**1.Develop a Python program that categorizes cyber threats based on severity levels.**

```
# List of cyber attacks and their severity
cyber_attacks = {
    "phishing": "Medium",
    "malware": "High",
    "ransomware": "Critical",
    "ddos": "High",
    "spyware": "Medium",
    "spam": "Low"
}

# User input
attack = input("Enter cyber attack name: ").lower()

print("\nCYBER ATTACK SUMMARY")
print("--------------------")

# Check and display summary
if attack in cyber_attacks:
    print("Attack Name :", attack.capitalize())
    print("Severity    :", cyber_attacks[attack])
else:
    print("Attack not found in the list.")
```

**3. Write a script to generate fake phishing emails for educational purposes**

```
import random
senders = [
    "Admin <admin@training-example.com>",
    "IT Helpdesk <helpdesk@secure-training.net>",
    "Security Team <security@education-lab.org>",
    "Bank Alert <alerts@bank-training.com>",
    "Customer Care <support@mock-service.org>"
]
subjects = [
    "URGENT: Verify your account immediately!",
    "Password Expiration Notice - Action Required",
```

```
    "Unusual Login Attempt Detected",
    "Congratulations! You've won a reward!",
    "Payment Failed: Update your billing info now"
]
greetings = [
    "Dear Customer,",
    "Hello User,",
    "Attention Account Holder,",
    "Dear Valued Member,"
]
bodies = [
    "We noticed unusual activity on your account. Please review it immediately.",
    "Your password will expire soon. Reset it to avoid service interruption.",
    "You are eligible for a limited-time prize. Claim it before it expires!",
    "We could not process your recent payment. Update your billing details.",
    "Your account may be suspended unless you verify your information."
]

fake_links = [
    "http://training-link-example.com",
    "http://secure-education.net/reset",
    "http://awareness-lab.org/verify",
    "http://mock-notification.edu/action",
    "http://cyber-training-demo.com/update"
]
signoffs = [
    "Best regards,\nIT Security Team",
    "Thank you,\nAccount Support",
    "Sincerely,\nCustomer Service",
    "Yours truly,\nSecurity Department"
]

def generate_phishing_email():
    """Generate a safe, fake phishing email for training."""
    sender = random.choice(senders)
    subject = random.choice(subjects)
    greeting = random.choice(greetings)
    body = random.choice(bodies)
    link = random.choice(fake_links)
```

```
    signoff = random.choice(signoffs)

    email = f"""
From: {sender}
Subject: {subject}

{greeting}

{body}

Click here to proceed: {link}

{signoff}
"""
    return email

if __name__ == "__main__":
    print("\n=== Fake Phishing Email (Training Purposes Only) ===\n")
    print(generate_phishing_email())
    print("=" * 60)
```

**4. Develop a Python script using YARA rules to detect malware signatures in files.**

```
import yara
rule = """
rule MalwareExample
{
    strings:
        $a = "malicious_code"          // Text string pattern
        $b = { E8 ?? ?? ?? ?? 83 C4 04 } // Hex pattern with wildcards

    condition:
        $a or $b
}
"""
rules = yara.compile(source=rule)
file_path = input("Enter the file path to scan: ")
matches = rules.match(file_path)
if matches:
    print("\n⚠ Malware detected! Signature(s) matched:")
```

```
    for match in matches:
        print(" -", match.rule)
else:
    print("\n✔ No malware signature detected in the file.")
```

**5.Write a Python script using scapy to capture and analyze network traffic.**

```
from scapy.all import sniff, IP, TCP, UDP
def analyze_packet(packet):
    if packet.haslayer(IP):
        src = packet[IP].src
        dst = packet[IP].dst

        if packet.haslayer(TCP):
            proto = "TCP"
        elif packet.haslayer(UDP):
            proto = "UDP"
        else:
            proto = "Other"

        print(f"Source IP: {src} --> Destination IP: {dst} | Protocol: {proto}")

print("Starting packet capture... (Press Ctrl+C to stop)")
sniff(prn=analyze_packet, count=10)
```

**6.Develop a Python program using pycryptodome to encrypt and decrypt a file using AES.**

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import os
# File names
input_file = "sample.txt"
encrypted_file = "encrypted.bin"
decrypted_file = "decrypted.txt"

key = get_random_bytes(16)
```

```
cipher = AES.new(key, AES.MODE_EAX)

with open(input_file, "rb") as f:
    data = f.read()

ciphertext, tag = cipher.encrypt_and_digest(data)

with open(encrypted_file, "wb") as f:
    f.write(cipher.nonce)
    f.write(tag)
    f.write(ciphertext)

print("File encrypted successfully.")

with open(encrypted_file, "rb") as f:
    nonce = f.read(16)
    tag = f.read(16)
    ciphertext = f.read()

cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
plaintext = cipher.decrypt_and_verify(ciphertext, tag)

with open(decrypted_file, "wb") as f:
    f.write(plaintext)

print("File decrypted successfully.")
```

**7.Write a Python script using bcrypt to hash passwords and verify them securely.**

```
import bcrypt
password = input("Enter password: ").encode('utf-8')

hashed_password = bcrypt.hashpw(password, bcrypt.gensalt())
print("\nHashed Password:", hashed_password)

check_password = input("\nRe-enter password to verify: ").encode('utf-8')

if bcrypt.checkpw(check_password, hashed_password):
    print("✔ Password verified successfully!")
else:
```

```
    print("❌ Password verification failed!")
```

**8.Write a Python program using cryptography to sign and verify messages**

```
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes

private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048
)
public_key = private_key.public_key()
message = b"Hello, this is a secure message"

signature = private_key.sign(
    message,
    padding.PKCS1v15(),
    hashes.SHA256()
)
print("Message signed successfully.")

try:
    public_key.verify(
        signature,
        message,
        padding.PKCS1v15(),
        hashes.SHA256()
    )
    print("✔ Signature verified. Message is authentic.")
except:
    print("❌ Signature verification failed.")
```

**9. Develop a Python script using pystegano to hide and extract messages in images**

```
from stegano import lsb
secret_message = input("Enter secret message to hide: ")

secret_image = lsb.hide("input.png", secret_message)
secret_image.save("hidden.png")
```

```
print("Message hidden successfully in hidden.png")

extracted_message = lsb.reveal("hidden.png")

print("Extracted Message:", extracted_message)
```