Lab Program 2: Acquisition of data: to learn the methods to acquire/gain access to data using virus/malware file.

General Steps

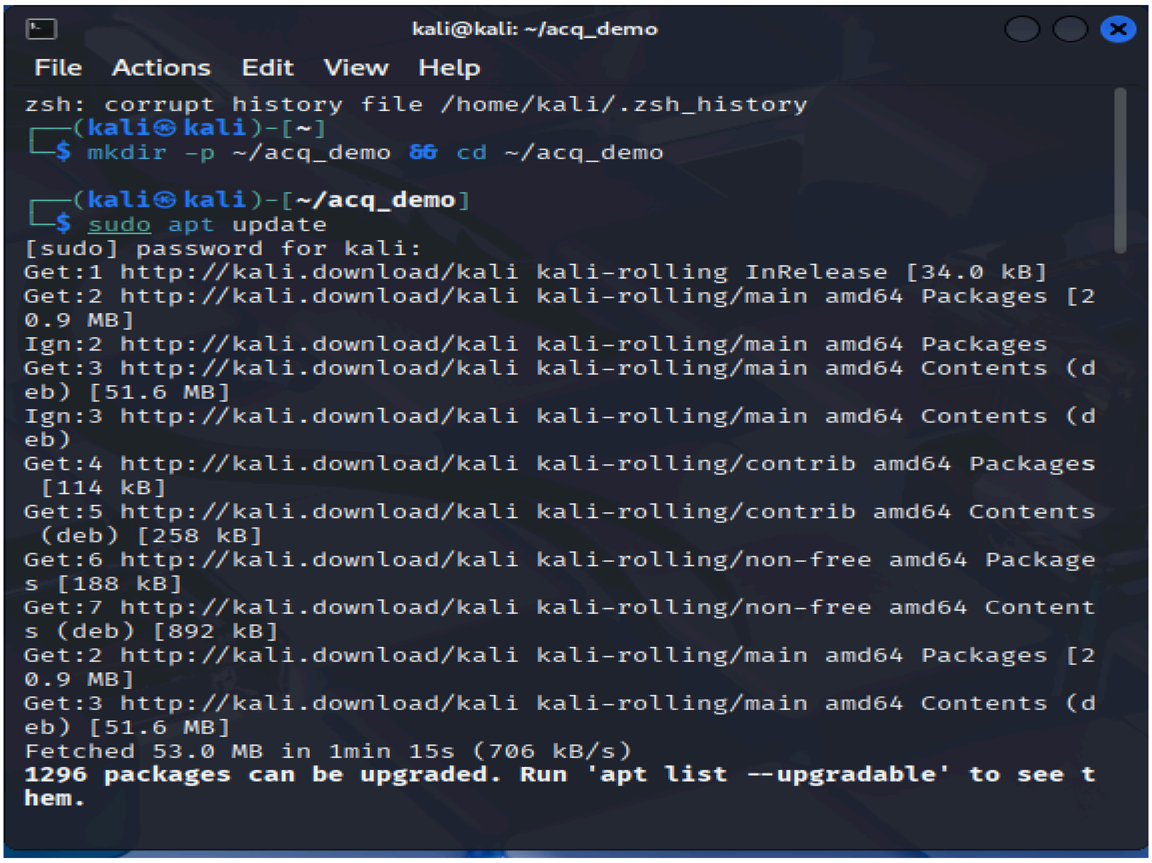
1. **Prepare a workspace and install tools (`steghide`, `zip`, `clamav`).**
2. **Create benign "victim" files (sample `secret.txt` and the EICAR test string) and verify them.**
3. **Simulate a malware run: run a bash script that copies the secret, zips it with a password, and stages it for exfiltration.**
4. **Hide the encrypted payload inside an image using `steghide`.**
5. **Conceal the stego-image (e.g., move/rename to a hidden location), then extract and decrypt the payload to recover the secret.**

1) Prepare workspace & install tools

mkdir -p ~/acq_demo && cd ~/acq_demo

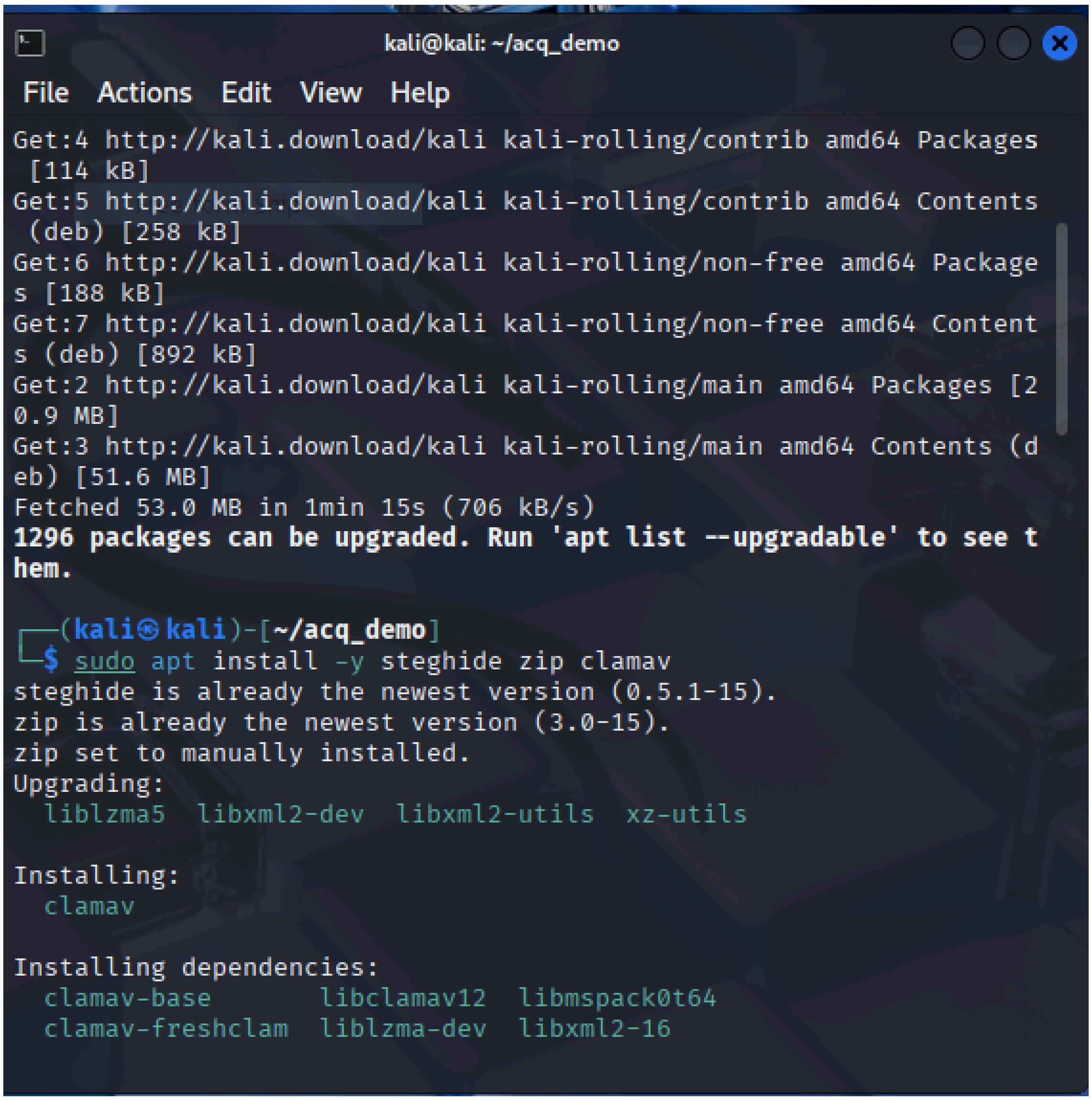
sudo apt update

sudo apt install -y steghide zip clamav



```
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~]
└─$ mkdir -p ~/acq_demo && cd ~/acq_demo

┌──(kali㉿kali)-[~/acq_demo]
└─$ sudo apt update
[sudo] password for kali:
Get:1 http://kali.download/kali kali-rolling InRelease [34.0 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [2
0.9 MB]
Ign:2 http://kali.download/kali kali-rolling/main amd64 Packages
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (d
eb) [51.6 MB]
Ign:3 http://kali.download/kali kali-rolling/main amd64 Contents (d
eb)
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages
 [114 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents
 (deb) [258 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Package
s [188 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Content
s (deb) [892 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [2
0.9 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (d
eb) [51.6 MB]
Fetched 53.0 MB in 1min 15s (706 kB/s)
1296 packages can be upgraded. Run 'apt list --upgradable' to see t
hem.
```



```
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages
 [114 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents
 (deb) [258 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Package
s [188 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Content
s (deb) [892 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [2
0.9 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (d
eb) [51.6 MB]
Fetched 53.0 MB in 1min 15s (706 kB/s)
1296 packages can be upgraded. Run 'apt list --upgradable' to see t
hem.

┌──(kali㉿kali)-[~/acq_demo]
└─$ sudo apt install -y steghide zip clamav
steghide is already the newest version (0.5.1-15).
zip is already the newest version (3.0-15).
zip set to manually installed.
Upgrading:
  liblzma5  libxml2-dev  libxml2-utils  xz-utils

Installing:
  clamav

Installing dependencies:
  clamav-base       libclamav12  libmspack0t64
  clamav-freshclam  liblzma-dev  libxml2-16
```

2) Create benign victim files and verify workspace

mkdir -p ~/acq_demo/victim_docs

echo "this is a test secret." > ~/acq_demo/victim_docs/secret.txt

# harmless AV test string (EICAR)

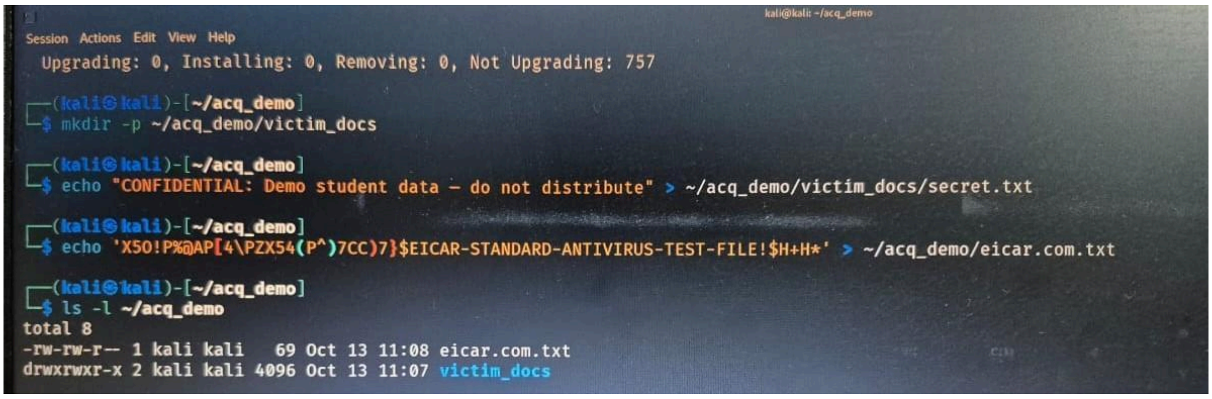
echo 'X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*' > ~/acq_demo/eicar.com.txt

# verify workspace and view the secret

ls -l ~/acq_demo

cat ~/acq_demo/victim_docs/secret.txt

Output: this is a test secret



```
 Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 757

┌──(kali㉿kali)-[~/acq_demo]
└─$ mkdir -p ~/acq_demo/victim_docs

┌──(kali㉿kali)-[~/acq_demo]
└─$ echo "CONFIDENTIAL: Demo student data – do not distribute" > ~/acq_demo/victim_docs/secret.txt

┌──(kali㉿kali)-[~/acq_demo]
└─$ echo 'X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*' > ~/acq_demo/eicar.com.txt

┌──(kali㉿kali)-[~/acq_demo]
└─$ ls -l ~/acq_demo
total 8
-rw-rw-r-- 1 kali kali   69 Oct 13 11:08 eicar.com.txt
drwxrwxr-x 2 kali kali 4096 Oct 13 11:07 victim_docs
```

3) Simulate "infection" — save the malware script and run it (debug run included)

Save this exact content to **~/acq_demo/simulate_malware.sh** (file only — do **not** include the run commands inside the file):

nano simulate_malware.sh

```
#!/bin/bash

# simulate_malware (robust, debug-friendly) — local-only demo

set -euo pipefail

echo "[*] script start: $(date)"

TMPDIR="$(mktemp -d /tmp/exfil.XXXXXX)" || { echo "[!] mktemp failed" >&2; exit 1; }

echo "[*] TMPDIR=${TMPDIR}"

if [ -z "${TMPDIR}" ] || [ ! -d "${TMPDIR}" ]; then

  echo "[!] Failed to create temp dir. TMPDIR='${TMPDIR}'" >&2

  exit 1

fi

trap 'echo "[*] cleaning ${TMPDIR}"; rm -rf "${TMPDIR}"' EXIT

SRC="${HOME}/acq_demo/victim_docs/secret.txt"

if [ ! -f "${SRC}" ]; then

  echo "[!] Source file missing: ${SRC}" >&2

  exit 1

fi

echo "[*] Found source: ${SRC}"

cp "${SRC}" "${TMPDIR}/secret.txt"
```

echo "[*] copied secret to ${TMPDIR}/secret.txt"

# create encrypted zip (password = pass123)

zip -j -P pass123 "${TMPDIR}/payload.zip" "${TMPDIR}/secret.txt"

echo "[*] created zip: ${TMPDIR}/payload.zip"

copy payload out to stable demo location

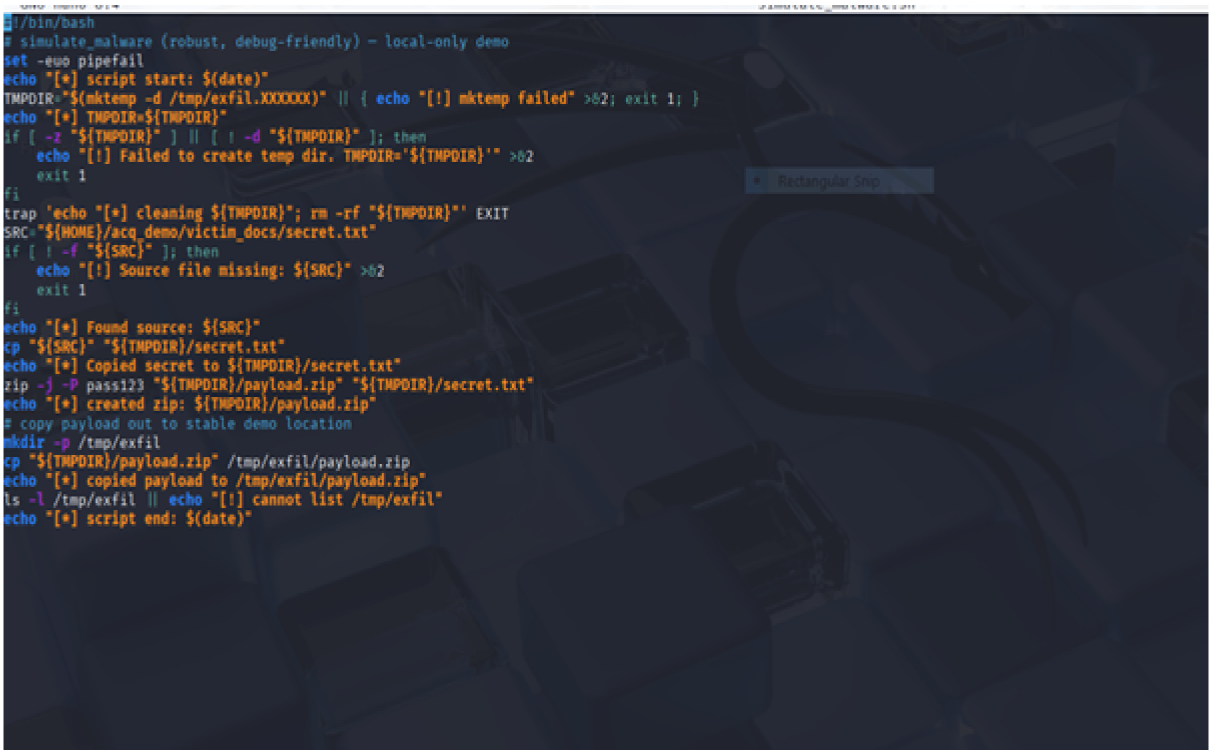
mkdir -p /tmp/exfil

cp "${TMPDIR}/payload.zip" /tmp/exfil/payload.zip

echo "[*] copied payload to /tmp/exfil/payload.zip"

ls -l /tmp/exfil || echo "[!] cannot list /tmp/exfil"

echo "[*] script end: $(date)"



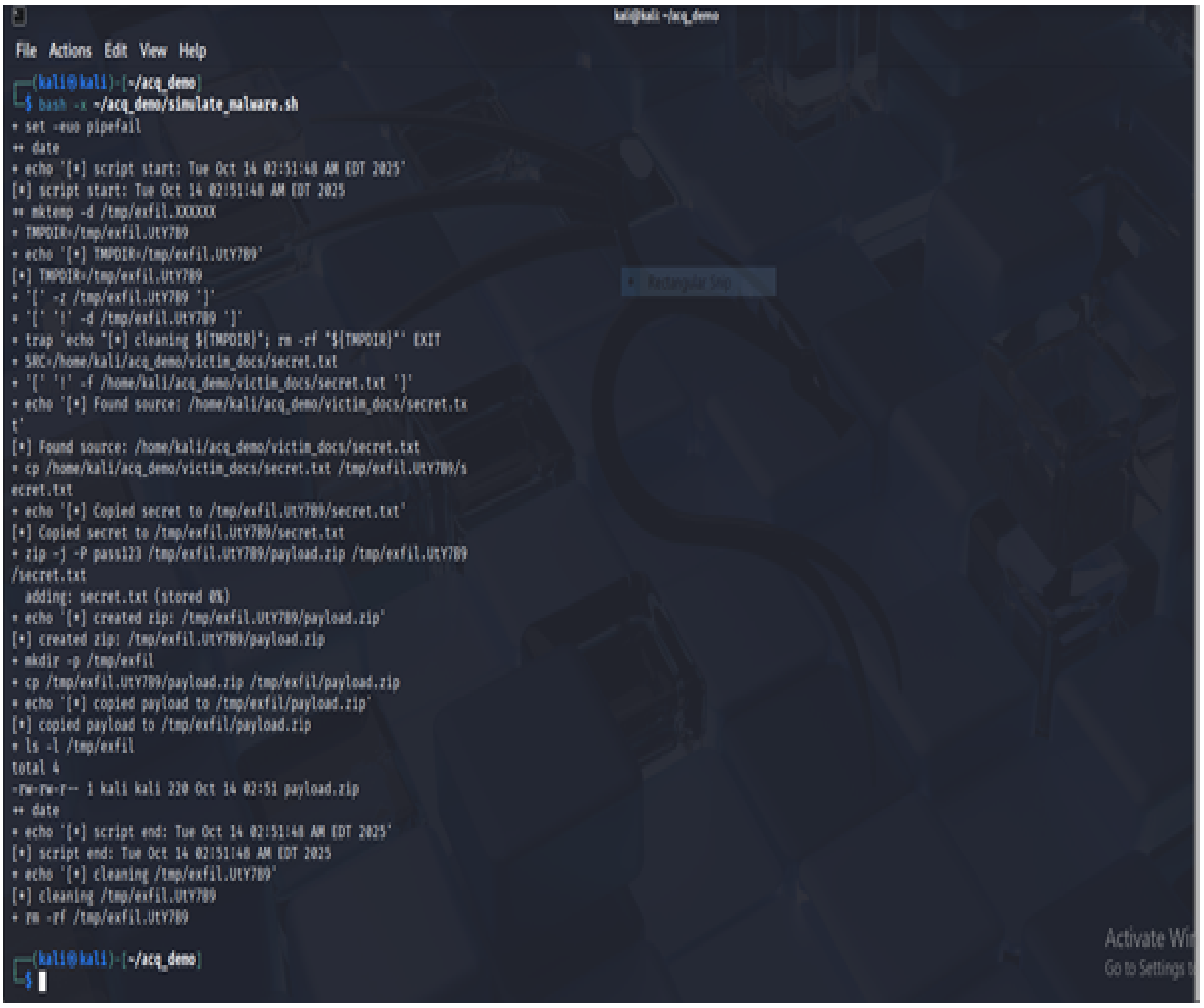
```
#!/bin/bash
# simulate_malware (robust, debug-friendly) - local-only demo
set -euo pipefail
echo "[*] script start: $(date)"
TMPDIR="$(mktemp -d /tmp/exfil.XXXXXX)" || { echo "[!] mktemp failed" >&2; exit 1; }
echo "[*] TMPDIR=${TMPDIR}"
if [ -z "${TMPDIR}" ] || [ ! -d "${TMPDIR}" ]; then
    echo "[!] Failed to create temp dir. TMPDIR='${TMPDIR}'" >&2
    exit 1
fi
trap 'echo "[*] cleaning ${TMPDIR}"; rm -rf "${TMPDIR}"' EXIT
SRC="${HOME}/acq_demo/victim_docs/secret.txt"
if [ ! -f "${SRC}" ]; then
    echo "[!] Source file missing: ${SRC}" >&2
    exit 1
fi
echo "[*] Found source: ${SRC}"
cp "${SRC}" "${TMPDIR}/secret.txt"
echo "[*] Copied secret to ${TMPDIR}/secret.txt"
zip -j -P pass123 "${TMPDIR}/payload.zip" "${TMPDIR}/secret.txt"
echo "[*] created zip: ${TMPDIR}/payload.zip"
# copy payload out to stable demo location
mkdir -p /tmp/exfil
cp "${TMPDIR}/payload.zip" /tmp/exfil/payload.zip
echo "[*] copied payload to /tmp/exfil/payload.zip"
ls -l /tmp/exfil || echo "[!] cannot list /tmp/exfil"
echo "[*] script end: $(date)"
```

Give executable permission

chmod +x ~/acq_demo/simulate_malware.sh

debug run (traces each executed line):

bash -x ~/acq_demo/simulate_malware.sh



```
┌──(kali㉿kali)-[~/acq_demo]
└─$ bash -x ~/acq_demo/simulate_malware.sh
+ set -euo pipefail
++ date
+ echo '[*] script start: Tue Oct 14 02:51:48 AM EDT 2025'
[*] script start: Tue Oct 14 02:51:48 AM EDT 2025
++ mktemp -d /tmp/exfil.XXXXXX
+ TMPDIR=/tmp/exfil.UtY789
+ echo '[*] TMPDIR=/tmp/exfil.UtY789'
[*] TMPDIR=/tmp/exfil.UtY789
+ '[' -z /tmp/exfil.UtY789 ']'
+ '[' '!' -d /tmp/exfil.UtY789 ']'
+ trap 'echo "[*] cleaning ${TMPDIR}"; rm -rf "${TMPDIR}"' EXIT
+ SRC=/home/kali/acq_demo/victim_docs/secret.txt
+ '[' '!' -f /home/kali/acq_demo/victim_docs/secret.txt ']'
+ echo '[*] Found source: /home/kali/acq_demo/victim_docs/secret.tx
t'
[*] Found source: /home/kali/acq_demo/victim_docs/secret.txt
+ cp /home/kali/acq_demo/victim_docs/secret.txt /tmp/exfil.UtY789/s
ecret.txt
+ echo '[*] Copied secret to /tmp/exfil.UtY789/secret.txt'
[*] Copied secret to /tmp/exfil.UtY789/secret.txt
+ zip -j -P pass123 /tmp/exfil.UtY789/payload.zip /tmp/exfil.UtY789
/secret.txt
  adding: secret.txt (stored 0%)
+ echo '[*] created zip: /tmp/exfil.UtY789/payload.zip'
[*] created zip: /tmp/exfil.UtY789/payload.zip
+ mkdir -p /tmp/exfil
+ cp /tmp/exfil.UtY789/payload.zip /tmp/exfil/payload.zip
+ echo '[*] copied payload to /tmp/exfil/payload.zip'
[*] copied payload to /tmp/exfil/payload.zip
+ ls -l /tmp/exfil
total 4
-rw-rw-r-- 1 kali kali 220 Oct 14 02:51 payload.zip
++ date
+ echo '[*] script end: Tue Oct 14 02:51:48 AM EDT 2025'
[*] script end: Tue Oct 14 02:51:48 AM EDT 2025
+ echo '[*] cleaning /tmp/exfil.UtY789'
[*] cleaning /tmp/exfil.UtY789
+ rm -rf /tmp/exfil.UtY789

┌──(kali㉿kali)-[~/acq_demo]
└─$
```

4) Hide the encrypted payload in an image (steganography) — commands only

Place a JPEG at ~/acq_demo/cover1.jpg before running these commands (or copy a system image into place):

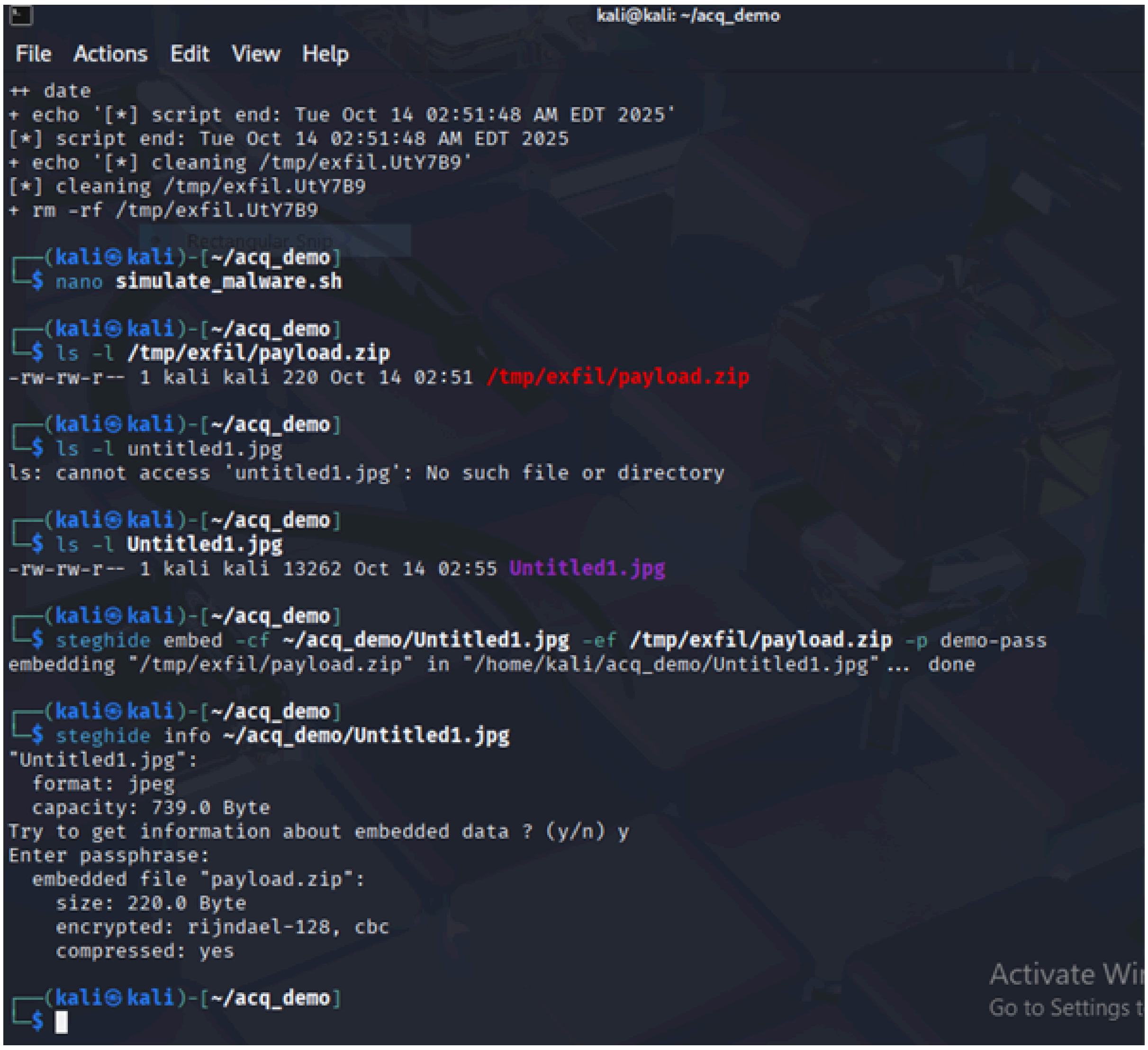
# verify files

```
ls -l /tmp/exfil/payload.zip
ls -l ~/acq_demo/cover1.jpg
```

# embed and inspect

```
steghide embed -cf ~/acq_demo/cover1.jpg -ef /tmp/exfil/payload.zip -p demo-pass
steghide info ~/acq_demo/cover1.jpg
```

type y and give pass phrase demo-pass



```
++ date
+ echo '[*] script end: Tue Oct 14 02:51:48 AM EDT 2025'
[*] script end: Tue Oct 14 02:51:48 AM EDT 2025
+ echo '[*] cleaning /tmp/exfil.UtY7B9'
[*] cleaning /tmp/exfil.UtY7B9
+ rm -rf /tmp/exfil.UtY7B9

┌──(kali㉿kali)-[~/acq_demo]
└─$ nano simulate_malware.sh

┌──(kali㉿kali)-[~/acq_demo]
└─$ ls -l /tmp/exfil/payload.zip
-rw-rw-r-- 1 kali kali 220 Oct 14 02:51 /tmp/exfil/payload.zip

┌──(kali㉿kali)-[~/acq_demo]
└─$ ls -l untitled1.jpg
ls: cannot access 'untitled1.jpg': No such file or directory

┌──(kali㉿kali)-[~/acq_demo]
└─$ ls -l Untitled1.jpg
-rw-rw-r-- 1 kali kali 13262 Oct 14 02:55 Untitled1.jpg

┌──(kali㉿kali)-[~/acq_demo]
└─$ steghide embed -cf ~/acq_demo/Untitled1.jpg -ef /tmp/exfil/payload.zip -p demo-pass
embedding "/tmp/exfil/payload.zip" in "/home/kali/acq_demo/Untitled1.jpg"... done

┌──(kali㉿kali)-[~/acq_demo]
└─$ steghide info ~/acq_demo/Untitled1.jpg
"Untitled1.jpg":
  format: jpeg
  capacity: 739.0 Byte
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "payload.zip":
    size: 220.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes

┌──(kali㉿kali)-[~/acq_demo]
└─$ █
```

5) Conceal stego-image and demonstrate extraction (local)

mkdir -p ~/.hidden_store

mv ~/acq_demo/cover1.jpg ~/.hidden_store/.cover1.jpg

ls -la ~/.hidden_store

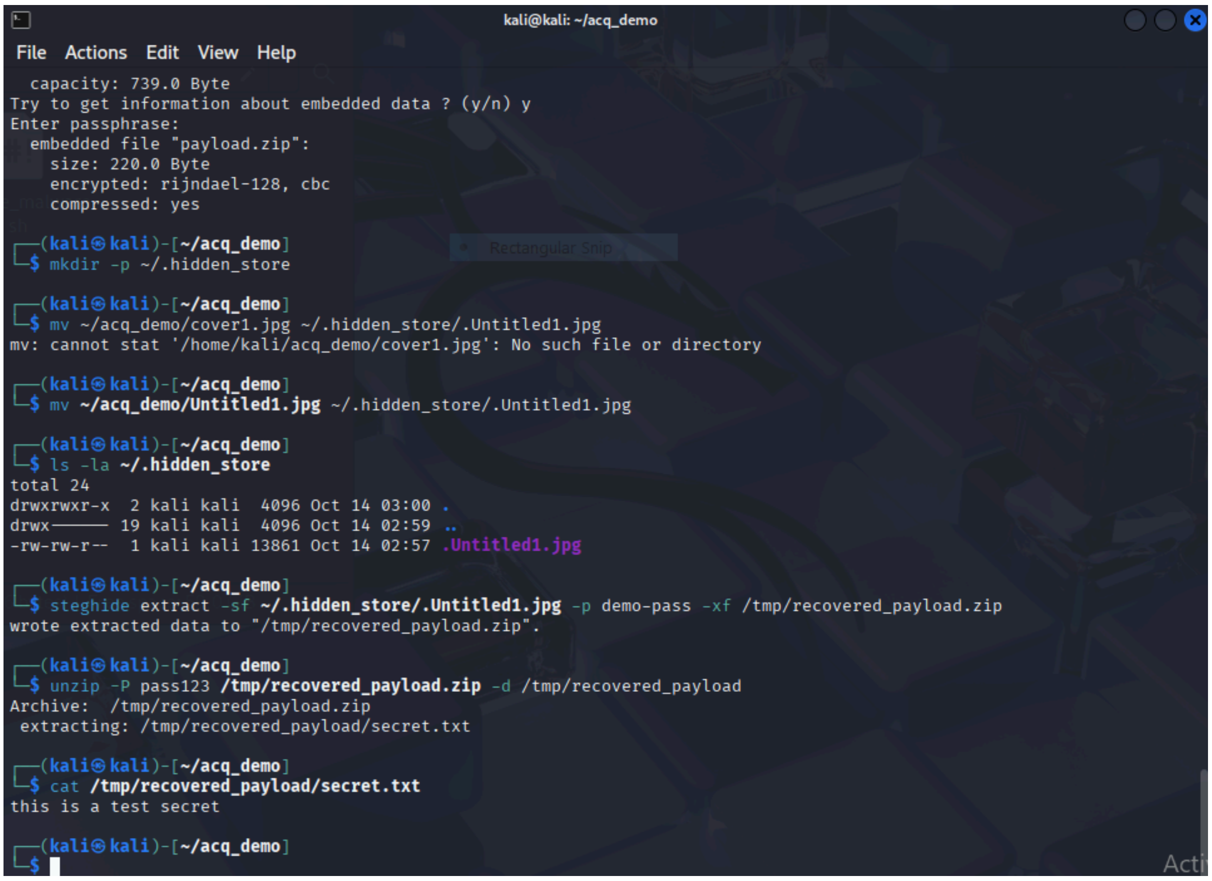
# extract embedded zip (steghide password 'demo-pass')

steghide extract -sf ~/.hidden_store/.cover1.jpg -p demo-pass -xf /tmp/recovered_payload.zip

# unzip with payload password 'pass123' and show the secret

unzip -P pass123 /tmp/recovered_payload.zip -d /tmp/recovered_payload

cat /tmp/recovered_payload/secret.txt



```
  capacity: 739.0 Byte
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "payload.zip":
    size: 220.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes

┌──(kali㉿kali)-[~/acq_demo]
└─$ mkdir -p ~/.hidden_store

┌──(kali㉿kali)-[~/acq_demo]
└─$ mv ~/acq_demo/cover1.jpg ~/.hidden_store/.Untitled1.jpg
mv: cannot stat '/home/kali/acq_demo/cover1.jpg': No such file or directory

┌──(kali㉿kali)-[~/acq_demo]
└─$ mv ~/acq_demo/Untitled1.jpg ~/.hidden_store/.Untitled1.jpg

┌──(kali㉿kali)-[~/acq_demo]
└─$ ls -la ~/.hidden_store
total 24
drwxrwxr-x  2 kali kali  4096 Oct 14 03:00 .
drwx------ 19 kali kali  4096 Oct 14 02:59 ..
-rw-rw-r--  1 kali kali 13861 Oct 14 02:57 .Untitled1.jpg

┌──(kali㉿kali)-[~/acq_demo]
└─$ steghide extract -sf ~/.hidden_store/.Untitled1.jpg -p demo-pass -xf /tmp/recovered_payload.zip
wrote extracted data to "/tmp/recovered_payload.zip".

┌──(kali㉿kali)-[~/acq_demo]
└─$ unzip -P pass123 /tmp/recovered_payload.zip -d /tmp/recovered_payload
Archive:  /tmp/recovered_payload.zip
 extracting: /tmp/recovered_payload/secret.txt

┌──(kali㉿kali)-[~/acq_demo]
└─$ cat /tmp/recovered_payload/secret.txt
this is a test secret

┌──(kali㉿kali)-[~/acq_demo]
└─$ 
```

Viva Questions:

1. How does a reverse TCP payload work in malware?
2. What are the common techniques used to obfuscate malware?
3. What are the ethical and legal implications of creating malware?
4. How can a system be secured against malware attacks?
5. What is the purpose of password-protecting malicious files during experiments?
6. What is a reverse shell, and how does it work?
7. What are some common issues faced during reverse shell connections?
8. How do you verify that a reverse shell is active?
9. What are the risks of using reverse shells in a live network?
10. Explain the role of firewalls and antivirus software in blocking reverse shells.
11. What are the ethical considerations of conducting experiments involving malware?
12. How do you ensure that malware created in the lab doesn't harm real-world systems?
13. What are the consequences of using malware outside a controlled environment?
14. Why is it important to isolate the lab environment during these experiments?
15. If the payload doesn't execute on the target machine, how would you troubleshoot?

16. What would you do if the reverse connection fails?
17. If the target has a firewall blocking outgoing connections, how would you proceed?
18. How would you secure your system against the same vulnerabilities you exploited?