SL Practical

Exp 6: Hashing

```
# Step 1: Update and install hashdeep
sudo apt update
sudo apt install hashdeep -y

# Step 2: Create a demo folder and move inside it
mkdir demo
cd demo

# Step 3: Create sample files
echo "hello john" > file1.txt
echo "this is test" > file2.txt

# Step 4: Generate hash list of all files (store outside folder)
hashdeep -r . > ../hash.txt

# Step 5: Audit the files (no change) → should PASS
echo "===== AUDIT PASS TEST ====="
hashdeep -a -v -k ../hash.txt -r .

# Step 6: Modify one file (simulate file change)
echo "this file is changed" > file1.txt

# Step 7: Audit again after modification → should FAIL
echo "===== AUDIT FAIL TEST ====="
hashdeep -a -v -k ../hash.txt -r .
```

Explaination:

1. **sudo apt update**
   → Updates Ubuntu packages.
2. **sudo apt install hashdeep -y**
   → Installs the hashdeep tool.
3. **mkdir demo**
   → Makes a new folder named *demo*.
4. **cd demo**
   → Opens that folder.
5. **echo "hello john" > file1.txt**
   → Creates file1.txt with text *hello john*.

6. **echo "this is test" > file2.txt**
   → Creates file2.txt with text *this is test*.
7. **hashdeep -r . > ../hash.txt**
   → Generates hashes of both files and saves in *hash.txt* (outside folder).
8. **hashdeep -a -v -k ../hash.txt -r .**
   → Audits and checks if files are same — shows **Audit passed**
9. **echo "this file is changed" > file1.txt**
   → Changes one file.
10. **hashdeep -a -v -k ../hash.txt -r .**
    → Audits again — now shows **Audit failed** because file changed.

Exp 7: Reconnaissance(information gathering) tools(WHOIS, dig, traceroute..)

```
# Step 1: Install all tools
sudo apt update
sudo apt install whois dnsutils traceroute nikto dmitry -y

# Step 2: WHOIS – Domain registration details
whois example.com

# Step 3: DIG – DNS record lookup
dig example.com
dig example.com MX

# Step 4: NSLOOKUP – Get domain to IP mapping
nslookup example.com

# Step 5: TRACEROUTE – Trace path to server
traceroute example.com

# Step 6: NIKTO – Website vulnerability scan
nikto -h http://example.com

# Step 7: DMITRY – Gather domain info and emails
dmitry -winse example.com
```

Explaination:

1. sudo apt update
   → Updates all system packages.
2. sudo apt install whois dnsutils traceroute nikto dmitry -y
   → Installs all reconnaissance tools.
3. whois example.com
   → Shows domain owner, registrar, and expiry info.

4. dig example.com
   → Displays DNS records and IP address.
5. dig example.com MX
   → Shows mail server (MX) records.
6. nslookup example.com
   → Finds IP address of the domain.
7. traceroute example.com
   → Shows all hops between your system and the target server.
8. nikto -h http://example.com
   → Scans website for security vulnerabilities.
9. dmitry -winse example.com
   → Collects domain info, subdomains, and email IDs.

Assignment 8 – Port Scanning using Nmap

Aim:
To install and use Nmap tool to scan open ports, detect OS, and check live hosts.

(verify once – not sure abt this assignment)

Commands:

```
# Step 1: Install Nmap
sudo apt update
sudo apt install nmap -y

# Step 2: Check if it installed properly
nmap --version

# Step 3: Ping Scan (to find live systems)
nmap -sn 192.168.1.0/24

# Step 4: Simple Port Scan
nmap 192.168.1.10

# Step 5: TCP Scan (checks open TCP ports)
nmap -sT 192.168.1.10

# Step 6: UDP Scan
sudo nmap -sU 192.168.1.10

# Step 7: OS Detection
sudo nmap -O 192.168.1.10
```

Step 8: Version Detection
nmap -sV 192.168.1.10

(Replace 192.168.1.10 with your target IP or website like example.com)

Explanation

1. sudo apt install nmap → installs Nmap.
2. nmap --version → checks version (confirm it's installed).
3. nmap -sn → shows which systems are ON in the network.
4. nmap IP → basic port scan.
5. nmap -sT → checks open TCP ports.
6. nmap -sU → checks open UDP ports.
7. nmap -O → tells which Operating System is used.
8. nmap -sV → shows which service version is running (like Apache, SSH, etc).

Assignment 9: DOS Attack

sudo hping3 -c 10000 -d 200 -w 64 -p 21 --flood --rand-source tsec.edu

sudo hping3 -S -P -U --flood -V --rand-source www.hping3testsite.com

Explaination

Command 1:

sudo hping3 -c 10000 -d 200 -w 64 -p 21 --flood --rand-source tsec.edu

Simple breakdown

- sudo → run as administrator (needed to send raw packets).
- hping3 → packet-crafting tool.
- -c 10000 → would send **10000 packets** (a large number).
- -d 200 → sets packet **data size** to 200 bytes (bigger packets = more traffic).
- -w 64 → sets TCP window size to 64 (packet header option).
- -p 21 → target **port 21** (FTP port).
- --flood → send packets **as fast as possible** (no delays) — this makes it a flood.
- --rand-source → use **random source IP addresses** for each packet (spoofs origin).
- tsec.edu → target domain.

Plain language: This tries to blast the target tsec.edu on FTP port with many medium-sized packets as fast as possible, pretending they come from random IPs.\

Command 2

sudo hping3 -S -P -U --flood -V --rand-source www.hping3testsite.com

Simple breakdown

- -S → set **SYN** flag in TCP packets (used to start TCP connections).
- -P → set **PUSH** flag (asks receiver to push data to application).
- -U → set **URGENT** flag (urgent pointer set).
- --flood → send packets **as fast as possible** (flood).
- -V → verbose output (shows more info while sending).
- --rand-source → randomize source IPs (spoofing).
- www.hping3testsite.com → target domain.

Plain language: This sends a continuous flood of specially-flagged TCP packets (SYN+PUSH+URG) with spoofed source addresses to the given site — the flags change packet behavior to make them look like different kinds of TCP traffic.

Assignment 10: IPTABLES

```
# show current rules
sudo iptables -L -n -v

# flush existing rules (start fresh)
sudo iptables -F
sudo iptables -t nat -F
sudo iptables -X

# allow loopback and established replies
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# allow SSH (port 22) so you don't get locked out
sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT

# allow HTTP (80)
sudo iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT

# allow ping (ICMP) - optional
sudo iptables -A INPUT -p icmp -j ACCEPT

# set default policy to DROP (deny by default)
sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT ACCEPT
```

# show final rules with counters
sudo iptables -L -n -v

Explaination:

1. sudo iptables -L -n -v
→ Shows all current firewall rules with details (ports, packets, bytes).

2. sudo iptables -F
→ Deletes (flushes) all rules in the default filter table.

3. sudo iptables -t nat -F
→ Clears all rules in the NAT table.

4. sudo iptables -X
→ Deletes all custom chains (if any).

5. sudo iptables -A INPUT -i lo -j ACCEPT
→ Allows all traffic from the loopback interface (system's own network).

6. sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
→ Allows replies to existing or related connections.

7. sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT
→ Allows new SSH connections on port 22.

8. sudo iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT
→ Allows new HTTP (web) connections on port 80.

9. sudo iptables -A INPUT -p icmp -j ACCEPT
→ Allows ping requests (ICMP).

10. sudo iptables -P INPUT DROP
→ Blocks all other incoming traffic by default.

11. sudo iptables -P FORWARD DROP
→ Blocks packet forwarding between networks.

12. sudo iptables -P OUTPUT ACCEPT
→ Allows all outgoing connections.

13. sudo iptables -L -n -v
→ Shows final configured rules with counters.

Assignment 12: GPG

```
# Step 1: Generate private and public key pairs (for sender and receiver)
gpg --gen-key
# or
gpg --full-generate-key

# Step 2: Export sender's public key (to share with others)
gpg --export -a sender_name > sender_pub.asc
# or
gpg --output sender_pub.asc --armor --export sender_email

# Step 3: Export sender's private key (for backup)
gpg --export-secret-key -a sender_name > sender_private.asc

# Step 4: Generate fingerprint of receiver's key
gpg --fingerprint receiver_email

# Step 5: Import receiver's public key into sender's keyring
gpg --import receiver_pub.asc

# Step 6: List all public keys
gpg --list-keys
# or list a specific user's key
gpg --list-keys receiver_email

# Step 7: Sign receiver's public key (to verify trust)
gpg --sign-key receiver_email

# Step 8: Encrypt file to send (create a file before this)
gpg --encrypt -r receiver_email message.txt
# or encrypt + sign in ASCII format
gpg --encrypt --sign --armor -r receiver_email message.txt

# Step 9: Decrypt received file
gpg -o myfile_decrypted.txt -d message.txt.gpg
```

Explanation:

1. gpg --gen-key → Creates public and private key pair (used for encryption & decryption).
2. gpg --export -a sender_name > sender_pub.asc → Exports sender's public key to share.
3. gpg --export-secret-key -a sender_name > sender_private.asc → Saves private key for backup.
4. gpg --fingerprint receiver_email → Shows unique fingerprint of receiver's key.

5. gpg --import receiver_pub.asc → Imports receiver's public key into keyring.
6. gpg --list-keys → Displays all public keys in your keyring.
7. gpg --sign-key receiver_email → Signs receiver's key to verify their identity.
8. gpg --encrypt -r receiver_email message.txt → Encrypts the file using receiver's public key.
9. gpg -o myfile_decrypted.txt -d message.txt.gpg → Decrypts the received file using your private key.

STEPS:

```
# Step 1: Install GPG (if not installed)
sudo apt update
sudo apt install gnupg -y
```

→ Installs GPG.

```
# Step 2: Generate keys for Sender
gpg --full-generate-key
# When prompted:
#  - Choose (1) RSA and RSA
#  - Enter keysize (e.g., 3072 or 4096)
#  - Set expiry (or 0 for no expiry)
#  - Enter Real name: Sender
#  - Enter Email address: sender@example.com
#  - Enter passphrase (optional for lab)
```

→ Creates Sender public/private key pair.

```
# Step 3: Generate keys for Receiver (repeat)
gpg --full-generate-key
# Enter Real name: Receiver
# Enter Email address: receiver@example.com
```

→ Creates Receiver public/private key pair.

```
# Step 4: List keys to verify both exist
gpg --list-keys
```

→ Shows Sender and Receiver public keys in keyring.

```
# Step 5: Export Sender's public key to file (to share)
gpg --export -a "sender@example.com" > sender_pub.asc
```

→ Creates ASCII file sender_pub.asc (shared public key).

```
# Step 6: Export Receiver's public key to file
gpg --export -a "receiver@example.com" > receiver_pub.asc
```

→ Creates receiver_pub.asc.

Step 7: (Optional) Export private keys for backup (DO NOT share)
gpg --export-secret-key -a "sender@example.com" > sender_private.asc
gpg --export-secret-key -a "receiver@example.com" > receiver_private.asc

→ Saves private keys (keep secure; only for backup).

Step 8: Show fingerprint (verify identity)
gpg --fingerprint "receiver@example.com"
gpg --fingerprint "sender@example.com"

→ Prints unique fingerprint for each key (verify with other person).

Step 9: Import Receiver's public key into Sender's keyring
gpg --import receiver_pub.asc

→ Sender can now encrypt to Receiver.

Step 10: Import Sender's public key into Receiver's keyring
gpg --import sender_pub.asc

→ Receiver can verify signatures from Sender.

Step 11: (Optional) Sign Receiver's public key (Sender trusts Receiver)
gpg --sign-key "receiver@example.com"
Follow prompts to confirm and enter your passphrase

→ Sender cryptographically signs Receiver's key (shows trust).

Step 12: Encrypt a file as Sender for Receiver
echo "This is a secret message" > message.txt
gpg --encrypt -r "receiver@example.com" message.txt
Output: message.txt.gpg

→ Creates encrypted file message.txt.gpg for Receiver.

OR: Encrypt and sign (Sender signs and encrypts)
gpg --encrypt --sign --armor -r "receiver@example.com" message.txt
Output: ASCII armored file message.txt.asc

→ Creates encrypted+signed ASCII file.

Step 13: Receiver decrypts the file
gpg -o message_decrypted.txt -d message.txt.gpg
or if ascii armored: gpg -o message_decrypted.txt -d message.txt.asc

→ Receiver gets decrypted file message_decrypted.txt.

Step 14: Verify signature (if file was signed)

gpg --verify message.txt.asc

→ Shows whether signature is valid and who signed it.

Step 15: List keys and secret keys
gpg --list-keys
gpg --list-secret-keys

→ Verify keys and secret key presence.

Step 16: Remove imported keys (cleanup - optional)
gpg --delete-key "sender@example.com"
gpg --delete-secret-key "sender@example.com"
gpg --delete-key "receiver@example.com"
gpg --delete-secret-key "receiver@example.com"

→ Removes keys from keyring (use with caution).