



中国科学技术大学
University of Science and Technology of China

实验2 格式化字符串漏洞

中国科学技术大学 计算机学院

曾凡平

实验环境

- ubuntu Linux虚拟机
 - 特别注意：
 - 不同版本的ubuntu系统(如9.x和14.x)和不同版本的gcc编译出来的可执行文件在内存分配上有细微的差异，本实验的结果是在ubuntu14下得到的。
 - 如果是其它版本的系统，需要修改格式化字符串中的**格式字符(以%开头的串，如%08x、%n、%s)**的个数。

1、格式化字符串攻击原理

Linux IA32下的进程映像

低地址						高地址
.text	.data	.bss	Heap	未使用	Stack	环境

- 其中：
- ——堆(Heap)：用于在程序中进行动态内存申请，如常用的malloc/new函数。
- ——栈(Stack)：函数中的局部变量以及在函数调用过程中产生的临时变量等。

printf()调用过程

- 以下面代码为例：
- `printf("A is %d and is at %08x, B is %u and is at %08x.\n",A,&A,B,&B)`
- 参量被逆序压栈，最后是格式化字符串的地址，`printf()`每次遍历格式化字符串中的一个字符，如果该字符不是格式化参数的首字符（由百分号指定），则复制输入该字符，若遇到一个格式化参数，就采取相应的动作，将栈中的变量`pop()`，与该参数对应。栈指针下移。
- 重点：若格式化参数个数>参量个数，`printf()`会从栈的当前指针开始，依次向下打印。

```
printf("A is %d and is at %08x, B is %u and is at %08x.\n",A,&A,B,&B)
```

格式化字符串的地址

A的值

A的地址

B的值

B的地址

.

.

.

栈底

- 常用的格式化字符有：
- —%s：打印地址对应的字符串
- —%n：对该printf()前面已输出的字符计数，将数值存入当前栈指针指向的栈单元存储的地址中。
- —%m.nx：十六进制打印，宽度为m，精度为n，在m前加0处理为左对齐。

攻击原理

- 格式化字符串攻击原理是利用格式化函数（如 `printf()`）的沿着堆栈指针向下打印的特性，通过**只提供格式化字符串但不提供对应的变量**，读取栈内空间的内容。更进一步，通过将某个要攻击的目标地址放入栈中，就可以利用格式化字符串读写里面的值。因此，它的攻击分为两步：
 - (1) 第一步，将目标地址放入栈中；
 - (2) 第二步，设计格式化字符串，读写目标地址里的值。

实验代码及分析

- vul_prog.c
- 定义变量user_input/*secret/int_input/a/b/c/d，其中，* secret 动态分配两个 int 空间，并用常量 SECRET1/SECRET2赋值， user_input 和 int_put 由用户输入， a/b/c/d未被初始化。输出程序中某些变量的地址，比如secret[1]的地址，它也是下述被攻击的目标地址。
- (1) 输入： int变量 int_input， 字符串user_input
- (2) 打印： user_input以及secret的变量地址和值

2. 任务1

- (1) 使程序崩溃
- 原理：设计包含%s或%n的格式化字符串，使其对应的栈中地址无效，运行结果出现段错误(segmentation fault)，程序崩溃。
- 编译并运行vul_prog.c，输入字符串时在适当的位置放置%s或%n，如：
- **%08x.%08x.%08x.%08x.%08x.%s**

(2) 读取指定变量secret[1]的值

- 原理：将目标变量的地址放入栈中，再设计包含%s的格式化字符串读取该地址中的内容。
 - 过程：首先，需要将原本在堆中的secret[1]的地址放入栈中。利用变量int_input，其位于栈中且值可任意输入。将secret[1]（目标变量）的地址转换成十进制，再通过int_input输入该十进制，便可以将该地址存在int_input对应的栈单元中，当print()打印格式化字符串时，采用%s能够读出int_input栈单元内的地址的值，目的达成。
 - secret[1]'s address is **0x 804b01c (on heap)**
 - Please enter a decimal integer
 - **134524956** 注：0x 804b01c 的十进制值
 - Please enter a string
 - **%08x.%08x.%08x.%08x.%s.%08x**
 - bf9588f8.00f0b2dd.bf958a04.0804b018.**U**.78383025

(3) 修改secret[1]为指定值

- 原理：将目标地址放入栈中之后，利用%m.n的格式，通过设定宽度和精度，控制%n的计数值，计数值就是写入secret[1]的指定值。
 - 过程：以修改secret[1]=0x1234为例。首先，将0x1234换算成十进制为4660,说明%n得到计数值为4660，即在%n之前共有4660个字符被打印。根据前面观察到的地址，在int_input地址之前，出现了4个32位的地址(%08x)，每个地址对应8个字符，选择最后一个位置采用%m.n的打印格式来增加字符，接下来是计算m/n的值，因为前面已经出现了3*9=27字符（加上‘.’），4660-27=4633。令n=4633，故最后一个位置写为%.4633u（或%4633x等），其后跟%n。
 - Please enter a string **#输入secret[1]的地址对应的十进制数**
 - %08x.%08x.%08x.%**4633**x%n

3. 任务2

- 如果第一个scanf语句(`scanf(“%d”, int input)`) 不存在, 即程序不让你输入一个整数, 对于已经实现地址随机化的操作系统而言, 任务1 中的攻击将会变得困难。注意每次运行程序, 得到不一样的地址! 地址随机化的引入使得许多攻击变得困难, 如缓冲区溢出、格式化字符串等。可以使用以下命令关闭地址随机化 (root权限下运行):
- `sysctl -w kernel.randomize_va_space=0`
- 关闭地址随机化后, 删除漏洞程序里的第一个scanf语句(**`scanf(“%d”, &int_input)`**), 重复任务1 里的攻击。步骤依然是先将目标地址放到栈中, 再通过**`%s(或%n)`**读(或写)目标地址中的值。


- Please enter a decimal integer
- Please enter a string
- %08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x
- bffffef78.00f0b2dd.bffff084.bffff024.0804b018.**78383025.3830252e.30252e78.252e7838**
- 可以发现，在secret地址之后是user_input的字符串的ASCII码对应的十六进制(0x78383025="%08x ").根据这一信息，解决方案就是将目标地址作为user_input的一部分放入栈空间中。

如何让scanf接受任意数字？

- 通常，scanf()会将键盘输入的字符转换成ASCII码再存入，比如输入字符5会存为0x35。若直接通过键盘输入，则需要将地址根据ASCII码反转换成键盘可输入的字符，比如0x31323231的键盘输入是1221。然而问题是，ASCII码表中只有128个字符，且0x80之后没有对应的字符，因此无法从键盘输入任意整数地址，故要解决的问题是：
- **如何让scanf接受任意数字？**
- 解决办法是使用文件。我们可以很容易地写一个C 程序 (write_string.c)将0x05（不是“5”）存入一个文件（如mystring），然后运行输入被重定向到mystring的漏洞程序（vul_prog）。换句话说，运行vul_prog < mystring. 这样，scanf将从文件mystring中获得输入而不是键盘。

write_string.c

- write_string.c 将一个格式化字符串写入了一个叫 mystring 的文件，前4 个字节由任意你想放入格式化字符串的数字构成，接下来的字节由键盘输入。
- 将要攻击的目标地址赋值给 write_string.c 中的 address 指针，编译 write_string.c，得到执行文件 write_string，运行 ./write_string，输入只包含 %x 的格式化字符串 user_input。如：
- %08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x

- 运行命令 `./vul_prog < mystring`，栈中内容如下：
- `secret[0]`'s address is `0x 804b018` (on heap)
- `secret[1]`'s address is `0x 804b01c` (on heap)
- Please enter a decimal integer
- Please enter a string
-  `bffffef78.00f0b2dd.bffff084.bffff024.0804b018.0804b01c`
`.78383025.3830252e.30252e78.252e7838`
- 可知已将目标地址放入了栈中，后面再采用与任务1中相同的方法，利用 `%n` 修改 `secret[1]`（目标地址）的值。即可完成攻击

过关测试和能力提升

1. 将secret[0]的值改为0x2345，可尝试不借助变量输入（即将scanf("%d", int input)语句注释再利用格式化字符串修改）。
2. 借助文件输入，将secret[1]的值改为0x3456。

能力提升(自修、不测试)

- 如果要修改的值大于0xFFFF，应如何修改write_string.c。