第十五章

緩衝區溢出

緩衝區溢出

- 2000 年 10 月 19 日，洛杉磯數以百計的飛機無法著陸或延遲，這是因為墨西哥方面的航管人員在輸入飛航描述資料時，輸入九個字（原來應該只能輸入五個），結果造成緩衝區溢出，軟體系統大亂所致。
- 這是誰的錯？是墨西哥的航管人員的疏忽還是程式設計師的錯誤？

緩衝區溢出

- 緩衝區溢出就是使用者在輸入資料時，因為程式設計的缺失，影響到程式的執行或效能的改變。

- 會造成這種問題，通常都是在使用緩衝區(記憶體)時，沒有檢查緩衝區(記憶體)的邊界範圍限制，或者根本就不知道要檢查這件事。

緩衝區溢出

- 在 C 語言中，gets()、strcat()、strcpy()、sprintf()、scanf()、vsprintf()、bcopy() 等，都可能造成緩衝區溢出的問題，但是在一般的學校教學或程式語言學習的書籍中，幾乎都不會提到這個問題。
- 上面的這些函數都不會檢查其所配置的緩衝區(記憶體)有多大，是否容量夠大到允許資料複製放入緩衝區(記憶體)。
- 即便是有經驗的程式設計師也可能因為忘記而忽略這個問題。

緩衝區溢出的原理

```
char str1[10];
char str2[]="abcdefghijklmn";
strcpy(str1,str2);
```

- 當這個程式被編譯及執行，緩衝區溢出就發生。
- 因為 str2 的字串大於 str1 所給的空間， strcpy 把 str2 複製到 str1 ，當然會發生問題。

緩衝區溢出的原理

- 要防止上述程式的緩衝區溢出，完整的程式碼類似如下：

```
const int BUFFER_SIZE = 10;
char str1[BUFFER_SIZE];
char str2[]="abcdefghijklmn";
/* 下面的程式碼會核對緩衝區要有足夠的空間*/
if ((str1 != 0) && (strlen(str2) < BUFFER_SIZE)) {
strncpy(str1,str2, BUFFER_SIZE);
} else {
/* 發生錯誤進行處理 */
}
```

緩衝區溢出的原理

- 緩衝區溢出是攻擊者進行攻擊時，最常用的技巧之一，攻擊者會利用執行中的程序(running process)權限來執行入侵程式碼，但是這也有其難度存在，攻擊者必須依據不同架構、作業系統而有不同的方法，並且要能正確猜中遠端緩衝區的位址。

攻擊者的基本知識

(1)程式語言與作業系統的能力。

(2)了解堆疊。

(3)記憶體的使用。

程式語言與作業系統的能力

1.C 函數及堆疊。

2.一點機器語言及組合語言的知識。

3.如何進行系統呼叫(system calls) 。

4.exec( ) 系統呼叫。

5.如何猜(測試)某些關鍵的參數。

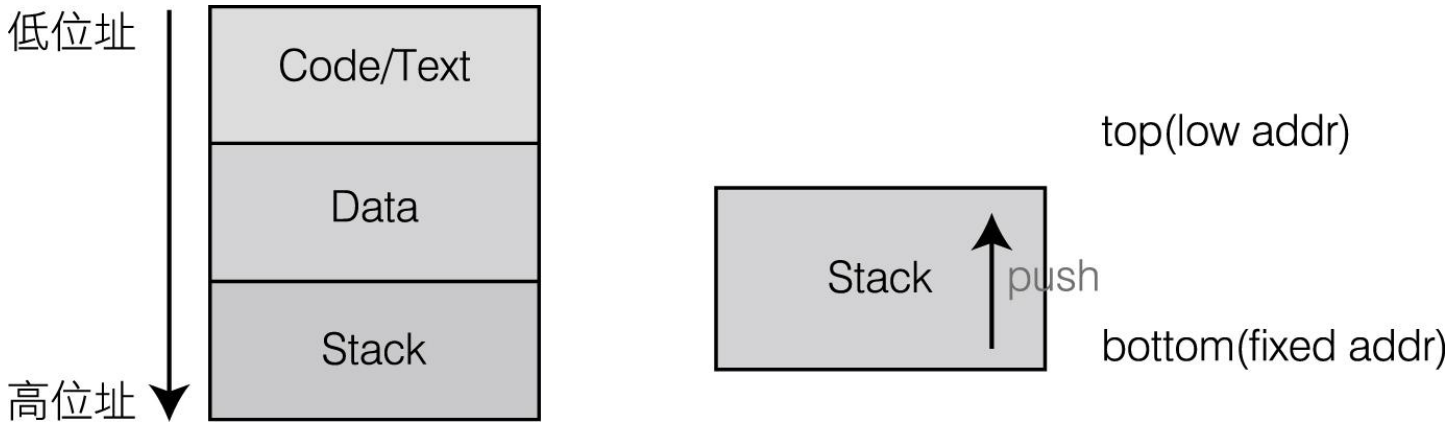
了解堆疊

1. PUSH – 放一個項目到堆疊裡面。
2. POP – 從堆疊的頂上移除一個項目。其運作事實上是送回指標所指的內容，並將指標改變。

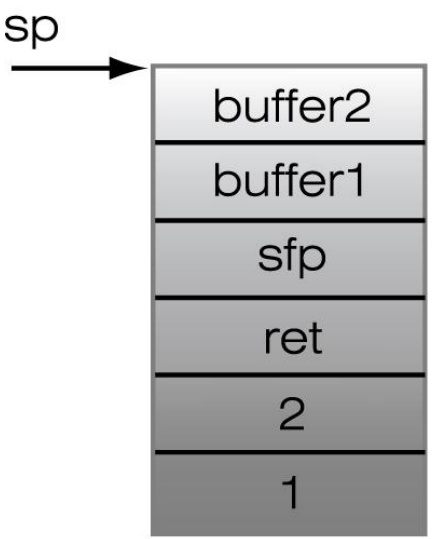
記憶體的使用

- 程式碼區段(Code/Text segment）。
- 資料區段(Data segment)。
- 堆疊區段(Stack segment)。

堆疊區段


低位址
Code/Text
Data
Stack
高位址
top(low addr)
Stack
push
bottom(fixed addr)

堆疊與程式



低位址

宣告的變數

呼叫函式的stack frame位址

返回位址

函式參數

高位址

```
void function(int a,int b)
{
        char buffer1[5];
        char buffer2[5];
}

void main()
{
        function(1,2);
}
```

Shellcode

```
char xd[] =
    "\x31\xc0"
    "\x50"
    "\x68\x6f\x72\x7a\x0a"
    "\x89\xe1"
    "\xb0\x04"
    "\x31\xdb"
    "\xb3\x01"
    "\xb2\x05"
    "\xcd\x80"
    "\x31\xc0"
    "\x31\xdb"
    "\xb0\x01"
    "\xcd\x80"
    ;
```

```
xor eax,eax
    push eax
    push 0x0a7a726f
    mov ecx,esp
    mov al,4h
    xor ebx,ebx
    mov bl,1h
    mov dl,5
    int 80h
    xor eax,eax
    xor ebx,ebx
    mov al,1h
    int 80h
```

緩衝區溢出的型態

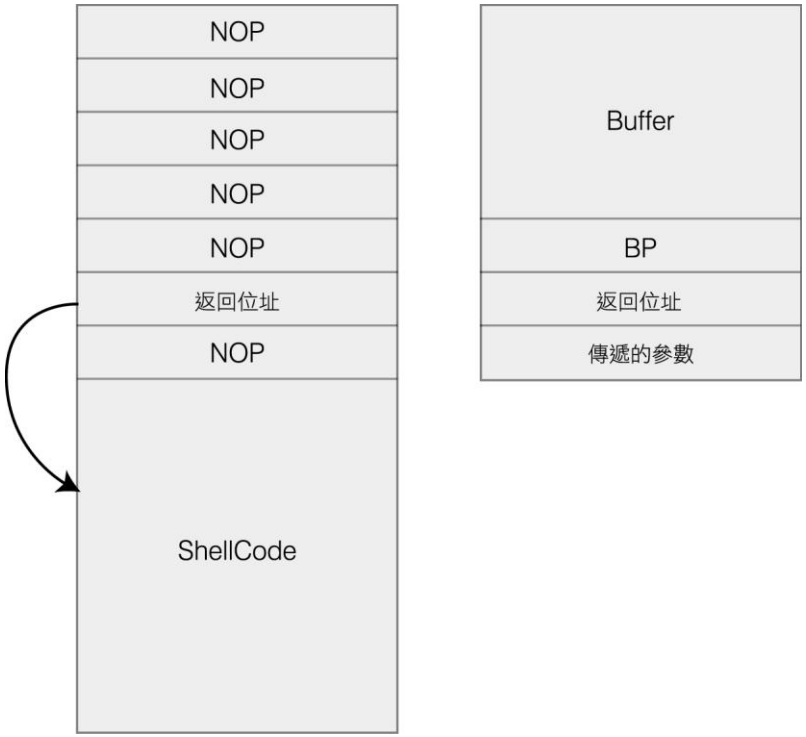
(1)靜態溢出(Stack overflow)

(2)堆積溢出(Heap overflow)

靜態溢出

```
void function(char *str) {
    char buffer[10];
    strcpy(buffer,str);
}
void main() {
    char large_str[255]; int I;
    for (I=0;I<255;I++) large_str[I] =‘A’;
    function(large_str);
}
```

靜態溢出



緩衝區溢出的對策

- 檢查程式碼，檢查字串的宣告，在函數或方法的區域變數宣告中核對邊界，檢查是否存在緩衝區溢出的問題。
- 餵大量資料給應用程式，尤其是資料輸入的部分，並且檢視程式有沒有不正常的執行狀況。
- 尋找更嚴謹及更安全的函數庫支援。
- 關閉堆疊執行。(似乎不是很好的方法)
- 更好的編譯器的技術。(只能期待)

工具程式

- **RAD (Return Address Defender)**

- RAD 會自動加上保護的程式碼到應用程式中，RAD 並不會改變堆疊的配置。

StackGuard

- 可防止程式遭受 "stack smashing" 攻擊，因為緩衝區溢出通常都會改寫函數返回位址，stackguard是個編譯器Patch，它產生一個 canary 資料放到返回位址的前面，如果當函數返回時，發現這個 canary 的值被改變了，就證明可能有人正在試圖進行緩衝區溢出攻擊，程式會立刻回應，發送一條入侵警告消息給 syslogd，然後終止程式。