Conestoga College

Course	**INF08965 - Network and Security**
Activity Title	**Cryptography**
Student Name	**Twinkle Akhilesh Mishra**
Student Number	**8894858**
Lab performed on (Date):	**20-Jan-2025**

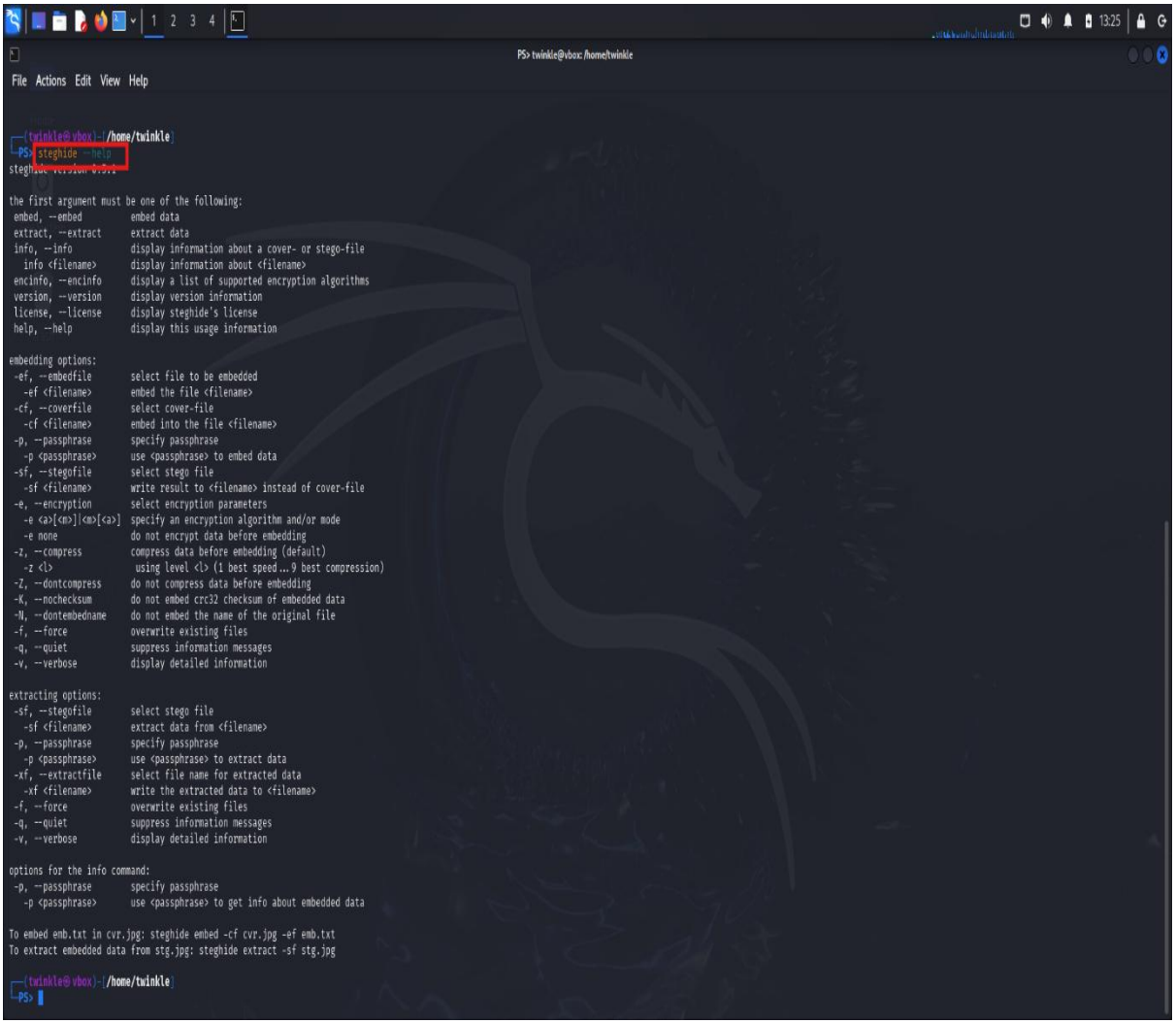
Objectives

- **Understand how Steganography works**
- **Learning about hashing algorithms and password cracking**
- **Learn how to use steghide and hashcat tools**

Output#1: Steganography

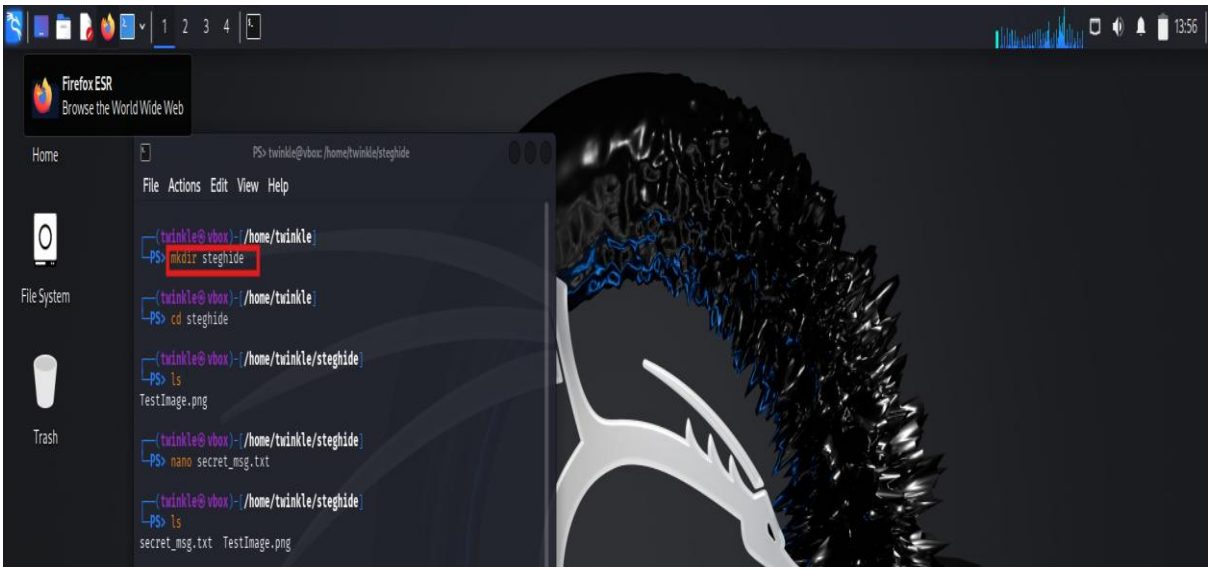
Step 1: Install "**Steghide**" on Kali Linux

- Open your terminal.
- Run the following command to install Steghide:
  sudo apt-get install steghide
- After the installation is complete, verify it by typing:
  steghide –help
- If the installation is successful, you'll see the help options for Steghide.



Step 2: Create a Working Directory.

- To keep your files organized, create a folder named "**steghide**"
  mkdir steghide
- Navigate to the folder: **cd steghide**



Step 3: Download a photo/image.

- Launch Firefox (or similar browser) on Kali Linux.
- Download the desired image from the web and place it to the **steghide folder**.
- Rename the image as **TestImage.jpg**.
- The command **ls** will confirm that the file exists in your directory.
- You should see the file **TestImage.jpg**, which is described below.

Step 4: Create a secret_msg text file.

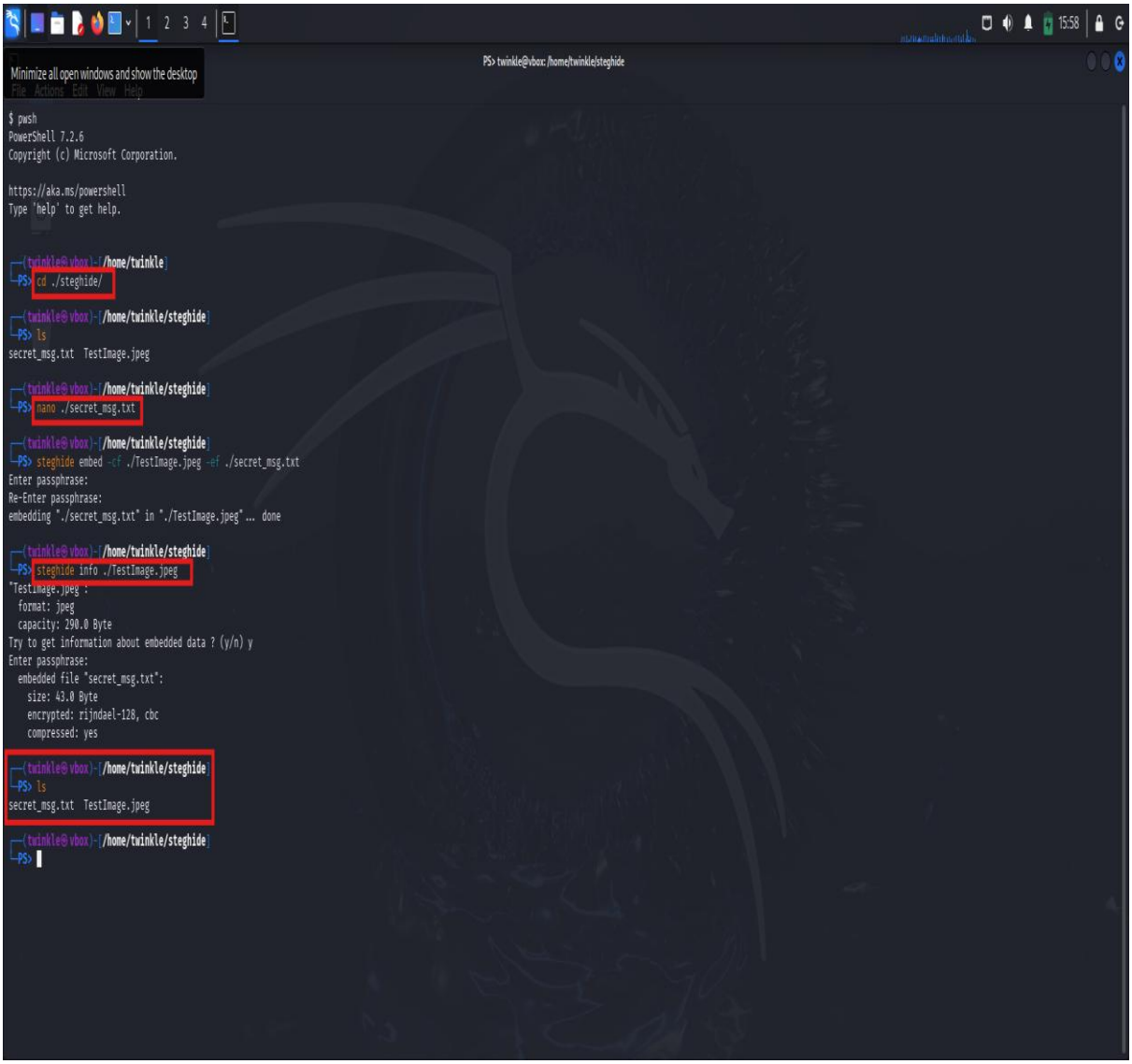
- To create a file called **secret_msg.txt**, use the Nano text editor as follows:
  nano secret_msg.txt
- In the editor, type your name and a secret message.
- Save the file by using Ctrl+O, then Enter.
- To exit the editor, press Ctrl+X.

Step 5: Embed the secret msg in the image

- Use Steghide to embed the text file (secret_msg.txt) into the image (TestImage.jpg):
  steghide embed -cf TestImage.jpg -ef secret_msg.txt
- You'll be prompted to set a passphrase. Enter a passphrase and include the hidden message within the image.

Step 6: Validate the embedded content.

- Check the image to ensure that the secret message is set up.
  steghide info TestImage.jpg
- The output should show that additional information is contained in the image.



Step 7: Retrieve the secret message

- To add security, delete the original secret_msg.txt file.
  rm secret_msg.txt
- To get the hidden message from the image, use:
  steghide extract –sf TestImage.jpg

Step 8: Verify the extracted file

- The **ls -l** command will display the details of the files in the directory:
  In addition to providing metadata about the file, this verifies that secret_msg.txt exists.

Step 9: Read the Extracted Secret Message

- To read the contents of the secret_msg.txt file, use the cat command:
  cat secret_msg.txt

The terminal will display the contents of the file, which should include name and the secret code as shown in below screenshot.

Output#2 : Answer the following questions and write your response in Output # 2.

1. Which encryption method is used in this example? (1 marks)
   - In this case, **Rijndael-128**, **CBC encryption** method is used.
   - Rijndael-128 is the name of the standard for the AES algorithm with a key length of **128 bits**.
   - CBC (Cipher Block Chaining) is a variant of the encryption method that guarantees the connection between the blocks of data using an initialization vector (IV) for strong encryption purpose.

2. Observe your original image and compare it with the one that has embedded hidden message. Can you spot any difference? (1 marks)
   - There is no way of knowing if the image with the embedded message is different from the original image because with steganography, information is captured inside the least significant bits (LSBs) of the image. This is so minute that it goes unnoticed because it does not alter the image in any way.

3. Perform online research to find out how safe steganography is. Can it be easily detected. (2 marks)
   - Steganography, when used together with strong encryption, can be fairly safe since it conceals a message's existence.
   - This is done by inserting information in the least significant bits (LSB) of the file, which is undetectable to a normal person. Nevertheless, it does have some shortcomings.
   - Detecting tools such as **StegExpose** and other steganalytical methods evaluate the alteration in average pixel value, file size, and other statistical changes.
   - These tools can uncover the secret data. In addition, the data can be destroyed or altered with the use of lossy compressions or resizing images.
   - Although it is a great tool when compared to spy tools, the overall dependability of steganography is rooted on its effective application and powerful encryption.

Output#3 :

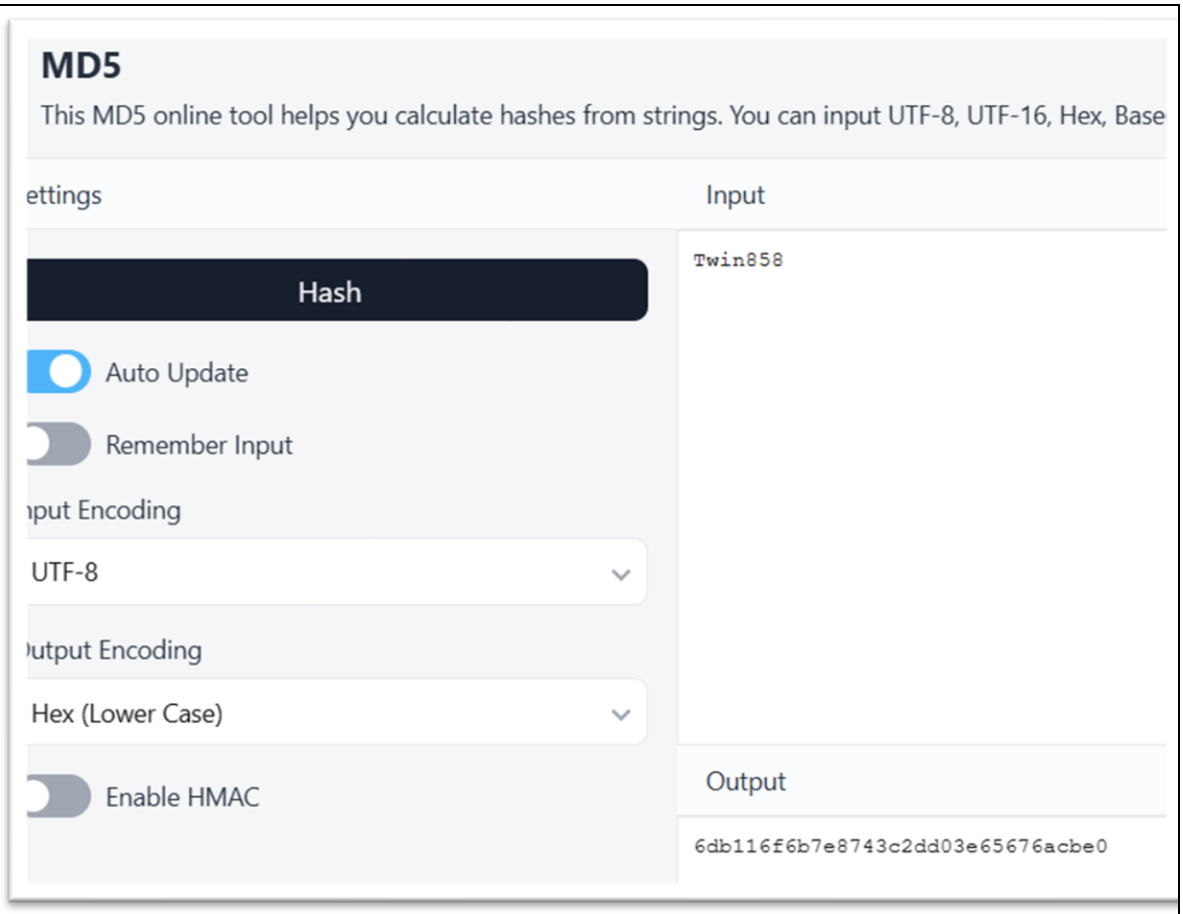
(B) Hashcat on Kali Linux

Step 1: **Password Hashing**

- Used MD5 hash generator to create an MD5 hash of **Twin858**.
- The resulting hash was **6db116f6b7e8743c2dd03e65676acbe0**.

Step 2: Save the Hash on a File:

- Stored the hash into a file called **hashtest** with this command:
  echo "6db116f6b7e8743c2dd03e65676acbe0" > hashtest



Step 3: Creating Hashcat Command

- Defined custom characters sets:
- -1 for Uppercase Letters: ?u
- -2 for Mixed and digitis: ?l?u?d
- -3 for Digits: ?d
- It was set with the pattern ?1?2?2?2?3?3?3 to the password structure.
- Then ran the hashcat command:
  hashcat -m 0 -a 3 -1 ?u -2 ?l?u?d -3 ?d hashtest ?1?2?2?2?3?3?3 --force

Step 4: Password Cracking

- As was seen on the output, the password was cracked by hashcat in a matter of mere seconds, showing Twin858 as the correct password used.
- **Confirmation of Results:**
  cat ~/.hashcat/hashcat.potfile


PS> kali@kali: /home/kali
File Actions Edit View Help
* Single-Hash
* Single-Salt
* Brute-Force
* Raw-Hash
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 0 MB
6db116f6b7e8743c2dd03e65676acbe0:Twin858
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
Hash.Target.....: 6db116f6b7e8743c2dd03e65676acbe0
Time.Started.....: Tue Jan 21 11:24:54 2025, (4 secs)
Time.Estimated ...: Tue Jan 21 11:24:58 2025, (0 secs)
Kernel.Feature ...: Pure Kernel
Guess.Mask.....: ?1?2?2?3?3?3 [7]
Guess.Charset....: -1 ?u, -2 ?l, -3 ?d, -4 Undefined
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 73900.2 kH/s (11.42ms) @ Accel:256 Loops:1024 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 325009408/456976000 (71.12%)
Rejected.....: 0/325009408 (0.00%)
Restore.Point....: 18432/26000 (70.89%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1024 Iteration:0-1024
Candidate.Engine.: Device Generator
Candidates.#1....: Marx775 → Tkas603
Hardware.Mon.#1..: Util: 91%
Started: Tue Jan 21 11:24:52 2025
Stopped: Tue Jan 21 11:25:00 2025