Blue Team Level 1 Certification (Standard)

Introduction to BTL1
- ~~Welcome to Blue Team Level 1!~~ 4 Topics
- ~~Lab and Forum Access~~

SECURITY FUNDAMENTALS DOMAIN
- ~~Introduction to Security Fundamentals~~ 1 Topic
- ~~Soft Skills~~ 7 Topics
- ~~Security Controls~~ 5 Topics | 1 Quiz
- ~~Networking 101~~ 6 Topics | 1 Quiz
- ~~Management Principles~~ 4 Topics | 1 Quiz

PHISHING ANALYSIS DOMAIN
- ~~PA1) Introduction to Emails and Phishing~~ 7 Topics | 1 Quiz
- ~~PA2) Types of Phishing Emails~~ 10 Topics | 2 Quizzes
- ~~PA3) Tactics and Techniques Used~~ 12 Topics | 2 Quizzes
- ~~PA4) Investigating a Phishing Email~~ 8 Topics | 2 Quizzes
- ~~PA5) Analysing URLs, Attachments, and Artifacts~~ 8 Topics | 1 Quiz
- PA6) Taking Defensive Actions 12 Topics | 1 Quiz
- PA7) Report Writing 7 Topics | 1 Quiz
- PA8) Phishing Response Challenge 3 Topics | 1 Quiz

THREAT INTELLIGENCE DOMAIN
- TI1) Introduction to Threat Intelligence 7 Topics
- TI2) Threat Actors & APTs 6 Topics | 2 Quizzes
- TI3) Operational Threat Intelligence 7 Topics | 1 Quiz
- TI4) Tactical Threat Intelligence 7 Topics | 1 Quiz
- TI5) Strategic Threat Intelligence 5 Topics | 1 Quiz
- TI6) Malware and Global Campaigns 6 Topics | 1 Quiz

DIGITAL FORENSICS DOMAIN
- DF1) Introduction to Digital Forensics 5 Topics
- DF2) Forensics Fundamentals 10 Topics | 5 Quizzes

Linux Artifacts – User Files

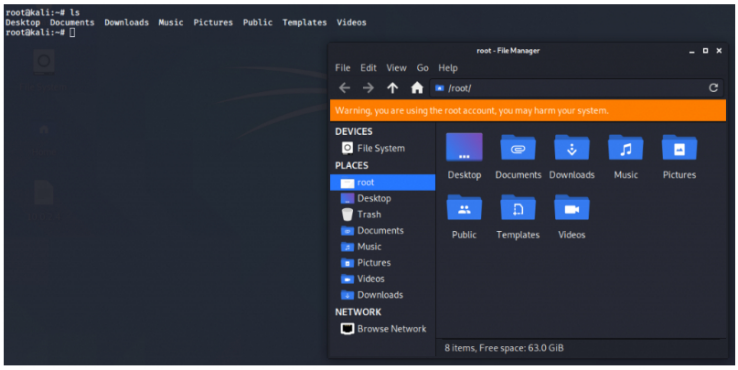
This lesson is going to focus on information of forensic interest regarding user files on a Linux system. There are some great places we can look to collect information, and potentially identify digital evidence. Some of these locations can also be great for using digital forensics skills during incident response. We will look at:

- **.bash_history**
- **Hidden files and directories**
- **Clear files and directories** (Desktop, Trash, Documents, Downloads)
- **Steganography**

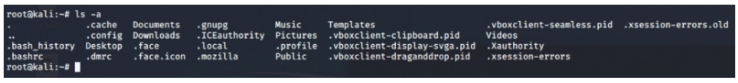
BASH HISTORY

Location

The file .bash_history resides in a user's home file. Can you see it?



No – you can't, and that's because it's hidden. Sort of. If we use the command `ls -a` instead of the usual `ls`, we see that we have identified additional hidden files.



Files or directories that begin with "." are not displayed using the conventional ls command, or in the graphical browser. This is used by system files, but can also be utilized by individuals to hide files (more on that in the next section).

Why is it Interesting?

This file includes a list of commands that have been run by the specific user. We have done two things in the below screenshot:

- The **LEFT** part of the terminal is the output of us printing the contents of `.bash_history` to the terminal.
- The **RIGHT** part of the terminal is us using the `history` command in the terminal.

DF3) Digital Evidence Collection
8 Topics | 1 Quiz
DF4) Windows Investigations
3 Topics | 3 Quizzes
DF5) Linux Investigations
4 Topics | 2 Quizzes
Section Introduction, Linux Investigations
Linux Artifacts – Passwd and Shadow
Activity) Password Cracking
Linux Artifacts – /Var/Lib and /Var/Log
Linux Artifacts – User Files
Activity) End of Section Review, Linux Investigations
DF6) Volatility
3 Topics | 1 Quiz
DF7) Autopsy
4 Topics | 1 Quiz
SECURITY INFORMATION AND EVENT MANAGEMENT DOMAIN
SI1) Introduction to SIEM
7 Topics | 1 Quiz
SI2) Logging
6 Topics | 2 Quizzes
SI3) Aggregation
2 Topics | 1 Quiz
SI4) Correlation
6 Topics | 1 Quiz
SI5) Using Splunk
5 Topics | 2 Quizzes
INCIDENT RESPONSE DOMAIN
IR1) Introduction to Incident Response
8 Topics | 1 Quiz
IR2) Preparation Phase
10 Topics | 2 Quizzes
IR3) Detection and Analysis Phase
7 Topics | 4 Quizzes
IR4) Containment, Eradication, and Recovery Phase
5 Topics | 1 Quiz
IR5) Lessons Learned and Reporting
7 Topics
IR6) MITRE ATT&CK
13 Topics | 2 Quizzes
BTL1 EXAM
Exam Preparation
Using RDP and SSH
How to Start Your Exam

```
root@kali:~# cat .bash_history
apt update && apt upgrade -u
apt update && apt upgrade -y
logout
exit
nmap -sV -T4 10.0.2.4
clear
man nmap
man nmap
clear
nmap -T4 -A 10.0.2.4
clear
nmap -T4 -sV 10.0.2.4
nmap -T4 -sO 10.0.2.4
clear
man nmap
nmap -T4 -o 10.0.2.4
nmap -T4 -sT -O 10.0.2.4
sudo cat /etc/shadow
clear
cat /etc/passwd
root@kali:~#
```

```
root@kali:~# history
    1  apt update && apt upgrade -u
    2  apt update && apt upgrade -y
    3  logout
    4  exit
    5  nmap -sV -T4 10.0.2.4
    6  clear
    7  man nmap
    8  man nmap
    9  clear
   10  nmap -T4 -A 10.0.2.4
   11  clear
   12  nmap -T4 -sV 10.0.2.4
   13  nmap -T4 -sO 10.0.2.4
   14  clear
   15  man nmap
   16  nmap -T4 -o 10.0.2.4
   17  nmap -T4 -sT -O 10.0.2.4
   18  sudo cat /etc/shadow
   19  clear
   20  cat /etc/passwd
   21  history
root@kali:~#
```

They're the same, because the terminal is writing any commands to the .bash.history file. So what's the point of reading the history file if we can just use the history command? Because users can execute the history -c command, which deleted all history from the terminal. On the right, when we run the history command again, it's all gone.

```
root@kali:~# history
    1  apt update && apt upgrade -u
    2  apt update && apt upgrade -y
    3  logout
    4  exit
    5  nmap -sV -T4 10.0.2.4
    6  clear
    7  man nmap
    8  man nmap
    9  clear
   10  nmap -T4 -A 10.0.2.4
   11  clear
   12  nmap -T4 -sV 10.0.2.4
   13  nmap -T4 -sO 10.0.2.4
   14  clear
   15  man nmap
   16  nmap -T4 -o 10.0.2.4
   17  nmap -T4 -sT -O 10.0.2.4
   18  sudo cat /etc/shadow
   19  clear
   20  cat /etc/passwd
   21  history
   22  clear history
   23  history clear
   24  clear
   25  history
root@kali:~# history -c
```

```
root@kali:~# history
    1  clear
    2  history
root@kali:~#
```

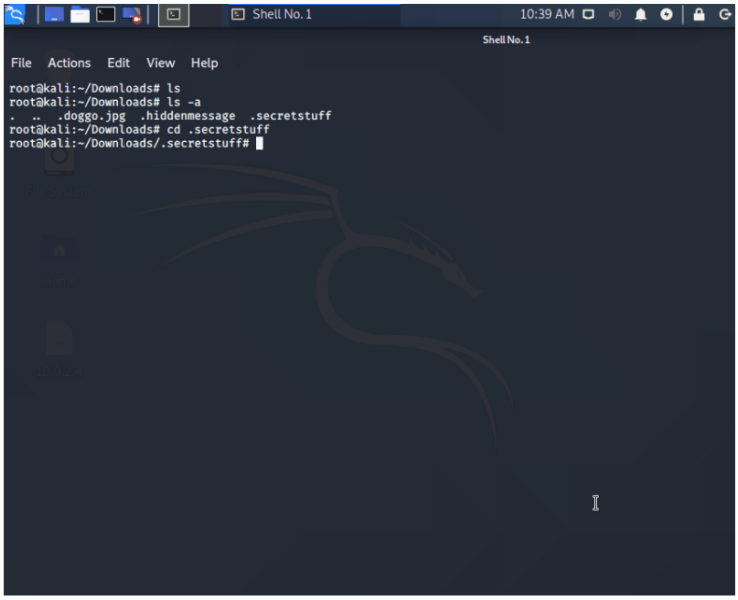
But when we go back and read the history file using cat .bash_history we can see that we still have a record of the commands that have been executed!

```
root@kali:~# cat .bash_history
apt update && apt upgrade -u
apt update && apt upgrade -y
logout
exit
nmap -sV -T4 10.0.2.4
clear
man nmap
man nmap
clear
nmap -T4 -A 10.0.2.4
clear
nmap -T4 -sV 10.0.2.4
nmap -T4 -sO 10.0.2.4
clear
man nmap
nmap -T4 -o 10.0.2.4
nmap -T4 -sT -O 10.0.2.4
sudo cat /etc/shadow
clear
cat /etc/passwd
clear
history
exit
root@kali:~#
```

Looking at the above command history, we can see that this user has been launching some nmap scans, and has also been reading the `/etc/passwd` and `/etc/shadow` files, which we'll cover later. An important note, as you may have seen in the above screenshots, that some commands are missing. This is because the terminal where the commands were entered needs to be closed before the commands can be written to the history file!

HIDDEN FILES

As covered above, the bash_history file begins with a period "." – and in Linux systems this results in the file being hidden from immediate view. While it's not a fool-proof way of hiding files or directories, it could easily be missed as it's hiding in plain sight. In the below screenshot we can see that this seemingly empty directory actually contains some hidden files, and other directories.



So what's the difference between `ls` and `ls -a`?

- **ls = list directory contents**
- **ls -a = list directory contents and do not ignore entries starting with .**

So make sure to always be using `ls -a` where appropriate to ensure you don't miss any files hidden using this very simple technique.

CLEAR FILES

Let's start off by defining what we mean by "clear files". These are any files that are accessible through standard means, such as the terminal or the graphical browser. This includes areas such as:

- **A user's desktop**
- **A user's default directories, including; Downloads, Music, Pictures, Public, Templates, Videos**
- **The Trash Bin**

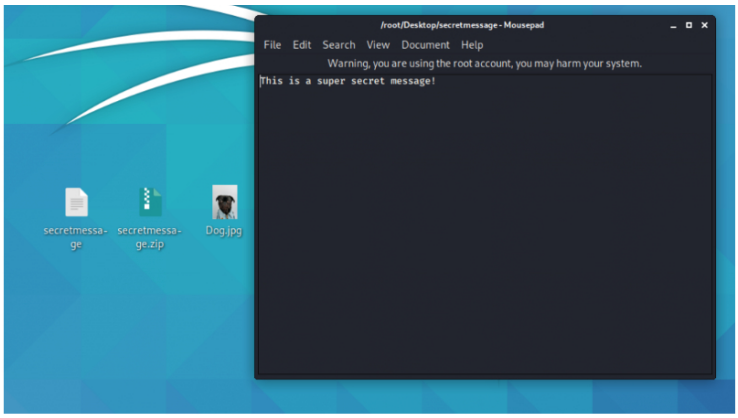
These areas can simply be looked through normally using a terminal or file browser, and may contain useful files if the suspect hasn't bothered to hide them. However, innocent-looking files can contain hidden data, which we will cover below.

STEGANOGRAPHY

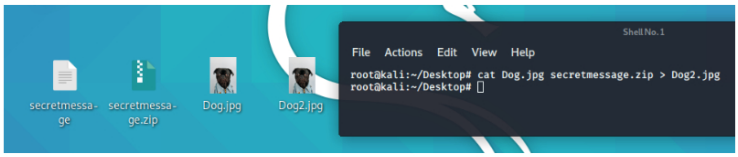
"*The practice of concealing messages or information within other non-secret text or data.*" An example of this would be having a text file that contains secret information, where the text file is actually hidden inside an innocent image file. If this image file was sent as an email attachment, the recipient would receive a normal image file. However, using the right tools, you can recover the hidden file. You can also insert hidden messages in the form of text strings within a file's metadata. We suggest you read the following short article that explains what steganography is, written by TechTarget. Steganography has been described as a counter-forensics technique, as it works to make data harder to find by hiding it in plain sight.

Hiding ZIP Files Inside Images

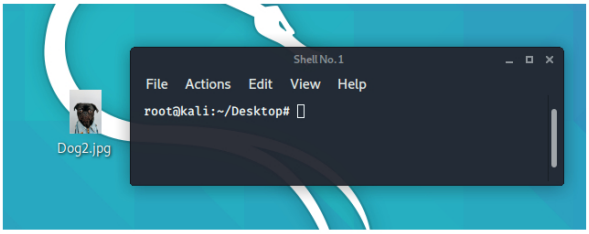
Let's go through an example where we want to hide a ZIP file inside an image file. On our Desktop we have a text file `secretmessage.txt`, the same text file but inside a ZIP container `secretmessage.zip`, and an image of a dog `Dog.jpg`.

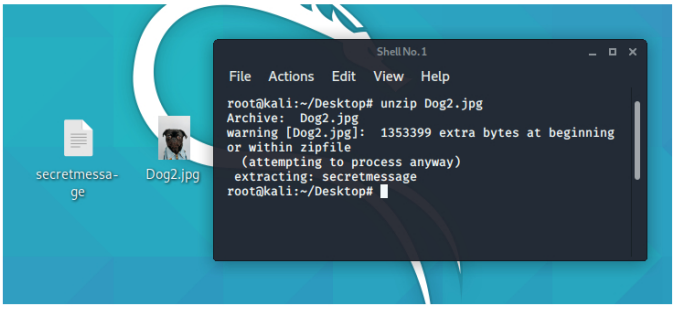


Using the command **`cat Dog.jpg secretmessage.zip > Dog2.jpg`** we can insert the ZIP file into the image of a dog, creating a new file named Dog2.jpg!



Without any technical knowledge, you'd just assume that this image is legitimate, and it can still be opened like a normal image file. But, if we use the **`unzip`** command on the image file, we'll retrieve the files that were inside the hidden ZIP file.



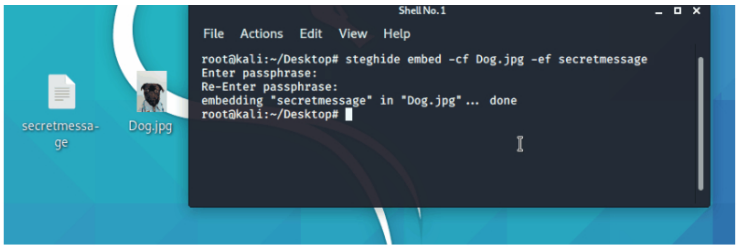


Using Steghide to Hide and Retrieve Files

In a similar way to what we achieved with the **`cat`** command, we can do the same using Steghide, but this time we can password protect the file we're hiding data inside, known as the cover file. We're going to hide secretmessage(.txt) inside Dog.jpg using this tool. The command we want to use is **`steghide embed -cf Dog.jpg -ef secretmessage`**.

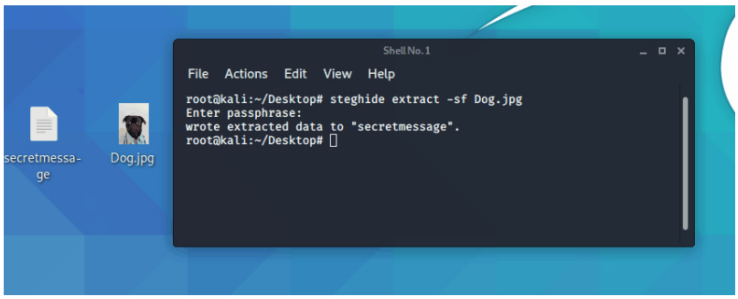
- **steghide** – summons the tool we want to use
- **embed** – selects the operation we want to use, in this case embedding a file in another
- **-cf Dog.jpg** – the 'cover file' flag is where we state the file that will hold the hidden file
- **-ef secretmessage** – the 'embed file' flag is where we state the file we want to hide inside the cover file

Once we've entered this command, we'll be prompted to enter a password, we're going to press [Enter] twice to not use a password.



Great, it worked! Now let's delete secretmessage and try to recover the copy we hid inside Dog.jpg. We want to use the command **`steghide extract -sf Dog.jpg`**.

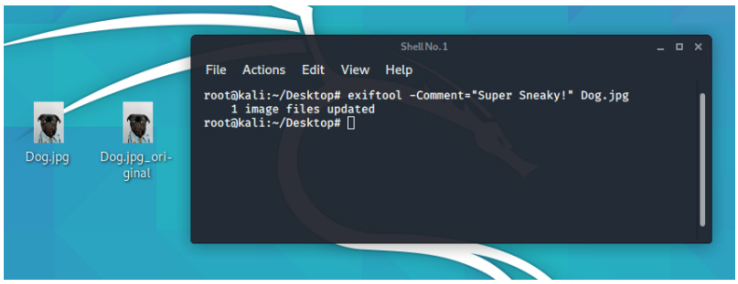
- **steghide** – summons the tool
- **extract** – selects the extract operation
- **-sf Dog.jpg** – the 'steganography file' flag is used to tell steghide which file we believe contains data hidden via steganography, which is our cover file from above.



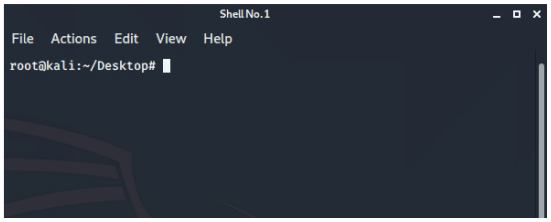
Obviously hiding files in this method with passwords makes it harder for forensic investigators, as they'll need to know or retrieve a valid password to retrieve the file. But tools exist that allow for the brute forcing of passwords to recover hidden files, such as https://github.com/Paradoxis/StegCracker.

Hiding Strings in Metadata

Another way to hide information is to insert text strings into the metadata of a file. We can embed and extract strings using a tool called ExifTool. Run the command **`exiftool`** on your Kali system, if it is not found, use the following command to install it: **`sudo apt-get install exiftool`**. We're going to embed the phrase "Super Sneaky!" into the metadata of Dog.jpg using the command **`exiftool -Comment="Super Sneaky!" Dog.jpg`**. Below you can see that when the command has completed it will generate a new file using the name of the original file, and replacing the original with <filename><.extension<+_original>, in our case the original file is renamed to **`Dog.jpg_original`**.



Now if we use exiftool on the new file, Dog.jpg, we can view all the metadata from this file, including the new comment we just added.



To add a few more seconds to an investigators day, this text string could be encoded in formats such as Base64 or Hexadecimal to make it less immediately obvious.

Previous Topic

Mark Complete

Back to Lesson

Privacy & Cookies Policy

Privacy - Terms