**CLASS 3**

**Format String Lab**

**吴瑞欣-E41614059**

1. **基础知识**
2. **What is format string?**

没用文档里的原文，用自己的理解就是，输出时该语言所规定输出格式.

1. **The Stack and Format Strings**

以printf为例子，表述字符串入栈时的情景。

printf("a has value %d,b value is %d,c is at address:%08x\n",a,b, &c);

栈内c的地址先入栈，然后b，a

1. **What if there is a miss-match**

**Can this program pass the compiler?**

– The function printf() is defined as function with variable length of arguments. Therefore, by looking at the number of arguments, everything looks fine.

– To find the miss-match, compilers needs to understand how printf() works and what the meaning of the format string is. However, compilers usually do not do this kind of analysis.

– Sometimes, the format string is not a constant string, it is generated during the execution of the program. Therefore, there is no way for the compiler to find the miss-match in this case.

**Can printf() detect the miss-match?**

– The function printf() fetches the arguments from the stack. If the format string needs 3 arguments, it will fetch 3 data items from the stack. Unless the stack is marked with a boundary,

printf() does not know that it runs out of the arguments that are provided to it.

printf() will continue fetching data from the stack. In a miss-match case, it will fetch some data that do not belong to this function call.

**翻译如下：**

**这个程序可以通过编译器吗？**

- 函数printf（）定义为具有可变长度参数的函数。因此，通过查看参数的数量，一切看起来都很好。

- 为了找到错过的匹配，编译器需要了解printf（）的工作原理以及格式字符串的含义。但是，编译器通常不会进行这种分析。

- 有时，格式字符串不是常量字符串，它是在执行期间生成的程序。因此，在这种情况下，编译器无法找到未匹配。

**printf（）可以检测到错过匹配吗？**

- 函数printf（）从堆栈中获取参数。如果格式字符串需要3个参数，它将从堆栈中获取3个数据项。除非堆栈标有边界，

printf（）不知道它用完了提供给它的参数。

printf（）将继续从堆栈中获取数据。在一个错过匹配的情况下，它将获取一些不属于此函数调用的数据。

结论：从上述文档内容可以看出，编译器为了效率没有对printf的工作原理以及格式字符串的含义进行分析；而printf函数的机制并未对miss-match这种情况做出有效检测，因而，格式化字符串存在漏洞。

1. **正文**

**Task1：**

**0、要求：**

• Crash the program.

• Print out the secret[0] value.

• Modify the secret[1] value.

• Modify the secret[1] value to a pre-determined value.

1、崩溃程序。

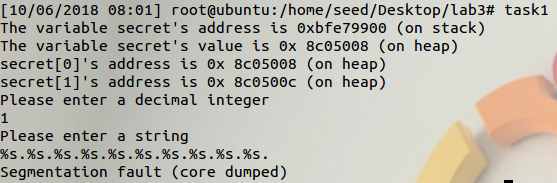
2、打印出secret [0]值。

3、修改secret [1]值。

4、将secret [1]值修改为预定值。

**1、崩溃程序：**

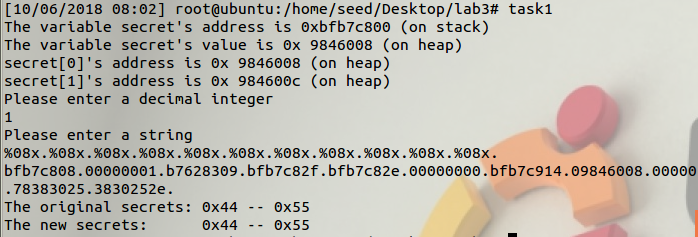
输入字符串:%s.%s.%s.%s.%s.%s.%s.%s.%s.%s.%s.



**2、打印出secret [0]值：**

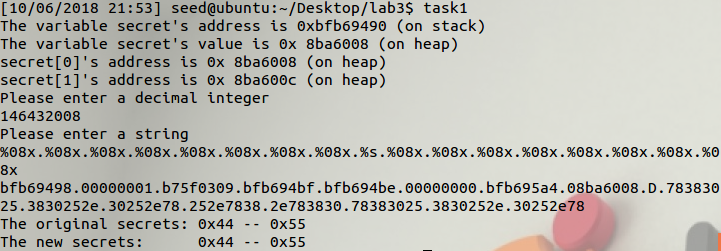
输入字符串：%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.

%08x.%08x.%08x.%08x.%08x.%08x.测试secret[0]与secret[1]的位置。

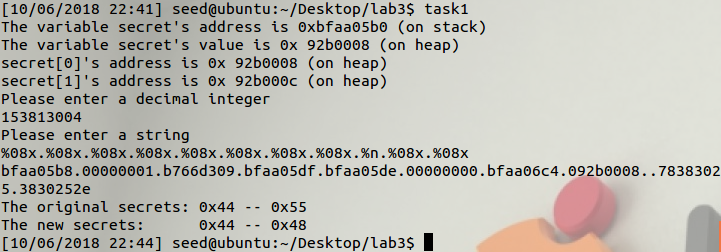


输入字符串：%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.%08x.

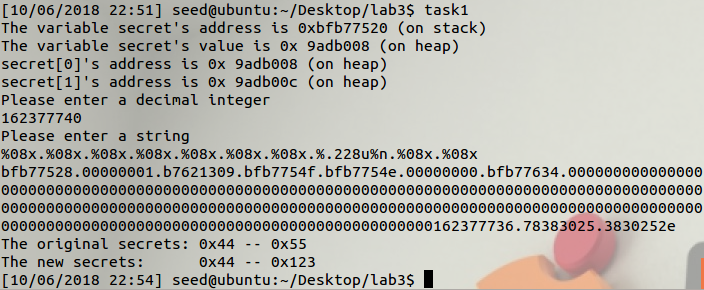
%08x.%08x.%08x.%08x.%08x.%08x.



1. **修改secret[1]的值**



1. **将secret [1]值修改为预定值。**



**Task2：**

1. 运行write\_string.c，传入数据%08x.%08x.%08x.%08x.%08x.%08x.

%08x.%08x.%219u%n.%08x.%08x得到mystring

1. 删除scanf("%d", &int\_input);重新执行

3、我的mystring默认权限不够，将其加权，或者用umask修改默认权限

4、执行sudo task1 < mystring

