一种格式化字符串脆弱性动态检测技术

匡春光，王春雷，何蓉晖
（北京系统工程研究所 信息系统安全技术重点实验室，北京 100101）

摘 要：为了提高计算机软件的安全性，对 C 程序中的格式化字符串脆弱性的原理、特征进行了分析，在分析的基础上提出了一种动态检测技术. 利用此检测技术实现的一种格式化字符串脆弱性检测工具能较准确地检测到 C 目标程序中的格式化字符串脆弱性. 分析结果对编写更安全的 C 程序具有参考价值，检测技术具有实用价值.

关键词：格式化字符串；脆弱性；动态检测；原理；特征；目标程序
中图分类号：TP309.2 文献标识码：A 文章编号：1000－7180(2012)02－0107－04

A Dynamic Detection Technique to Format String Vulnerability

KUANG Chun-guang, WANG Chun-lei, HE Rong-hui
(National Key Laboratory of Science and Technology on Information System Security,
Beijing Institute of System Engineering, Beijing 100101, China)

Abstract: In order to improve the security of computer software, the principle and the feature of format string vulnerability in C programs are analyzed. Based on the analysis, a dynamic detection technique is advanced. By taking use of the technique, a detection tool is produced. The tool can efficiently detect format string vulnerabilities in C binary programs. The analysis result can be a reference to make more secure C programs, and the detection technique is practical.

Key words: format string; vulnerability; dynamic detection; principle; feature; binary program

http://www.kuaishu.com

1 引言

C 程序中的格式化字符串脆弱性很常见而且危害很大. 格式化字符串脆弱性可能导致信息泄露，还可能导致用户名、口令等重要信息被改写，更严重的是，还可能通过改写函数的返回地址、全局偏移表 GOT(Global Offset Table)中的值、异常处理器的指针等使系统执行恶意代码. Tsai T 等人提出通过检测返回地址和帧指针是否被覆盖来检测格式化字符串脆弱性[1]，但该技术不能阻止信息泄露. Cowan C 等人提出了一种通过替换输出类函数的检测方法[2]，该方法的误报率比较高. Shankar U 等提出了一种基于类型系统的编译时检测技术[3]，这种技术的最大缺点是需要源代码. Ringenburg F 等人提出的 white-listing 技术检测部分重要内存是否被改写[4]，同样，该技术也不能阻止信息泄露. 针对产生格式化字符串脆弱性的根本原因，本文提出了一种格式化字符串脆弱性动态检测技术.

2 格式化字符串脆弱性的原理、特征

C 程序中的格式化字符串脆弱性是由于不正确使用 C 的输出类标准库函数[v][f]printf()、[v]s[n]printf()和 syslog()引起的[5-6]. 这些函数都可以有格式化字符串参数和变量参数，格式化字符串是其中可包含特殊格式化标识的字符序列，变量参数与特殊格式化标识对应[7]. 特殊格式化标识主要包括%d、%o、%x、%u、%c、%s、%f、%e 和%g，它们用于指定被输出的变量的输出格式；特殊格式化标识还包括%n，%n 的作用是把在%n 之前已输出的字符数写到内存中，此内存的地址由与%n 对应的变量指定.

下面以 prinf()为例说明格式化字符串脆弱性

收稿日期：2011－06－29；修回日期：2011－08－01

的原理.

prinf( ) 的一种正确的使用形式为：prinf ("string＝%s", str)，其中"string＝%s"为格式化字符串参数，str 为变量参数，格式化字符串中的特殊格式化标识和后面的变量参数在个数和类型上正确对应. 系统调用 prinf 时内存的情况类似图 1，系统调用 prinf 前会把所有参数按倒序压入栈中(栈是内存的一部分，延伸方向自高向低)，在此例中，系统先把 str 的地址压入栈中[8]，然后把格式化字符串的地址压入栈中，str 和格式化字符串也保存在内存中，但在较高地址处，如果 str 的值为"abc"，以上语句将输出"string＝abc".

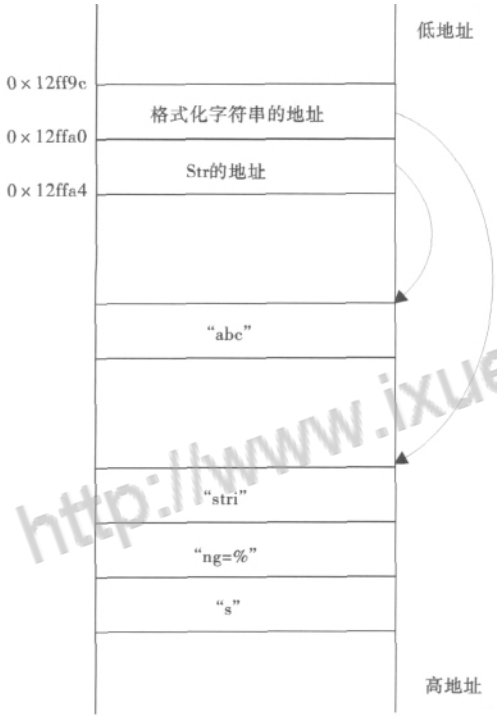

图 1 正确调用时内存情况

不正确的使用形式有两种. 这两种形式分别导致内存信息泄露和内存信息被改写.

2.1 泄露内存信息

第一种不正确的使用形式是格式化字符串中的特殊格式化标识的个数多于变量参数的个数，但多出的特殊格式化标识不包括%n. 假定程序中包括如下语句序列：

```
fgets(str,512,stdin);
prinf(str);
```

程序运行到 fgets(str,512,stdin)时，如果用户输入"abc %x"，其下的 prinf(str)实际上是 prinf("abc %x")，此语句中的特殊格式化标识为一个，变量参数为 0 个. 系统调用 prinf 前只会把格式化字符串的地址压入栈中，此语句在输出"abc"后，尽管栈中实际上没有和%x 对应的变量，但系统还是假设有相应的变量，系统会把栈中的下一个值作为假设中的变量参数，因此系统会把栈中的下一个值以 16 进制的形式输出. 设系统调用 prinf 时内存的情况如图 2 所示，保存格式化字符串地址内存的地址为 0x12ffa0，系统将把地址为 0x12ffa4 的内存中的值 0x12345 以 16 进制的形式输出. 如果用户输入更多特殊格式化标识，系统将继续输出内存中更多的值，这样就达到了泄露内存信息的目的.

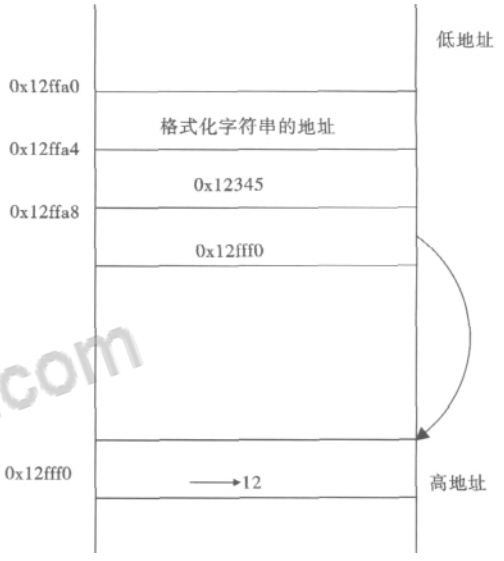

图 2 错误调用时内存情况 1

2.2 改写内存信息

第二种不正确的使用形式是格式化字符串中的特殊格式化标识的个数多于变量参数的个数，而且多出的特殊格式化标识包括%n. 在以上例子中，程序运行到 fgets(str,512,stdin)时，如果用户输入"\x01\x01\x01\x01 %07x%n"，其下的 prinf(str)实际上是 prinf("\x01\x01\x01\x01 %07x%n")，此语句中的特殊格式化标识为 2 个，变量参数为 0 个. 此语句输出 4 个 1 和 1 个空格后，尽管栈中实际上没有和%07x 对应的变量，但系统还是假设有相应的变量，系统会把栈中的下一个值作为假设中的变量参数，因此系统会把栈中的下一个值以 16 进制的形式输出. 设系统调用 prinf 时内存的情况如图 2 所示，保存格式化字符串的地址的内存的地址为 0x12ffa0，系统将输出地址为 0x12ffa4 的内存中的值 0x12345. 同样，尽管栈中实际上没有和%n 对应的变量，但系统还是假设有相应的变量，系统会把栈中的下一个值作为假设中的变量参数，因此系统会把栈中的下下一个值作为地址，然后往此地址处写

http://www.ixueshu.com

入 12(因为此时此语句已输出“1111 0012345”，即已输出 12 个字符). 如图 2 所示，系统会把地址为 0x12ffa8 的内存中的内容 0x12fff0 作为地址，然后往 0x12fff0 处写入 12，这样就达到了改写内存的目的.

2.3　执行恶意代码

结合两种不正确的使用形式可以达到执行恶意代码的目的. 在以上例子中，程序运行到 fgets(str，512，stdin)时，如果用户输入很多%x，即“%x%x%x…”，用户可以了解到内存中的很多信息，设系统调用 prinf 时内存的情况如图 3 所示，用户输入 16 个或多于 16 个%x 后，就可以了解到 0x12ffe0 处保存的信息. 设 0x12ffe0 处保存了某函数的返回地址，重新运行程序，当程序运行到 fgets(str，512，stdin)时，如果用户输入“\xe0\xff\x12\x00AAAA”后，再输入很多%x，即输入“\xe0\xff\x12\x00AAAA%x%x%x…”，用户可以通过其中的%x 输出此格式化字符串本身，依据第几个%x 输出了格式化字符串本身的前四个字符，可了解到保存此格式化字符串的内存与栈顶的距离(32)，有趣的是，此时格式化字符串本身也被当作参数使用了. 再次运行程序，当程序运行到 fgets(str，512，stdin)时，如果用户输入“\xe0\xff\x12\x00AAAA”后，再输入 7 个%5000x 和一个%n，即输入“\xe0\xff\x12\x00AAAA%5000x%5000x%5000x%5000x%5000x%5000x%5000x%n”，就可以改写 0x12ffe0 处的内容为 35 008. 因为此格式化字符串中的第 8 个特殊格式化标识为“%n”，系统假设第 8 个变量参数应保存在 0x12ffc0 处，此时 0x12ffc0 处保存的是 0x12ffe0(因为 little_endian 的原因，所以输入的是“\xe0\xff\x12\x00”)，因此系统会改写 0x12ffe0 处的内容；到%n 时，此语句已输出 8＋5 000＊7＝35 008个字符，因此，0x12ffe0 处的内容被改写为 35 008. 如果想把内容改为别的值，还可以相应地修改各个%5000x 中的 5000 为别的值，且这 7 个值可以不同，被改写后的内容一般被设计为恶意代码的起始地址. 因为 0x12ffe0 处原来保存了某函数的返回地址，系统执行相应的函数后，会把新的值 35 008 依然当作函数的返回地址对待，这样，系统就会转移到 35 008 处，即恶意代码的起始地址处，这就意味着，系统将执行恶意代码.

3　检测技术

以上分析的两种不正确的使用形式的一个共同特点是格式化字符串中的特殊格式化标识的个数多于变量参数的个数. 因此，产生格式化字符串脆弱性的根本原因只有一个，就是特殊格式化标识的个数多于变量参数的个数. 如果格式化字符串可由用户输入，则肯定存在问题，如果格式化字符串不能由用户输入，而是一个常量，则要比较特殊格式化标识的个数和变量参数的个数. 针对产生格式化字符串脆弱性的根本原因，提出了一种格式化字符串脆弱性动态检测技术.

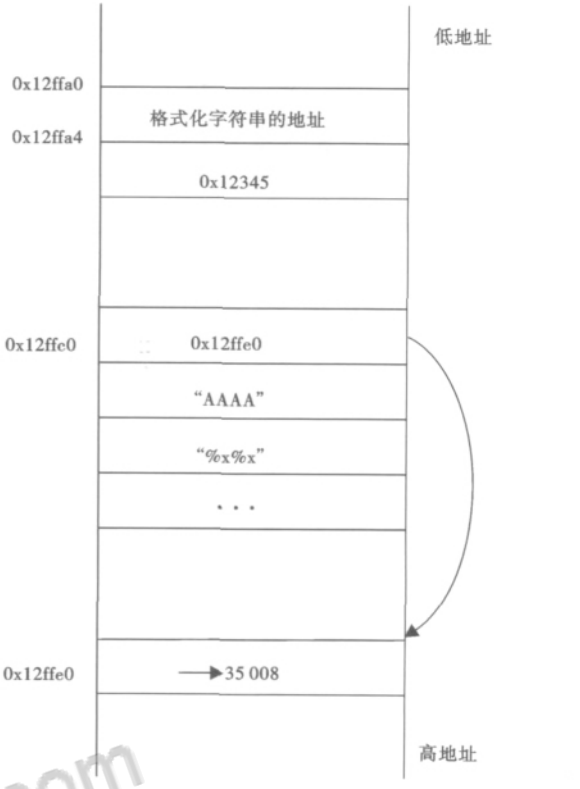

图 3　错误调用时内存情况 2

该检测技术是在程序执行过程中捕捉输出类标准库函数[v][f]printf()、[v]s[n]printf()和 syslog()，并获取它们的格式化字符串参数的值和变量参数的个数，通过格式化字符串参数的值分析其中含特殊格式化标识的个数，比较特殊格式化标识的个数和变量参数的个数，分析是否存在格式化字符串脆弱性.

该检测技术的实现借助了插桩工具 Pin，在插桩工具 Pin 环境下实现了一个检测工具，具体检测过程如下：

(1) 通过 Pin 启动需要进行脆弱性分析的目标程序，使该目标程序运行在 Pin 的插桩环境中.

(2) 通过检测工具动态跟踪目标程序执行过程中调用的输出类标准库函数[v][f]printf()、[v]s

http://www.ixueshu.com

[n]printf()和 syslog().

(3) 获取被跟踪到的函数的格式化字符串参数的值和变量参数的个数.

(4) 通过格式化字符串参数的值分析其中含特殊格式化标识的个数.

(5) 比较特殊格式化标识的个数和变量参数的个数,分析是否存在格式化字符串脆弱性.

(6) 目标程序执行结束后,退出分析流程.

(7) 输出被分析的目标程序中存在的格式化字符串脆弱性,指出脆弱性在目标程序中的位置及相关函数.

4 实验结果

利用格式化字符串脆弱性检测工具对以下源代码对应的目标代码进行分析.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(int argc,char *argv[])
4 {
5 int x=5;
6 int y=10;
7 int z=15;
8 char *buf=" mnopq" ;
9 char *var1="\x21\x21\x21\x21";
10 char *var2=" %12x\n";
11 char var[80];
12 char *h_buf;
13 printf(" x=0x%x,y=0x%x,z=0x%x,buf=%%%s\n",x,y,z,buf);
14 strncpy(var,var1,20);
15 strcat(var,var2,15);
16 printf(var);
17 h_buf=malloc(80);
18 sprintf(h_buf," x=0x%x,y=0x%x\n",x);
19 printf(h_buf);
20 printf(argv[1]);
21 printf("\n");
22 return 0;
23}
```

分析得到如下结果.

```
(0x401000:_main)(0x401075)->(0x40138e:_printf)
(0x401000:_main)(0x401097)->(0x4010cd:_sprintf)
(0x401000:_main)(0x4010b2)->(0x40138e:_printf)
```

对照相应的汇编代码可知,分析结果表明源代码中的第 16 行的 printf、第 18 行的 sprintf 和第 20 行的 printf 中存在格式化字符串脆弱性. 分析结果和实际情况一致.

5 结束语

本文分析了 C 程序中的格式化字符串脆弱性的原理、特征,并通过分析总结出产生格式化字符串脆弱性的根本原因是格式化字符串中特殊格式化标识的个数多于变量参数的个数;然后依据产生格式化字符串脆弱性的根本原因提出了一种动态检测技术;最后给出了实验结果,实验结果表明该检测技术是有效的.

参考文献:

[1] Tsai T, Singh N. Libsafe 2.0: detection of format string vulnerability exploits [M]. White Paper Version 3-21-01, Avaya Labs, Avaya Inc, 2001.

[2] Cowan C, Barringer M, Beattie S. FormatGuard: automatic protection from printf format string vulnerabilities [C]// proceedings of the lOth USENIX Security Symposium. Washington, DC, USA: IEEE, 2001.

[3] Shankar U, Talwar K, Foster J S. Detecting format string vulnerabilities with type qualifiers [C] // proceedings of the lOth USENIX Security Symposium. Berkeley,CA,USA:USENIX Association,2001.

[4] Ringenburg F, Grossman Dan. Preventing format-string attacks via automatic and efficient dynamic checking [C]// proceedings of the 12th ACM Conference on Computer and Communications Security. New York,NY,USA:ACM,2005.

[5] 马富达, 蔡皖东. 基于嵌入微 LINUX 的开机安全认证系统 [J]. 微电子学与计算机, 2010, 27(7): 181-184.

[6] John Viega, Bloch J T, Tadayoshi Kohno. ITS4: A static vulnerability scanner for C and C++code [C] // proceedings of Annual Computel Security Applications Conference. New or leans, LA:IEEE,2000.

[7] 程铃. MANET 入侵检测技术的研究 [J]. 微电子学与计算机, 2010, 27(6): 64-67.

[8] Dzintars Avots, Michael Dalton, Benjamin Livshits. Improving software security through a C pointer analysis [C]// proceedings of the 27th International Conference on Software Engineering. 2005.

作者简介:

匡春光 女,(1971-),副研究员,硕士. 研究方向为计算机软件脆弱性.

http://www.ixueshu.com

免费论文查重：http://www.paperyy.com
3亿免费文献下载：http://www.ixueshu.com
超值论文自动降重：http://www.paperyy.com/reduce_repetition
PPT免费模版下载：http://ppt.ixueshu.com

阅读此文的还阅读了：

1. 车轮扁疤动态检测技术
2. 一种船舶舱室人员动态检测及跟踪方法设计
3. 一种高效的动态历史拟合技术
4. 铁路信号动态检测技术分析讨论
5. 动态·检测
6. 中国农村家庭贫困脆弱性动态研究
7. 一种检测存活心肌的新方法——彩色室壁动态分析技术
8. 浅谈铁路信号动态检测技术
9. 一种新型织物动态吸湿检测装置的研制
10. 一种动态自动称量技术
11. RTK终端实时动态检测技术研究
12. 浅谈铁路信号动态检测技术
13. 基于动态检测技术的智能ETC车道系统
14. 信号动态检测技术的探讨
15. 一种shellcode动态检测与分析技术
16. 一种分布式脆弱性检测技术的研究
17. 动态视频目标检测技术
18. 动态·检测
19. 矿山储量动态检测技术的应用分析
20. “和”是一种动态运作
21. 计算机安全漏洞动态检测技术
22. 一种基于动态注入的DLL木马检测技术
23. 一种提高电缆耐压检测效率的技术
24. 矿山储量动态检测技术的应用研究
25. 一种检测动物弓形虫病的新型动态免疫胶体金层析技术的建立

26. 一种虚拟化环境的脆弱性检测方法

27. 一种改进的动态电压凹陷实时检测技术研究

28. 软件设计中安全漏洞动态检测技术的探究

29. 一种新型动态称重技术的特点及应用

30. 动态·检测

31. 一种线路动态张力检测装置结构设计

32. 经颅多普勒全程动态检测技术

33. 德国接触网动态检测技术

34. 检测技术及动态硬度计发展应用

35. 基于动态光散射技术的地沟油检测

36. 一种无色渗透检测新技术

37. 一种多方通话动态控制技术

38. 一种基于动态污点的内存越界访问检测框架

39. Windows平台下的格式化字符串漏洞利用技术

40. 一种动态视力检测装置

41. 格式化字符串攻击检测算法

42. 动态·检测

43. 中国金融脆弱性的状况及研究动态

44. 动态·检测

45. 一种电缆在线检测技术

46. 一种采集终端检测的技术与方法

47. 一种格式化字符串脆弱性动态检测技术

48. Idid:一种基于免疫的动态入侵检测模型

49. 基于C源码的脆弱性检测技术研究

50. 动态·检测