata you can view the following:

1. File creation and modification dates

2. Camera model used to take the picture

3. Geolocation / GPS information contained in the picture

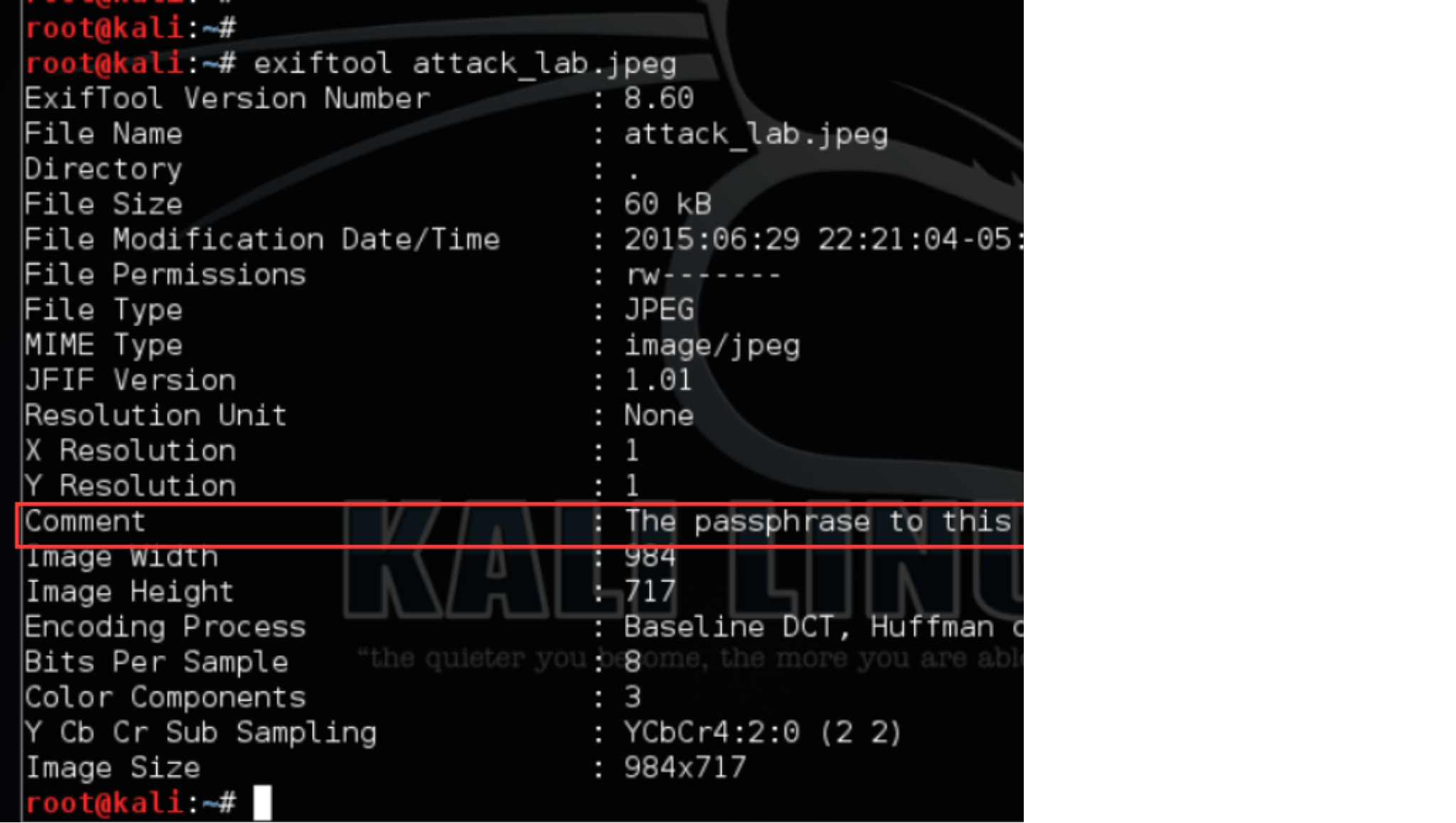
4. Applications used to create or edit the file

**Step 5: Adding a comment in the metadata**

I can also create a comment in the metadata. When I created my hidden message using steganography, I also needed to use a passphrase to encrypt my file. Perhaps I want my passphrase to be easily accessible.

NOTE: This is just an example. I realize this is extremely insecure, because metadata is very easy to scan. We will use ExifTool with the **–Comment=*our comment*** command to achieve this:

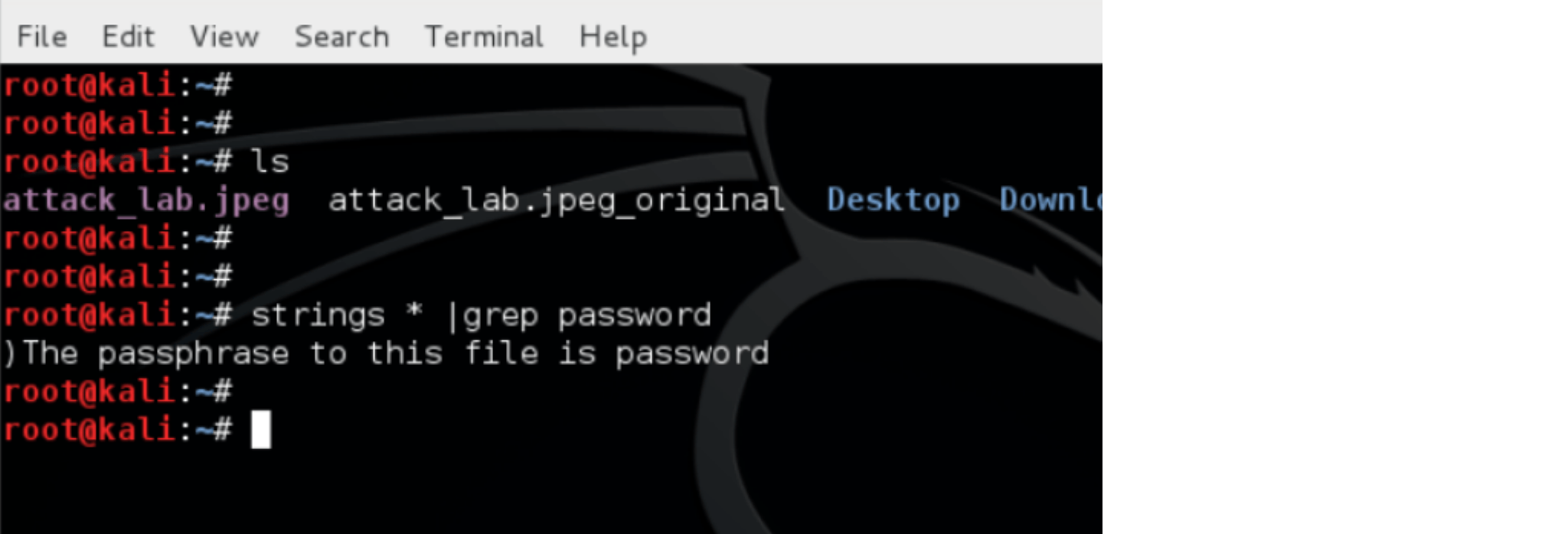
**Step 6: Searching for passwords in metadata**



As a word of caution, I have seen many people think they are being clever by using this method when trying to

hide information. Unlike steganography, information in an image’s metadata is not encrypted.

In the following example I use the **strings** command to search all metadata that might contain passwords:



Part IV: Detecting Steganography

Detecting steganography is very difficult without the proper tools and training. It typically requires a strong knowledge of statistical analysis, which is a subject I do not know very well. Having said that, I do know how to find references for those wanting to further their steganographic detection capabilities:

o [Steganography Countermeasures and detection–](http://en.wikipedia.org/wiki/Steganography#Countermeasures_and_detection) A Wikipedia page worth a read to cover the topic’s basic

information.

o [An Overview of Steganography for the Computer Forensics Examiner–](http://www.garykessler.net/library/fsc_stego.html) Has quite an extensive list of tools and useful information.

o [Steganography Detection–](http://www.uri.edu/personal2/imarcus/stegdetect.htm) This is additional information about steganography.

o [Steganography Detection with Stegdetect–](http://www.outguess.org/detection.php) This is an automated tool for detecting steganographic content in images. It is capable of detecting several different steganographic methods to embed hidden

information in JPEG images. Stegdetect hasn’t been updated in quite a while but it appears to be the best

free tool I could find using a quick search.