Cyber Security Internship Tasks - CodeAlpha

Task 1: Basic Network Sniffer (Python Code)

This task involves creating a basic network sniffer in Python that captures and analyzes network traffic.

Python Code:

```
import socket
import struct

def sniff_packets():
    conn = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(3))
    print("Sniffing started... Press Ctrl+C to stop.")
    try:
        while True:
            raw_data, addr = conn.recvfrom(65536)
            dest_mac, src_mac, eth_proto = struct.unpack('!6s6sH', raw_data[:14])
            print(f"\nEthernet Frame:")
                  print(f"Destination MAC: {get_mac(dest_mac)}, Source MAC: {get_mac(src_mac)}, Protocol:
{eth_proto}")
    except KeyboardInterrupt:
        print("\nSniffing stopped.")

def get_mac(bytes_addr):
    return ':'.join(format(b, '02x') for b in bytes_addr)

if __name__ == "__main__":
    sniff_packets()
```

Task 2: Phishing Awareness Training (Presentation Outline)

This task is about educating people on phishing and how to prevent it.

Cyber Security Internship Tasks - CodeAlpha

Presentation Outline:

1. Introduction to Phishing
   - Definition and real-world impact

2. Types of Phishing Attacks
   - Email phishing, Spear phishing, Smishing, Vishing, Clone phishing

3. Recognizing Phishing
   - Signs and red flags (e.g., fake links, urgent tone, attachments)

4. Examples
   - Screenshots and analysis of common phishing attempts

5. Prevention Tips
   - Don't click unknown links, use 2FA, keep software updated

6. What to Do
   - Report to IT/Security team, scan with antivirus, avoid replying

7. Conclusion
   - Stay alert and always verify

Task 3: Secure Coding Review (Python Example)

This task involves identifying security issues in code and improving it.

Vulnerable Code:

```
import sqlite3

def login(username, password):
    conn = sqlite3.connect('users.db')
    cursor = conn.cursor()
    query = f"SELECT * FROM users WHERE username='{username}' AND password='{password}'"
    cursor.execute(query)
    result = cursor.fetchone()
    return result
```

Issues:

- SQL Injection vulnerability
- No password hashing

Secure Version:

```
import sqlite3
import bcrypt

def secure_login(username, password):
    conn = sqlite3.connect('users.db')
    cursor = conn.cursor()
    cursor.execute("SELECT password FROM users WHERE username=?", (username,))
    result = cursor.fetchone()
    if result and bcrypt.checkpw(password.encode('utf-8'), result[0]):
        return True
    return False
```

Secure Practices:

- Use parameterized queries
- Hash passwords using bcrypt
- Validate all user inputs