Offensive Software Exploitation

Summer 2020

Ali Hadi

@binaryz0ne

Part #6

Jumping Strategies

The Art of Moving in Memory

This work is based on the work of Peter “Corelanc0d3r”, Exploit Writing (Jumping to Shellcode) article…

Jumping Strategies

- Using “jmp esp” was an almost perfect scenario
- Not that ‘easy’ every time!
- Let’s check what other ways to execute/jump to shellcode
- Also, what if you are faced with small buffer sizes!

JMP (or CALL)

Cited [1]

jump (or call) a register that points to the shellcode.

- Use a register that contains the address where the shellcode resides and put that address in EIP.

POP RET

Cited [1]

pop return

- None of the registers point directly to the shellcode, but further down the stack is an address that points to the shellcode

- Load that value into EIP by first putting a pointer to pop ret, or pop pop ret, or pop pop pop ret (all depending on the location of where the address is found on the stack) into EIP.

PUSH RET

Cited [1]

push return

- Slightly different than the “call register” technique
- Can’t find a jmp r32 or call r32 opcode anywhere, then push the address on the stack and then do a ret
- Find a push r32 followed by a ret

JMP [reg + offset]

Cited [1]

jmp [reg + offset]

- Register that points to the buffer containing the shellcode does not point at the beginning of the shellcode!
- Find an instruction which will add the required bytes to the register and then jumps to the register

Blind Return

Cited [1]

blind return

- ESP points to the top of the stack (by definition)
- A ret instruction will pop the last value from the stack and will put that address in EIP

POPAD

Cited [1]

popad (pop all double)

- Loaded order: EDI, ESI, EBP, EBX, EDX, ECX and EAX
- As a result, the ESP register is incremented after each register is loaded (triggered by the popad)
- One popad will thus take 32 bytes from ESP and pops them in the registers in an orderly fashion

Short Jumps

Cited [1]

- Need to jump over just a few bytes
- Short jump (jmp) opcode is 0xEB
- Use a jmp instruction followed by the number of bytes

Example:

- You want to jump 30 bytes, the opcode is 0xEB,0x1E

Conditional Jumps

Cited [1]

jump if condition is met

- Technique is based on the states of one or more of the status flags in the EFLAGS register (CF,OF,PF,SF and ZF)

- If the flags are in the specified state (condition), then a jump can be made to the target instruction specified by the destination operand

- This target instruction is specified with a relative offset (relative to the current value of EIP)

Backward Jumps

Cited [1]

- What if you want to perform a backward jump?
  - jump with a negative offset

- Then, get the negative number and convert it to hex
- Take the dword hex value and use that as argument to a jump
  - 0xEB or 0xE9

Example #1: jump back 7 bytes

- -7 = FFFFFFF9
- so jump -7 would be "\xEB\xF9\xFF\xFF“

Backward Jumps – Cont.

Cited [1]

Example #2: jump back 400 bytes

- -400 = FFFFFE70
- Then jump -400 bytes = "\xE9\x70\xFE\xFF\xFF"

- Pay attention, this opcode is 5 bytes long!

- Note: if you need to stay within a DWORD size (4-byte limit), then you may need to perform multiple shorter jumps in order to get where you want to be…

Weird Relative Backward Jump ☺

Cited [1]

"\x59\xFE\xCD\xFE\xCD\xFE\xCD\xFF\xE1\xE8\xF2\xFF\xFF\xFF“

- Explanation

\x59	*POP ECX*
\xFE\xCD	*DEC CH*
\xFE\xCD	*DEC CH*
\xFE\xCD	*DEC CH*
\xFF\xE1	*JMP ECX*
\xE8\xF2\xFF\xFF\xFF	*CALL [relative -0D]*

- Could be adjusted to fit your needs

References

[1] Peter "Corelanc0d3r", Exploit Writing (Jumping to Shellcode), https://www.corelan.be/index.php/2009/07/23/writing-buffer-overflow-exploits-a-quick-and-basic-tutorial-part-2/

[2] Memory Corruption 101, NYU Poly, Dino Dai Zovi

[3] Vulnserver, Stephen Bradshaw, http://grey-corner.blogspot.com/,

[4] Grayhat Hacking: The Ethical Hacker's Handbook, 3rd Edition

[5] The Shellcoders Handbook

[6] Exploit-DB: http://www.exploit-db.com/

[7] The Art of Exploitation, 2nd Edition

[8] Vulnerability Discovery, http://www.thegreycorner.com/2010/01/introduction-to-vulnerability-discovery.html

[9] SEH Based Overflow Exploit Tutorial, http://resources.infosecinstitute.com/seh-exploit/