EXP 1

Maintain confidentiality

1. echo “different passwords and username” > log_file.txt
2. chmod 600 log_file.txt
3. sudo adduser alice(enter details and passowrds)
4. su alice
5. cat log_file.txt (it’ll show permission denied)

EXP 2

Maintain integrity

1. sha256 /var/log/syslog (checking before)
2. sudo nano /var/log/syslog (after opening modify the file with Jan 1 12:00:00 UnauthorizedAccess: Admin login) then press Ctrl+O and Ctrl+X
3. sha256 /var/log/syslog (checking after)
4. observe the difference before and after

EXP 3

Maintain availability

1. sudo apt install apache2-utils
2. python3 -m http.server 808 (click on the http link)
3. then keeping current server active and open a new terminal
4. ab -n 1000 -c 100 http://localhost:8080/ (c is concurrent requests and n is requests)
5. then just press ctrl+C to stop the attack and observe total request

EXP 4

Do 1 and 3 again

EXP 5

DAC implementation

1. mkdir dac_demo && cd dac_demo
2. touch confidential.txt
3. chmod 600 confidential.txt
4. sudo adduser alice
5. sudo chown alice:alice confidential.txt
6. su alice
7. cat confidential.txt
8. change the user back to admin or wtv and reacess the fie it wont allow u

EXP 6

MAC implementation

1. sudo apt install policycoreutils selinux-utils selinux-basics
2. sudo selinux-activate
3. sudo selinux-config-enforcing
4. sudo nano /etc/selinux/config (this is to check if its in enforcing or not)
5. ls -Z confidential.txt
6. su alice
7. cat confidential.txt
8. exit
9. sudo cat /var/log/audit/audit.log

EXP 7

RBAC implementation

1. mkdir rbac && cd rbac
2. sudo adduser alice
3. sudo groupadd managers
4. sudo usermod -aG managers alice
5. touch manager_notes.txt
6. sudo chown :managers manager_notes.txt
7. sudo chmod 770 manager_notes.txt
8. su alice
9. cat manager_notes.txt
10. exit
11. there shld be no output for alice ie permission in granted

EXP 8

Private and public key

1. mkdir key && cd key
2. openssl genpkey -algorithm RSA -out private.key -aes256
3. openssl rsa -pubout -in private.key -out public.key
4. openssl req -new -key private.key -out user.csr
5. openssl req -x509 -key private.key -in user.csr -out user_cert.crt -days 365
6. openssl genpkey -algorithm RSA -out ca.key -aes256
7. openssl req -x509 -key ca.key -out ca.crt -days 3650
8. openssl x509 -req -in user.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out user_signed_cert.crt -days 365

EXP 9

Scanning open holes with nmap

1. sudo apt update && sudo apt upgrade -y
2. sudo apt install nmap -y
3. ip a
4. nmap -sV -p- 10.10.10.6

EXP 10: Perform AES Encryption & Decryption (CHECK NOTES AT THE BOTTOM)

1. sudo apt update && sudo apt upgrade -y
2. sudo apt install python3 python3-pip -y
3. pip3 install cryptography pandas faker
4. python3 --version
5. pip3 list | grep -E "cryptography|pandas|faker"
6. nano aes_key.py (this will make a python file, write code in step 7)
7.
   ```
   from cryptography.fernet import Fernet
   key = Fernet.generate_key()
   with open("aes_key.key", "wb") as key_file:
               key_file.write(key)
               print(f"Generated AES Key: {key.decode()}") to copy press ctrl+shift+c, to paste use
   ctrl+shift+v
   ```
8. python3 aes_key.py(write this back in ubuntu)
9. nano aes_encrypt.py
10.
    ```
    from cryptography.fernet import Fernet
    # Load AES key
    key = open("aes_key.key", "rb").read()
    cipher = Fernet(key)
    # Message to encrypt
    message = "Confidential Data: Do not share!"
     # Encrypt the message
    encrypted_message = cipher.encrypt(message.encode())
    print(f"Encrypted Message: {encrypted_message.decode()}")
    ```
11. python3 aes_encrypt.py
12. nano aes_decrypt.py(again open a new tab, in which u will paste the below code)
13.
    ```
    from cryptography.fernet import Fernet
    # Load AES key
    key = open("aes_key.key", "rb").read()
    cipher = Fernet(key)
    # Encrypted message (replace with actual encrypted message, the output from
    aes_encrypt.py)
    encrypted_message = b'ENCRYPTED_MESSAGE_HERE'
    # Decrypt the message
    decrypted_message = cipher.decrypt(encrypted_message).decode()
    print(f"Decrypted Message: {decrypted_message}")
    ```
14. python3 aes_decrypt.py

EXPERIMENT 11: Perform RSA Encryption & Decryption (CHECK NOTES AT THE BOTTOM)

1.
   ```
   sudo apt update && sudo apt upgrade -y
   ```
2.
   ```
   sudo apt install python3 python3-pip -y
   ```
3.
   ```
   pip3 install cryptography pandas faker
   ```
4.
   ```
   python3 --version
   ```
5.
   ```
   pip3 list | grep -E "cryptography|pandas|faker"
   ```
6.
   ```
   nano rsa_key.py
   ```
7.
   ```
   from cryptography.hazmat.primitives.asymmetric import rsa
   from cryptography.hazmat.primitives import serialization
   # Generate private key
   private_key = rsa.generate_private_key(
           public_exponent=65537,
           key_size=2048

   )

   with open("rsa_private.pem", "wb") as f:

           f.write(private_key.private_bytes

           (

                   encoding=serialization.Encoding.PEM,
                   format=serialization.PrivateFormat.TraditionalOpenSSL,
                   encryption_algorithm=serialization.NoEncryption()

           )

   )

   # Generate public key

   public_key = private_key.public_key()

   # Save public key

   with open("rsa_public.pem", "wb") as f:

           f.write(public_key.public_bytes

           (

                   encoding=serialization.Encoding.PEM,
                   format=serialization.PublicFormat.SubjectPublicKeyInfo

           )

   )

   print("RSA key pair generated and saved.")
   ```

8. nano rsa_encrypt.py

```
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization

# Load the public key

with open("rsa_public.pem", "rb") as f:

        public_key = serialization.load_pem_public_key(f.read())

# The message to encrypt

message = b"Secure Data Transfer”


# Encrypt the message using RSA public key

encrypted = public_key.encrypt

(

        message,

        padding.OAEP

        (

                mgf=padding.MGF1(algorithm=hashes.SHA256()),
                algorithm=hashes.SHA256(),

                label=None

        )

)

print(f"Encrypted Data: {encrypted}")
```

9. nano rsa_decrypt.py
10. from cryptography.hazmat.primitives.asymmetric import padding

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization

# Load the private key
with open("rsa_private.pem", "rb") as f:
        private_key = serialization.load_pem_private_key(f.read(), password=None)
```

```
#The encrypted message (use the result from rsa_encrypt.py)
 encrypted_message = b'ENCRYPTED_MESSAGE_HERE' # Replace with the actual encrypted message

# Decrypt the message using RSA private key
decrypted = private_key.decrypt
(
        encrypted_message,
        padding.OAEP
        (
                mgf=padding.MGF1(algorithm=hashes.SHA256()),
                algorithm=hashes.SHA256(),
                label=None

        )

)

print(f"Decrypted Data: {decrypted.decode()}")
```

11. python3 rsa_decrypt.py
    there may be a syntax error in the output when you run this. this may happen due to the backslashes that may appear in the encrypted message. This may or may not happen but if it does explain this to maam ig

EXPERIMENT 12: Perform Data Anonymization (CHECK NOTES AT THE BOTTOM)

1. sudo apt update && sudo apt upgrade -y
2. sudo apt install python3 python3-pip -y
3. pip3 install cryptography pandas faker
4. python3 --version
5. pip3 list | grep -E "cryptography|pandas|faker"
6. sudo apt install faker
7. nano mask.py
8. import pandas as pd
9.
   ```
   # Data containing SSNs
   data = {"SSN": ["123-45-6789", "987-65-4321", "555-44-3333"]}
   df = pd.DataFrame(data)

   # Masking SSN (showing only the last 4 digits)
   df["Masked_SSN"] = df["SSN"].str.replace(r"\d{3}-\d{2}", "***-**-", regex=True)

   # Display the masked
   print(df)
   ```

10. Nano fake_data.py
    ```
    from faker import Faker
    # Create a Faker instance
    fake = Faker()

    # Generate and print 5 fake records
    for _ in range(5):
        print(fake.name(), "-", fake.email(), "-", fake.phone_number())
    ```

NOTES

- When you are saving txt or py files after putting text/codes, to save first do Ctrl+O then press Enter and then Ctrl+X to exit
- If you want to make changes to the txt or py file, simply run “nano filename” again