Applying Return-oriented Programming

Dr. Jared DeMott
CHIEF HACKING OFFICER

@jareddemott www.vdalabs.com

Overview

Bypass ASLR

Pivot

ROP

- Gadgets
- Chaining
- Techniques

Demo

Close Learning Path

Defeating ASLR

- Not in use
  - Newer protections always take a while to roll out
- OS vulnerability
  - Was a while before ASLR was in good shape
- Address Leak
  - Same or Separate vulnerability
    - Overwrite size of an array that lets attacker search memory for ntdll.dll

Pivot

- Exchange the stack pointer
  - With a register under attacker control
  - xchg eax, esp
- Or something else
  - In our vulnerability
    - Add to ESP
    - Because our input is on the stack

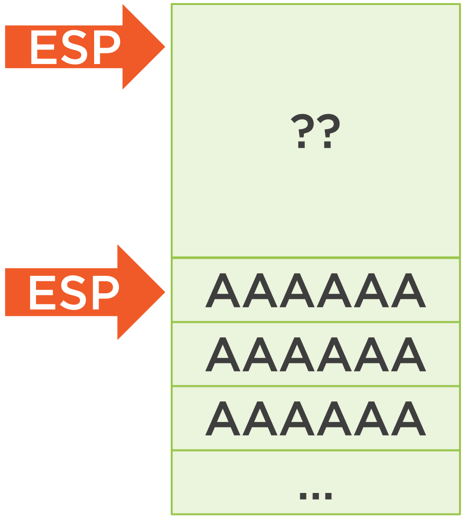
SEH Overwrite

ESP Above Buffer

- ADD ESP, nnn; RET

Example

- add esp,40Ch
- 81 c4 0c040000



Return Oriented Programming

EIP vs. ESP

Classic EIP Code

- N ops = N instructions
- EIP increments
- ESP fluctuates
- The CPU increments EIP automatically

ROP code

- N ops = N frames/gadgets
- ESP increments
- EIP fluctuates
- We have to control ESP through RET instructions

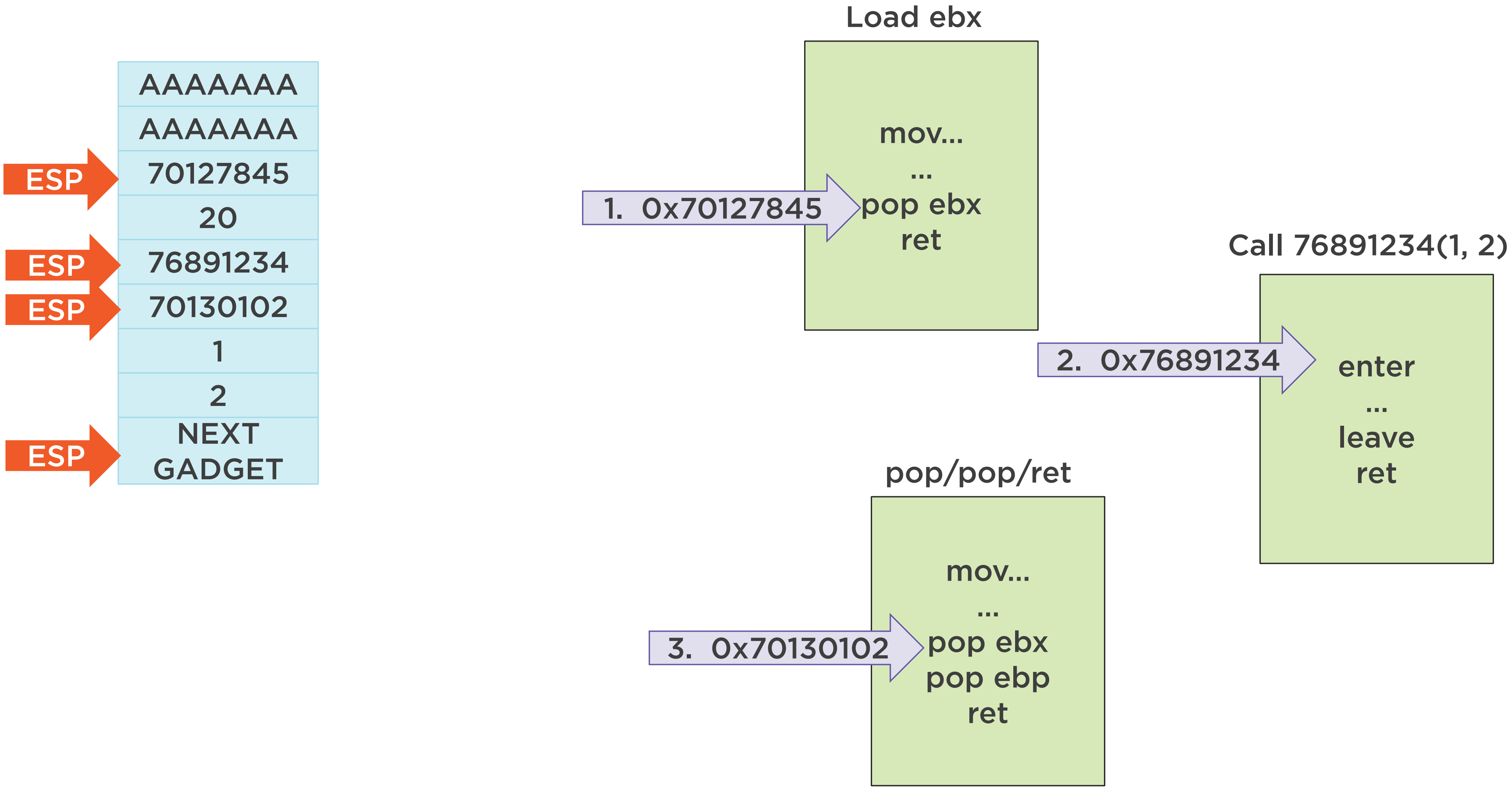
Transform EIP Code to ROP

Load register

Call a function

- With params (1, 2)

```
mov ebx, 20
push 1
push 2
call 76891234
```



AAAAAAA

AAAAAAA

ESP 70127845

20

ESP 76891234

ESP 70130102

1

2

ESP NEXT GADGET

Load ebx

1. 0x70127845 → mov... ... pop ebx ret

Call 76891234(1, 2)

2. 0x76891234 → enter ... leave ret

pop/pop/ret

3. 0x70130102 → mov... ... pop ebx pop ebp ret

Gadgets

ROP Primitives

- Each gadget corresponds to a frame
- As compact as possible
- Commonly required operations
- Should be implemented from function epilogues
  - Must end with a RET
  - Newer
    - JOP -> JMP ending
- Create a dictionary of your own

Gadget	Instruction
Load value into register	POP reg; RET
Read memory at address	POP r1; MOV r2, [r1]; RET
Write value at address	POP r1; POP r2; MOV [r2], r1; RET
Add	ADD reg, n; RET
Increment	INC reg; RET
Call a function	Addr of function
Call a function pointer	POP reg; CALL [reg]; RET
Remove 1-3 dwords from the stack	POP (POP, POP) RET
Remove 6 dwords	ADD ESP, 24; RET
Remove 7 dwords	POPAD; RET
Stack Pivot (esp=eax or esp=ebp)	XCHG EAX, ESP; RET or LEAVE; RET
NOP	RET

Instruction	Opcode
RET	C3
RET n	C2 16bits
POP EAX	58
POP ECX	59
MOV EAX, [ECX]	8B 01
MOV [EAX], ECX	89 08
MOV [ECX], EAX	89 01
INC EAX	40
ADD EAX, n	83 C0 8bits
POP EBX/EDX/ESI/EDI/EBP	5B/5A/5E/5F/5D
POPAD	61
ADD ESP, 24	83 C4 18
CALL [EAX]	FF 10
XCHG EAX, ESP	94
LEAVE	C9 (mov esp, ebp; pop ebp)

Finding Gadgets

Windbg

- .load pykd.pyd
- !py mona

Immunity Debugger

- !find_gadget
- !mona

More

- https://github.com/JonathanSalwan/ROPgadget
- https://github.com/sashs/Ropper

ROP Techniques

Many Techniques on Windows

- VirtualProtect
  - Change page permissions where shellcode is
  - Jump to code
- VirtualAlloc or NtAllocateVirtualMemory
  - Allocate a new RWX page
  - Copy shellcode there
  - Jump to it
- Plenty of other ways

VirtualProtect

Changes the Protection on a Region of Committed Pages in the Virtual Address Space of the Calling Process

```
BOOL WINAPI VirtualProtect(
    __in   LPVOID lpAddress,          (0471c340)
    __in   SIZE_T dwSize,          (00000201)
    __in   DWORD flNewProtect,     (00000040)
    __out  PDWORD lpflOldProtect (7c38c510)
    );
```

Just a Writable Location

Demo

Code Reuse Exploit

- Examine crash
- Use mona to create a ROP chain
- Update exploit
- Win!

Lab 5

- Browser bug to play with
- Pivot
- Create a ROP chain
- Land the exploit

More to Learn in Exploitation?

- EMET
- Flash
- UaFs and TC
- Isolated heap and deferred free
- Kernel exploits
  - Chaining exploits to escape sandbox
- CFI/CFG

Domain and Application Specific

- Auto, embedded, ICS, IoT, etc.

Summary

The Battle Goes On

- Some vendors bent on protecting
  - Newer OS/chip/compiler mitigations
- Attackers
  - Newer exploit techniques
  - Or revert back to less complicated
- Thank you
  - Security for Hackers and Developers Learning path