# 关于缓冲区溢出漏洞的解决方法①

丁永尚[1,2]　何福男[2](1.苏州大学 计算机科学与技术学院 江苏 苏州 215104;
2.苏州工业职业技术学院 江苏 苏州 215104)

**摘　要：** 为了免去以往手工检查源程序缓冲区溢出漏洞的繁琐和不全面性，利用 LINUX 下的两个重要软件 LEX 和 YACC，编写 C 语言的词法和语法分析程序，并在其中添加相关语句的语意动作代码，以达到在对 C 源程序进行语法分析的同时，输出里边的函数调用关系的目的。此外，所生成的函数关系调用表将被放到数据库里边，进行列表、查询和统计；以便进行程序的维护。

**关键词：** 安全；缓冲区溢出；编译；LEX YACC

## Solutions to Buffer Overflow Leak

DING Yong-Shang[1,2] (Suzhou Institute of Industrial Technology, Soochow University, Suzhou 215104, China)
HE Fu-Nan (Suzhou Institute of Industrial Technology, Suzhou 215104, China)

**Abstract:** To avoid the overload with details of the manual audit, this paper builds a compiler of C by using two important tools LEX and YACC of UNIX. It adds ourown code to output the relation of function invoked or invoking, while processing the lexical analysis and syntax analysis. Furthermore, the relation table will be put in the database so that it can be listed, queried and counted to maintain the source code.

**Keywords:** safety; buffer overflow; compile; LEX; YACC

## 1　引论

随着电子商务的普及，安全性不再是一件微不足道的事情，它已经开始成为一个重要的问题；不够安全的软件将会导致大公司的衰落或者消亡，这只是时间问题。其中缓冲区溢出是最大的安全问题之一。

根据有关资料显示，当今被广泛利用的薄弱环节中多达百分之五十都是缓冲区溢出。而且，该分析指出这一比例正随着时间而不断上升。在 1998 年 Lincoln 实验室用来评估入侵检测的的 5 种远程攻击中，有 2 种是缓冲区溢出。由于某些原因，开发人员尚未积极推动消除缓冲区溢出这一软件安全性的主要陷阱。

其实引起灾难的方法总是惊人地简单。一部分采用错误的语言设计(通常是 C 和 C++)，再结合程序员拙劣的编写手段，就会发生这种大问题。尽管象 Java 这样不具备某些令人难以置信的异常编程方式的、现代“安全”的语言避免了缓冲区溢出，但缓冲区溢出问题也会发生在非 C 和 C++ 的语言。

引起缓冲区溢出问题的根本原因是 C(与其后代 C++)本质就是非安全的。没有边界来检查数组和指针的引用，也就是开发人员必须检查边界(而这一行为往往会被忽视)，否则会有遇到安全问题的风险。标准 C 库中存在许多非安全字符串操作函数，包括：strcpy()、strcat()、sprintf()、gets()、scanf()、sscanf()、fscanf()、vfscanf()、vsprintf ()、vscanf()、vsscanf()、streadd()、strecpy()、strtrns()等。

为什么缓冲区溢出是安全性问题呢?

我们可能会想：“有什么大不了，一点点水的溢出根本不会对任何人造成伤害。”那么将我们的类推再引深一点，试想水溢出在有许多暴露在外的电线的工作台上。根据水滴溅的位置，火花会四处飞散。同样地，当缓冲区溢出时，额外的数据会摧残程序将来可能要访问的其它有用的数据。有时，这些其它数据的更改会导致安全性问题。

如果读者对缓冲区溢出现象还不以为然的话，那

① 收稿时间:2009-01-19



© 中国科学院软件研究所 http://www.c-s-a.org.cn

么我们就继续试想一种简单的情况。这种情况就是直接在缓冲区后面的内存中分配一个布尔标志。这个标志决定运行程序的用户是否可以访问专用文件。如果有不怀好意的用户覆盖缓冲区，则会更改标志的值，从而使攻击者可以非法访问专用文件。

另外，还有很多情况可以让攻击者利用缓冲区溢出达到攻击的目的。例如通过摧毁堆栈，在 UNIX 机器上，攻击者达到获得一个交互式 shell 的目的；在 Windows 环境里，达到下载一个恶意的特洛伊木马程序到机器上并执行该程序的目的等。

由此可见缓冲区溢出确实是一个很大的安全问题之一，应该引起我们足够的认识，并进一步研究各种解决方法以尽量避免产生安全问题。那么下文我们就探讨解决问题的方法。

## 2 解决问题的原理和可供选择的方案

在具体的实践中，我们解决问题的原理是(在此我们以C代码为例)，用 linux 下的应用软件 lex 和 yacc 生成通用的 c 编译器，此编译器在对一个 c 源文件进行词法和语法的检查的同时，把 c 源程序的函数调用关系导出来，并放进数据库里边以进行列表、查询和统计(数据库这部分用 visual c++开发)；从而发现 c 源程序里是否调用了容易产生缓冲区溢出漏洞的 c 字符串操作库函数；如果有，可以采用合理的替换方法，在完成通用功能的同时，避免直接使用 c 字符串操作库函数。

本课题的目的就是，把源程序里的函数调用关系(尤其是 gets，fprintf 等 c 库函数)找出来；以便对函数调用逐个进行分析以发现易产生缓冲区溢出的地方；然后，采用合理的替换方法，在完成相同功能的同时，避免直接使用没有边界检查的 c 字符串操作库函数。

因此有三个方案可供选择，以达到此目的。它们是：

① 手工分析 c 源程序，找出函数调用关系；

② 直接编写 c 语言词法和语法分析程序，以实现对 c 源程序的扫描并导出函数调用关系；

③ 利用 linux 上的词法生成工具 lex 和语法生成工具 yacc 编写 c 语言编译器，在对 c 源程序进行语法分析的同时导出函数调用关系。

## 3 方案分析和论证

以上三个方案相比较，方案一显然过于原始、笨拙；不仅使程序员的工作量大大加重，事实上手工分析是非常困难的！从根本看来，代码往往是复杂的；分析大型的程序(如 Sendmail，大约有 50,000 行代码)不是一个简单的任务；即便是有耐心的程序员也会使某些问题从其身边溜过；例如，在 1996 年 9 月，在 Sendmail 中发现并修正了几个新的可利用的缓冲区溢出之后，进行了广泛的手工安全性审查。但是，不到四个月，又发现了另一个手工审查时遗漏的可利用的缓冲区溢出。又如，再近一点，Wagner 加利福尼亚大学的一个研究生，在 Sendmail 中找到了一些新的缓冲区溢出；但是，他发现的那些溢出好象并不是可利用的；虽然如此，至少 Wagner 发现的一个溢出是 1996 年以来未被手工审查到的。

方案二中，直接编写 c 语言的词法和语法分析程序是一件很繁琐并且容易出错的工作；另外，即便是编写此软件的程序员本身以后对软件进行维护和修改也将很困难。记得，在本人读大学时，曾对 pasical 语言的一个子集手工编写过词法和语法分析程序；虽然后来成功完成了任务，但是，在编写的过程中出现了很多错误，并且源程序比较长；后来尝试用 lex 对这个语言子集编写词法分析程序，结果，lex 版本的程序只是原来程序的三分之一；我们的经验是程序中的错误数一般与它的长度成正比；如果不用 lex，我们要花三倍的时间来编写和排除错误。

相对以上两个方案，方案三，也就是本设计采用的方案弥补了上边的全部弱点。此方案中我们用到了 lex 和 yacc。Lex 代表 Lexical Analyzar。Yacc 代表 Yet Another Compiler Compiler。 Lex 和 Yacc 是 LINUX 的两种非常重要的、功能强大的工具。它们是特意为编写编译程序和解释程序的人设计的工具，对非编译程序编写人员所感兴趣的许多应用程序也非常有用。在输入中查找模式或者拥有输入或命令语言的任何应用程序都适于采用 lex 和 yacc。而且，它们允许快速应用程序原型设计，容易修改，而且程序的维护简单。Linux 下的 GUN C 编译程序也就是 GCC 就是用 lex 和 yacc 开发出来的。

因此，方案三中，我们只需要搞懂 linux 工具 lex 和 yacc 的工作原理和使用规则，例如常规表达式、声明、匹配模式、变量、BNF 范式、Yacc 语法等；其次搞懂 c 语言的语法结构，然后在相应的识别模式后边添加自己的语意动作就可以了。这样既缩短了编写时间，而且使源程序非常明朗而便于修改和维护。具体本课题所采用的方案三的流程如图 1 所示。



© 中国科学院软件研究所 http://www.c-s-a.org.cn

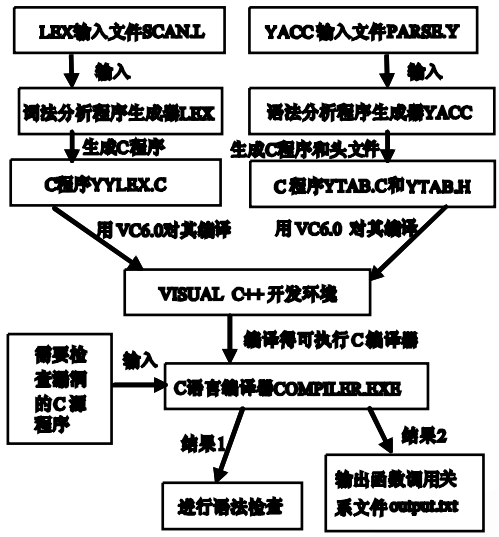


图 1　设计流程图

如图 1 所示，本研究最终得到的软件是 c 编译器 compiler.c。其功能是，对一个 c 源程序进行语法错误检查(把语法正确的代码在终端输出，知道发现语法错误时停止并退出)的同时；输出里边的函数调用关系。

函数调用关系输出在 compiler.exe 生成的两个文本文件里边，即文件 invoke.txt 和 output.txt。其中，文件 invoke.txt 是用英文表达函数调用关系，我们直接可以看懂，是为了方便程序员直接阅读函数调用结果而设计；文件 output.txt 的格式是，每一行输出三个字符串，分别是函数调用者的名字、函数被调用者的名字和被调用的行数，有多少对函数调用就有多少行字符串；此文件是为了数据库读入数据方便而设计，即文件 output.txt 是 C 编译器 compiler 与数据库程序的连接点，所生成的函数关系调用表将被放到数据库里边，进行列表、查询和统计；以便进行程序的维护。从而也免去了手工检查 c 程序的麻烦。用本软件对源程序进行扫描，不仅可以准确全面的发现函数调用关系，而且效率极高。为程序员下一步用安全函数替换有潜在安全问题的函数(没有边界检查的函数，例如 strcpy、strcat、sprintf、gets、scanf 等。)提供极大的便利，从而可以更快捷和高效的检查缓冲区溢出潜在危险。

总之，本设计的成果不仅避免了手工检查大量 c 程序的烦琐和容易出错的问题；而且也不需要直接编写 c 语言词法和语法分析程序，以实现对 c 源程序的扫描并导出函数调用关系。用本软件对源程序进行自动扫描，不仅可以准确全面的发现函数调用关系，而且效率极高。从而更快捷和高效的检查缓冲区溢出潜在危险。

## 4　结语

本次研究的成果将对安全漏洞尤其是缓冲区溢出漏洞的检查起关键性的作用。但是，由于技术和时间上的种种原因，本设计还存在一定的问题。例如，本研究只对 C 源程序进行词法和语法扫描并检查函数调用关系，没有把范围扩充到对 C++源代码的扫描；但是，利用本研究的方法，达到对 C++代码的扫描并不难，尽管 C++的语法规则会复杂很多。

针对以上问题，如果有进一步研究的可能，本人将编写 C++语法的 BNF 范式，把扫描的范围扩充到 C++代码。

## 参考文献

1 Louden KC. Compiler Construction Principles and Practice,北京:机械工业出版社, 2000.
2 Levine JR, Mason T. Lex 与 Yacc.第二版,北京:机械工业出版社, 2003.
3 吕映芝,张素琴,等.编译原理.北京:清华大学出版社, 1998.
4 Stroustrup B, The C++ Programming Language, special edition,北京:高等教育出版社, 2001.
5 贾明,严世贤.Linux 下的 C 编程.北京:人民邮电出版社, 2001.
6 陈雅秀.Linux 指令辞典.北京:中国铁道出版社, 2001.



© 中国科学院软件研究所 http://www.c-s-a.org.cn