

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

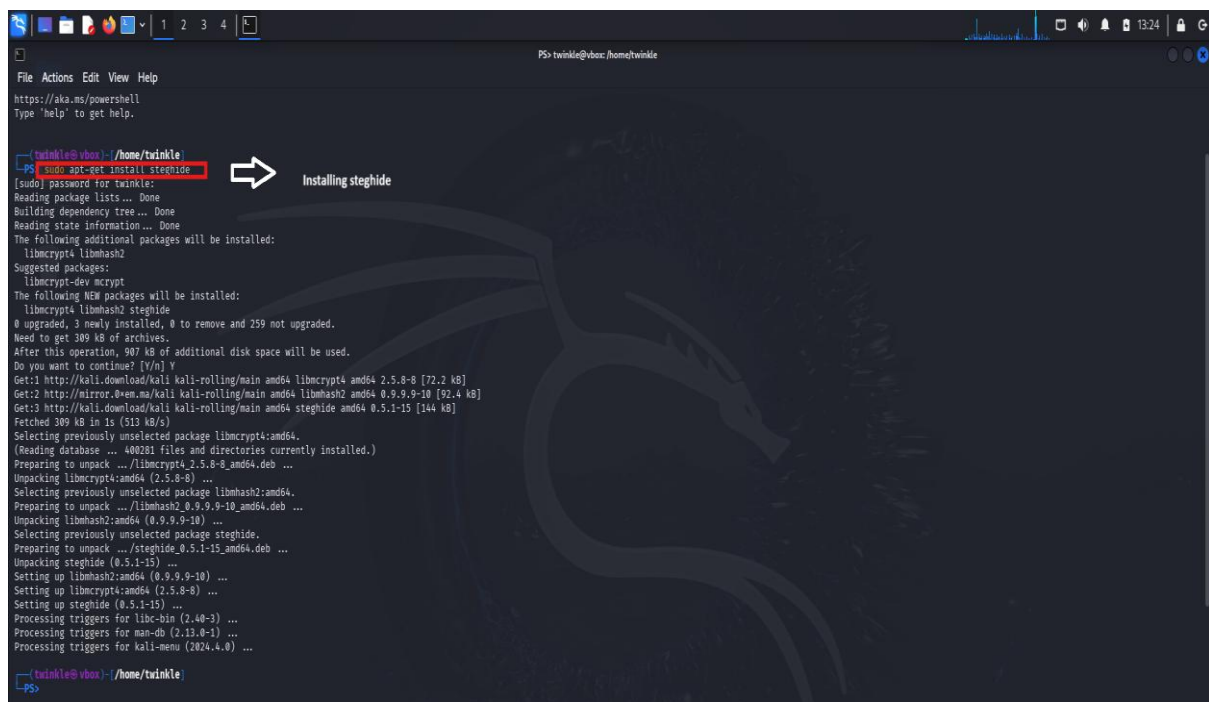
Objectives

- 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.



```
twinkle@vbox: /home/twinkle
PS> twinkle@vbox: /home/twinkle
PS> sudo apt-get install steghide
[sudo] password for twinkle:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libmcrypt4 libmhash2
Suggested packages:
  libmcrypt-dev mrcrypt
The following NEW packages will be installed:
  libmcrypt4 libmhash2 steghide
0 upgraded, 3 newly installed, 0 to remove and 259 not upgraded.
Need to get 309 kB of archives.
After this operation, 907 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://kali.download/kali kali-rolling/main amd64 libmcrypt4 amd64 2.5.8-8 [72.2 kB]
Get:2 http://mirror.0xen.ma/kali kali-rolling/main amd64 libmhash2 amd64 0.9.9-10 [92.4 kB]
Get:3 http://kali.download/kali kali-rolling/main amd64 steghide amd64 0.5.1-15 [144 kB]
Fetched 309 kB in 1s (513 kB/s)
Selecting previously unselected package libmcrypt4:amd64.
(Reading database ... 480281 files and directories currently installed.)
Preparing to unpack .../libmcrypt4_2.5.8-8_amd64.deb ...
Unpacking libmcrypt4:amd64 (2.5.8-8) ...
Selecting previously unselected package libmhash2:amd64.
Preparing to unpack .../libmhash2_0.9.9-10_amd64.deb ...
Unpacking libmhash2:amd64 (0.9.9-10) ...
Selecting previously unselected package steghide.
Preparing to unpack .../steghide_0.5.1-15_amd64.deb ...
Unpacking steghide (0.5.1-15) ...
Setting up libmhash2:amd64 (0.9.9-10) ...
Setting up libmcrypt4:amd64 (2.5.8-8) ...
Setting up steghide (0.5.1-15) ...
Processing triggers for libc-bin (2.40-3) ...
Processing triggers for man-db (2.13.0-1) ...
Processing triggers for kali-menu (2024.4.0) ...
twinkle@vbox: /home/twinkle
PS>
```

```
PS> twinkle@vbox:/home/twinkle
File Actions Edit View Help

twinkle@vbox:~/home/twinkle
PS> steghide --help
steghide version 0.5.1

the first argument must be one of the following:
embed, --embed          embed data
extract, --extract      extract data
info, --info            display information about a cover- or stego-file
info, --info <filename> display information about <filename>
encinfo, --encinfo      display a list of supported encryption algorithms
version, --version       display version information
license, --license       display steghide's license
help, --help             display this usage information

embedding options:
-e, --embedfile          select file to be embedded
-e, --embedfile <filename> embed the file <filename>
-cf, --coverfile         select cover-file
-cf, --coverfile <filename> embed into the file <filename>
-p, --passphrase         specify passphrase
-p, --passphrase <passphrase> use <passphrase> to embed data
-sf, --stegofile         select stego file
-sf, --stegofile <filename> write result to <filename> instead of cover-file
-e, --encryption         select encryption parameters
-e <ca>[<m>][<md>][<ca>] specify an encryption algorithm and/or mode
-e none                  do not encrypt data before embedding
-z, --compress           compress data before embedding (default)
-z <level>               using level <level> (1 best speed...9 best compression)
-Z, --dontcompress       do not compress data before embedding
-K, --nochecksum         do not embed crc32 checksum of embedded data
-N, --dontembedname      do not embed the name of the original file
-f, --force              overwrite existing files
-q, --quiet              suppress information messages
-v, --verbose            display detailed information

extracting options:
-sf, --stegofile         select stego file
-sf, --stegofile <filename> extract data from <filename>
-p, --passphrase         specify passphrase
-p, --passphrase <passphrase> use <passphrase> to extract data
-xf, --extractfile       select file name for extracted data
-xf, --extractfile <filename> write the extracted data to <filename>
-f, --force              overwrite existing files
-q, --quiet              suppress information messages
-v, --verbose            display detailed information

options for the info command:
-p, --passphrase         specify passphrase
-p, --passphrase <passphrase> use <passphrase> to get info about embedded data

To embed emb.txt in cvr.jpg: steghide embed -cf cvr.jpg -ef emb.txt
To extract embedded data from stg.jpg: steghide extract -sf stg.jpg

twinkle@vbox:~/home/twinkle
PS>
```

Step 2: Create a Working Directory.

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

```
Firefox ESR
Browse the WorldWide Web

Home
File System
Trash

PS> twinkle@vbox:/home/twinkle/steghide
File Actions Edit View Help

twinkle@vbox:~/home/twinkle
PS> mkdir steghide

twinkle@vbox:~/home/twinkle
PS> cd steghide

twinkle@vbox:~/home/twinkle/steghide
PS> ls
TestImage.png

twinkle@vbox:~/home/twinkle/steghide
PS> nano secret_msg.txt

twinkle@vbox:~/home/twinkle/steghide
PS> ls
secret_msg.txt TestImage.png
```

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.

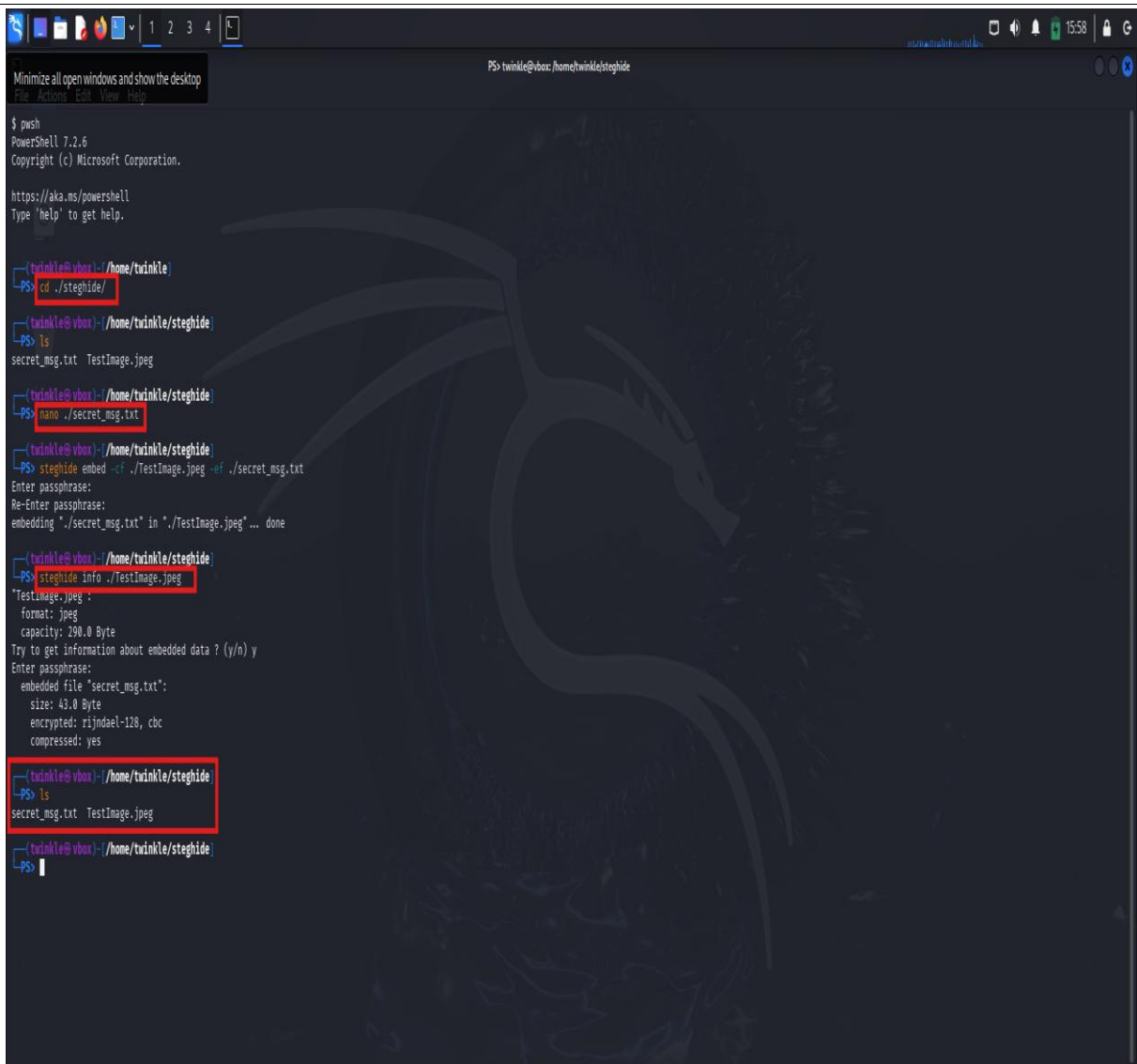
- To create a file called **secret_msg.txt**, use the Nano text editor as follows:
nano secret_msg.txt
- In the editor, type your name and a secret message.
- Save the file by using Ctrl+O, then Enter.
- To exit the editor, press Ctrl+X.

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.



```
PS> twinkle@vbox:/home/twinkle/steghide

$ push
PowerShell 7.2.6
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

twinkle@vbox: ~/home/twinkle
PS> cd ./steghide/
twinkle@vbox: ~/home/twinkle/steghide
PS> ls
secret_msg.txt TestImage.jpeg
twinkle@vbox: ~/home/twinkle/steghide
PS> nano ./secret_msg.txt
twinkle@vbox: ~/home/twinkle/steghide
PS> steghide embed -sf ./TestImage.jpeg -ef ./secret_msg.txt
Enter passphrase:
Re-Enter passphrase:
embedding "./secret_msg.txt" in "./TestImage.jpeg"... done
twinkle@vbox: ~/home/twinkle/steghide
PS> steghide info ./TestImage.jpeg
TestImage.jpeg :
format: jpeg
capacity: 290.0 Byte
Try to get information about embedded data ? (y/n) y
Enter passphrase:
embedded file "secret_msg.txt":
size: 43.0 Byte
encrypted: rijndael-128, cbc
compressed: yes
twinkle@vbox: ~/home/twinkle/steghide
PS> ls
secret_msg.txt TestImage.jpeg
twinkle@vbox: ~/home/twinkle/steghide
PS>
```

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.

The screenshot shows a terminal window with the following commands and outputs:

- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> ls`
secret_msg.txt TestImage.jpeg
List of files inside steghide folder
- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> nano ./secret_msg.txt`
Embedding the file secret_msg.txt into TestImage.jpeg
- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> steghide embed -cf ./TestImage.jpeg -ef ./secret_msg.txt`
Enter passphrase:
Re-Enter passphrase:
embedding "./secret_msg.txt" in "./TestImage.jpeg" ... done
Embedding the file secret_msg.txt into TestImage.jpeg
- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> steghide info ./TestImage.jpeg`
"TestImage.jpeg":
format: jpeg
capacity: 290.0 Byte
Try to get information about embedded data ? (y/n) y
Enter passphrase:
embedded file "secret_msg.txt":
size: 43.0 Byte
encrypted: rijndael-128, cbc
compressed: yes
Retrieving the information about the image TestImage.jpeg
- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> ls`
secret_msg.txt TestImage.jpeg
- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> rm ./secret_msg.txt`
Removing secret_msg.txt file
- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> steghide extract -sf ./TestImage.jpeg`
Enter passphrase:
wrote extracted data to "secret_msg.txt".
extracting the hidden file i.e secret_msg.txt from the image TestImage.jpeg
- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> ls -l`
total 12
-rw-rw-r-- 1 twinkle twinkle 43 Jan 26 15:59 secret_msg.txt
-rw-rw-r-- 1 twinkle twinkle 5826 Jan 26 15:57 TestImage.jpeg
- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> cat ./secret_msg.txt`
Name: Twinkle Mishra
Code: BHOuo\$8bbjgf7
Displaying the content inside secret_msg.txt
- `twinkle@vbox: ~/home/twinkle/steghide`
`PS> clear`

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
```

MD5

This MD5 online tool helps you calculate hashes from strings. You can input UTF-8, UTF-16, Hex, Base

Settings	Input
<div>Hash</div> <div><input checked="" type="checkbox"/> Auto Update</div> <div><input type="checkbox"/> Remember Input</div> <div>Input Encoding</div> <div>UTF-8</div> <div>Output Encoding</div> <div>Hex (Lower Case)</div> <div><input type="checkbox"/> Enable HMAC</div>	<div>Twin858</div> <div>Output</div> <div>6db116f6b7e8743c2dd03e65676acbe0</div>

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?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

```