

《软件安全》实验报告

姓名：李雅帆

学号：2213041

班级：信安班

一、实验名称：

格式化字符串漏洞

二、实验要求：

以第四章示例 4-7 代码，完成任意地址的数据获取，观察 Release 模式和 Debug 模式的差异，并进行总结。

三、实验过程：

1.流程分析：

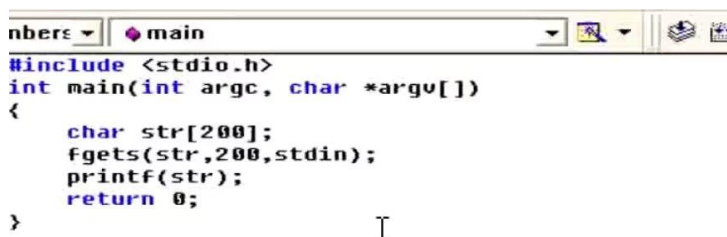
程序首先创建了一个大小为 200 的字符数组，然后从命令行获取一个用户输入的字符串，并将这个字符串进行格式化输出。由于我们要输入的字符串含有格式化字符，但并未给提供对应的参数，因此在输出的过程中函数会自动在堆栈中取获取需要的参数。

本次实验要探究的就是在 Debug 模式和 Release 模式下，函数堆栈的情况，以及执行函数获得的结果的不同。

2.实验过程：

(1) 创建 VC6 项目：

打开虚拟机 XP 系统后，在 VC6 中创建 Win32 的控制台文件，添加源文件并将代码复制到源文件中，debug 无误后，得到相关的程序。



```
nbers | main
#include <stdio.h>
int main(int argc, char *argv[])
{
    char str[200];
    fgets(str,200,stdin);
    printf(str);
    return 0;
}
```

(2) debug 模式：

打开 debug 模式并执行到在程序中设置的断点位置，在 debug 模式的栈帧中，会 push ebp, sub eip, 抬高栈帧，给出大量的一片区域，接下来三个 push 是在整个区域的顶部保存了调用函数前函数中一些计算器的值，接下来利用两条指令来对 108 这么大的空间进行初始化，初始化为 ccccc，在调用 fgets 前有三个 push。

0040147B	- E8 30400000	call __crtGetEnvironmentStringsA	__crtGetEnvironmentStringsA
00401480	- A3 8A354200	mov dword ptr [__aenuptr],eax	
00401485	- E8 16450000	call __setargv	__setargv
0040148A	- E8 C1430000	call __setenvp	__setenvp
0040148F	- E8 CC3D0000	call __cinit	__cinit
00401494	- 8B00 08364200	mov ecx,dword ptr [__environ]	
0040149A	- 8900 0C364200	mov dword ptr [__initenv],ecx	
004014A0	- 8B15 08364200	mov edx,dword ptr [__environ]	
004014A6	- 52	push edx	
004014A7	- A1 08364200	mov eax,dword ptr [__argv]	
004014AC	- 50	push eax	
004014AD	- 8B00 FC354200	mov ecx,dword ptr [__argc]	
004014B3	- 51	push ecx	
004014B4	- E8 4CFBFFFF	call 00401005	main
004014B9	- 83C4 0C	add esp,0C	
004014BC	- 89A5 E4	mov dword ptr [ebp-1C],eax	
004014BF	- 8B55 E4	mov edx,dword ptr [ebp-1C]	
004014C2	- 52	push edx	
004014C3	- E8 D83D0000	call exit	exit
004014C8	- 8B45 EC	mov eax,dword ptr [ebp-14]	
004014CB	- 8B08	mov ecx,dword ptr [eax]	
004014CD	- 8B11	mov edx,dword ptr [ecx]	
004014CF	- 8955 E0	mov dword ptr [ebp-20],edx	
004014D2	- 8B45 EC	mov eax,dword ptr [ebp-14]	
...			
CC		int3	
CC		int3	
> 55		push ebp	ge.main(void)
. 8BEC		mov ebp,esp	
. 81EC 00010000		sub esp,100	
. 53		push ebx	
. 56		push esi	
. 57		push edi	
. 8DBD F8FEFF		lea edi,[ebp-100]	
. B9 42000000		mov ecx,42	
. B8 CCCCCCCC		mov eax,CCCCCCCC	
. F3:AB		rep stos dword ptr [edi]	
. 68 302A4200		push offset _iob	
. 68 C8000000		push 0C8	
. 8D85 38FFFF		lea eax,[ebp-0C8]	
. 50		push eax	
. E8 CC000000		call fgets	fgets
. 83C4 0C		add esp,0C	
. 8D8D 38FFFF		lea ecx,[ebp-0C8]	
. 54		int3	

测试在 Debug 模式下输入"AAAA%x%x%x%x"

```
AAAA%x%x%x%x
```

查看寄存器模块，会发现在 0x0012FB8 中存取了字符串

		3 2 1 0	E S P U O 2 D I
		FST 0000	Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
		FCW 027F	Prec NEAR,53 Mask 1 1 1 1 1 1
0012FE60	0012FE88	ASCII "AAAA%x%x%x%x"	
0012FE64	000000BB		
0012FE68	00422A30	offset main._iob	
0012FE6C	0012DA06		
0012FE70	0012ADB4		
0012FE74	7FFD8000		
0012FE78	00000000		

在 debug 模式下，每 3 个 push，就对应着一个 add，所以调用 add 后，则会返回栈帧状态。

```
C:\Program Files\Microsoft Visual S
AAAA%x%x%x%x
AAAA12da0612adb47ffd8000cccccccc
```

发生了内存泄漏。也可以看到，debug 模式下，如果想读取 str 的地址，需要很多的格式化字符。

(3) release 模式:

在 release 模式中, 可以看到, 调试信息比 debug 更少, 在主函数中, 并没有发现 ebp 入栈, 而是发现了一个 sub 抬高, 此外, 也没有 push 和寄存器的值, 比较简洁, 效率更高。

004011D0	85C0	test eax,eax	
004011D2	75 08	jnz short 004011DC	
004011D4	6A 1C	push 1C	
004011D6	E8 9A000000	call 00401275	
004011D8	59	pop ecx	
004011DC	8365 FC 00	and dword ptr [ebp-4],00000000	
004011E0	E8 610A0000	call 00401C46	
004011E5	FF15 00504000	call dword ptr [<&KERNEL32.GetCommandLineA>]	[KERNEL32.GetCommandLineA]
004011E8	A3 E46D4000	mov dword ptr [406DE4],eax	
004011F0	E8 6A130000	call 0040255F	[ge.0040255F]
004011F5	A3 C4684000	mov dword ptr [4068C4],eax	
004011FA	E8 13110000	call 00402312	[ge.00402312]
004011FF	E8 55100000	call 00402259	[ge.00402259]
00401204	E8 BD0C0000	call 00401EC6	[ge.00401EC6]
00401209	A1 00694000	mov eax,dword ptr [406900]	
0040120E	A3 04694000	mov dword ptr [406904],eax	
00401213	50	push eax	
00401214	FF35 F8684000	push dword ptr [4068F8]	
0040121A	FF35 F4684000	push dword ptr [4068F4]	
00401220	E8 DBFDFFFF	call 00401000	
00401225	83C4 0C	add esp,0C	[Arg3 => [406900] = 0 Arg2 = 0 Arg1 = 0 ge.00401000]
00401228	8945 E4	mov dword ptr [ebp-1C],eax	
0040122B	50	push eax	
0040122C	E8 C20C0000	call 00401EF3	
00401231	8B45 EC	mov eax,dword ptr [ebp-14]	
00401234	8B08	mov ecx,dword ptr [eax]	

没有 ebp 入栈, 只有栈顶抬高

81EC C0000000	sub esp,0C8
8D4424 00	lea eax,[esp]
68 30604000	push offset 00406030
68 C8000000	push 0C8
50	push eax
E8 47000000	call 00401061
8D4C24 0C	lea ecx,[esp+0C]
51	push ecx
E8 0C000000	call 00401030
33C0	xor eax,eax
81C4 D0000000	add esp,0D0
C3	ret
90	nop
90	nop
90	nop
53	push ebx
56	push esi
BE 50604000	mov esi,offset 00406050
57	push edi
56	push esi
E8 50020000	call 00401299
8BF8	mov edi,eax

这个时候, 可知 0x0012FEBC 为要输入的字符串的地址

0012FEAC	0012FEBC	ASCII "AAAA%X%X%X%X"
0012FEB0	0012FEB0	ASCII "AAAA%X%X%X%X"
0012FEB4	000000BB	
0012FEB8	00406030	ASCII "nQ"

最终输出结果: 输入: AAAA%x%x%x%x ;

输出: AAAA: AAAA18FE84BB40603041414141。

在输出结果中 0x41 是 A 的阿斯克码值。

AAAA%X%X%X%X
AAAA12FEB0BB40603041414141

4. Release 和 Debug 模式的差异

在 Debug 模式下, 编译出的版本包含有很多的调试信息, 程序局部变量的内存空间会被扩大, 并且全部初始化为 0xCC, 并且会在栈中保存一些寄存器的信息, 如 EDI,ESI,EDX 等。而在 Release 模式下, 往往会进行优化, 删除调试信息, 以达到代码最小和速度最优的目的, 此时栈中不再保留不必要的内容, 并且函数局部变量的空间也不会超过所需要的大小。

四、心得体会：

在进行格式化字符串漏洞实验时, 我认识到了编程中的安全风险和漏洞可能带来的严重后果。在 Debug 模式下, 由于编译器会为了调试方便而保留更多的信息, 使得观察函数堆栈和变量更加容易, 有助于发现潜在的问题。而在 Release 模式下, 由于进行了优化, 函数堆栈信息会被缩减, 使得漏洞难以被发现, 从而增加了安全风险。

这个实验让我更加了解了漏洞利用的原理, 也让我认识到编程中安全意识的重要性。在编程过程中, 需要注意输入的合法性检查和避免使用不安全的函数, 以减少潜在的安全漏洞。同时, 及时更新代码、学习安全编程最佳实践, 也是保障软件安全的重要措施。