dop

本次攻击通过操纵数据指针，修改内存变量，在不违反控制流逻辑的前提下使程序按攻击者的意图去执行。攻击所需的数据保存在内存中。

以下代码是用来调度攻击的：

while(limit--){

choose = limit + 1;

//从管道中取数据给buf

read(attack\_read, &buf, 256); //缓冲区溢出漏洞点，

if(choose == 1)

pmf1->foo();

else if(choose == 2)

pmf2->foo();

else

set\_string(shellcode);

\*p = \*q;

}

while函数为调度器,limit为循环变量，choose为条件变量。

调度器中read(attack\_read, &buf, 256)为选择器，是可以被攻击者控制的输入，每次读取攻击者的输入，并且通过溢出可以控制循环变量和条件分支。

\*p = \*q为功能配件。

Systemaddr（）函数的功能是泄露地址。Attact（）函数实现攻击。

漏洞程序简介：

假设攻击者熟知源代码与栈布局，程序存在一个内存泄露使得攻击者获知了栈中某个变量的地址，因而知道了所有栈变量的地址。

结构体保存的是函数指针，指向结构体的数据指针，vul函数的栈布局。

正常情况下，每次输入一个字符，程序依次输出hello。

攻击实现的过程：

在漏洞程序vul中声明了一串连续的内存空间，其布局为：（按地址由低到高显示）

|  |  |  |
| --- | --- | --- |
| 变量 | 类型 | 大小 |
| buf | char | 1B |
| old\_value | int | 4B |
| new\_value | int | 4B |
| limit | int | 4B |
| choose | int | 4B |
| p | int\* | 8B |
| q | int\* | 8B |
| shellcode | char[] | 256B |

各指针指向的内容如下：

|  |
| --- |
| pmf1->mf1->f1() |
| pmf2->mf2->f2() |
| pmf3->mf3->systemaddr() |
| p->old\_value |
| q->new\_value |

通过5次精心构造的溢出，达到攻击目的。每次溢出情况如下：

1. 第一次缓冲区溢出，将栈中的变量覆盖为：

|  |  |
| --- | --- |
| buf | ‘A’ |
| new\_value | 41414141 |
| old\_value | 41414141 |
| limit | 9 |
| choose | 1 |
| p | p->pmf1->mf1->f1变为p->pmf1->foo()->systemaddr() |
| q | q->pmf3->mf3->systemaddr() |

选择器中choose==1,pmf1->foo(),程序执行完\*p=\*q后，会把pmf3的值给pmf1，此时pmf1->foo指向systemaddr函数。

1. 以第一次缓冲区溢出相同参数进行第二次缓冲区溢出，选择器中choose==1,pmf1->systermaddr()，执行地址泄露程序，泄露system函数地址给攻击程序attact。
2. 第三次缓冲区溢出，将栈中的变量覆盖为：

|  |  |
| --- | --- |
| buf | ‘A’ |
| new\_value | system低32位地址 |
| old\_value | system高32位地址 |
| limit | 9 |
| choose | 2 |
| p | p->mf3低字节 |
| q | q->old\_value |

选择器中根据choose==2，pmf2->foo()，执行f2（）。

\*p = \*q使得mf3位置的低32位地址变为&system的低32位地址，从而将system地址低字节保存在函数符号表f3低字节处。

1. 第四次缓冲区溢出，将栈中的变量覆盖为：

|  |  |
| --- | --- |
| buf | ‘A’ |
| new\_value | system低32位地址 |
| old\_value | system高32位地址 |
| limit | 9 |
| choose | 2 |
| p | p->mf3高字节 |
| q | q->new\_value |

选择器中根据choose==2，pmf2->foo()，执行f2（）

\*p = \*q使得mf3位置的高32位地址变为&system的高32位地址。从而将system地址的高字节保存在函数符号表f3高字节处，此时pmf3->foo()为system函数。

1. 第五次缓冲区溢出，将栈中的变量覆盖为：，

|  |  |
| --- | --- |
| buf | ‘AAAAAAAAA’ |
| new\_value | system低32位地址 |
| old\_value | system高32位地址 |
| limit | 0 |
| choose | 3 |
| p | p->pmf2->mf2->f2() |
| q | q->mf3->mf3->system() |

shellcode = "/bin/sh -c "cat ./input/key.txt>./output/dop\_attack\_output.txt""（把参数命令通过溢出保存在shellcode中）。choose = 3，执行函数set\_string(&shellcode),使rdi寄存器址为shellcode地址。

\*p = \*q = &system,使得\*pmf2 = &system。

由于limit = 0，漏洞函数跳出循环，执行pmf2->foo()，而foo指针已修改为system函数的地址，实际执行的是system函数，其参数保存在rdi寄存器中，也就是指向shellcode的指针，即system（"ls"）输出当前文件夹下的文件名