移动互联网应用漏洞发掘与检测技术研究

赵　鑫　朱东来　杨　宏
中国移动北京公司　100053

【摘要】在移动互联网时代的今天，信息安全问题已成为业界关注的热点。本文对移动互联网应用中的安全漏洞问题进行了研究，主要针对互联网应用程序中的安全问题，特别是运行于主机与各类终端上的应用程序中可能存在的安全漏洞。本文阐述了移动互联网各类程序中漏洞产生的原理、漏洞的危害，对漏洞检测技术进行了研究，特别对危害极大的缓冲区溢出漏洞进行了详细探讨。本文提出了缓冲区溢出的几种检测方法，即通过进行缓冲区边界检查和指针完整性检查，并辅以攻击数据行为检测和编写正确的代码，来达到检测缓冲区溢出的目的。这几种方法可以有效地检测缓冲区溢出问题，从而避免缓冲区溢出带来的危害。

【关键词】移动互联网应用　信息安全　安全漏洞　漏洞检测

一、引言

随着传统互联网、移动通信技术和应用条件的改进和发展，移动互联网得到迅速发展。基于移动互联网的移动电子商务(M-Commerce)、移动即时通信(MoblieInstant Messaging)、移动社交(Mobile Social NetworkingServices)、手机支付等各种新型的业务开始渗入人们的日常生活。与此同时，移动互联网上终端用户身份的认证、用户敏感信息的保护、移动互联网的安全监管等安全问题愈加突显，成为关注的热点。不仅如此，全球对信息系统的安全性要求越来越高，信息安全作为非传统安全因素，已与各国的政治安全、国防安全、经济安全、文化安全共同成为国家安全的重要组成部分。信息

安全紧密围绕国家安全，为国民经济建设服务，其本质是要解降信息化过程中可能存在的安全威胁，并采用恰当的综合保护措施，克服系统的脆弱性，控制和降低信息安全的风险度，保障信息化的顺利进行。

本文对移动互联网应用中的安全漏洞问题进行了研究，主要针对互联网应用程序中的安全问题，特别是运行于主机与各类终端上的应用程序中可能存在的安全漏洞。本文阐述了移动互联网各类程序中漏洞产生的原理、漏洞的危害，对漏洞检测技术进行研究，特别对危害极大的缓冲区溢出漏洞进行了详细探讨。

一般来说，漏洞（Vulnerability）是指系统或应用程序存在的某种未曾预料到的未授权、不安全状态，它的出现往往是由于某些编程、配置或操作上的缺陷（Flaw）以及一些特定的条件引起的，攻击者往往需要构造条件，利用这些缺陷，使系统或应用程序由正常状态进入该状态，从而破坏正常的安全机制。

缓冲区溢出漏洞利用技术主要是从溢出点与 Shellcode 技术两部分来进行研究。那么，缓冲区溢出漏洞检测技术就可以分别针对这两个方面展开。针对溢出点主要采取的是缓冲区边界检查及指针完整性检查，而攻击数据特征检测则主要针对 Shellcode。

二、缓冲区溢出漏洞原理

缓冲区是程序运行的时候机器内存中的一个连续块，它可以在栈（stack）、堆（heap）或未初始化数据段（bss）中。正常情况下，系统将数据存放在其分配好的一定大小的一段缓冲区内，若存在某些编程错误，使得攻击者可以通过控制该缓冲区或其它缓冲区内的数据，来控制程序流程，从而达到攻击目标，称该漏洞为缓冲区溢出漏洞，称这样的攻击为缓冲区溢出攻击。就目前的发展情况来看，缓冲区溢出攻击一般分为基于栈（stack-based）的缓冲区溢出攻击与基于堆（heap/bss-based）的缓冲区溢出攻击。

在进程的内存空间中，栈（stack）起到了保存有关当前函数调用上下文的容器的作用。许多内容都可能进入栈空间，其中包括：函数的非静态局部变量值、堆栈基址、当函数返回时程序应该跳转到的返回地址以及传递到函数中的参数等，其结构如图 1 所示。

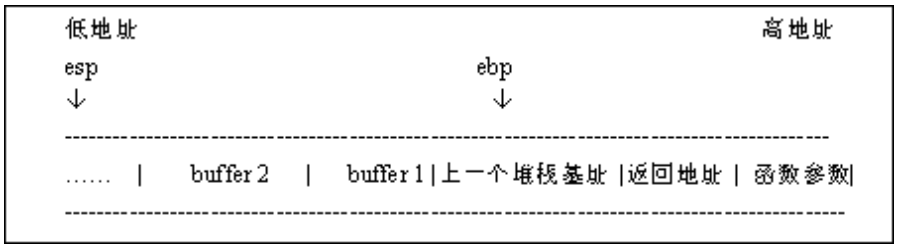

图 1　函数栈空间结构

当程序员编写程序时出现边界检验错误，不限制写入 buffer 中内容的长度时，就会出现由于写入内容过多，导致缓冲区溢出，堆栈结构被破坏。结果，可能引起程序运行失败，产生严重的系统崩溃，也可能由于精心构造的字符串，而使得返回地址被某个设计好的地址覆盖，改变程序运行路径，执行攻击者设置的其它指令。

这种边界检查错误多发生在调用一些字符串或内存拷贝函数时，如在 C/C++语言编写的程序中使用 strcpy、strcat、sprintf、vspritnf、memcpy 等函数，而不限制拷贝字符数目。

三、漏洞发掘技术

目前，漏洞发掘主要采用的方法有黑盒测试，补丁比对，静态分析、动态调试等几种方法。这些技术都不完备，各有其优缺点，在漏洞发掘过程中往往需要相互补充。

安全漏洞属于软件缺陷的一个子集，软件测试技术是漏洞发掘中的一项重要技术。黑盒测试是软件测试技术中的一种重要手段，已经发展了很多年，理论比较成熟。通过测试输入非常规的命令行参数、交互时的输入、环境变量等数据来触发潜在的安全漏洞，成功的标志是导致有溢出漏洞的程序异常退出。在 Linux 系统上通常的出现“段错误”的提示信息，在 Windows 系统下通常弹出出错对话框。不过，黑盒测试无法真正理解程序的流程，而且，在没有产品详细功能介绍的条件下，面对太多的可能情况，黑盒测试很难把目标的问题全部暴露出来。

补丁比对方法对于开源的软件比较方便，通过直接比较补丁文件与源文件代码，就可以很快的确定漏洞代码出现的位置。但对于大多数不开源的软件来说，只能通过反汇编来比较二进制的补丁文件，但往往源程序作些许改动，经过编译得到的汇编程序就会有很大改动，这样从反汇编文件中就很难看出真正关键的改动。目前，补丁比

对的工具比较多，比较常用的有：UltraCampare、examdiff、gvim。但它们对于较大的文件一般都无法处理。

静态分析中，目前发展比较成熟的是基于源码的静态分析，它能检查包括缓冲区漏洞、格式化字符串、竞争条件等常见的安全问题。这些工具的基本原理是将源代码与有漏洞的特征代码进行比对，是建立在文本分析的方法上。其分析结果准确性较低，只能作为参考。而基于反汇编的静态分析技术主要是基于 IDA Pro 的脚步挖掘技术。IDA Pro 强大的反汇编功能对于分析反汇编代码有很大帮助，同时，它还提供 IDC 自动化脚本功能。IDC 是 IDA Pro 提供的类似于 C 语言的脚本语言，程序的开始点同样是 main 函数。IDA Pro 提供给 IDC 的内置函数可以深入到程序的流程和指令的分析，甚至可以直接模拟芯片指令对程序进行模拟计算。

对于动态调试，主要的目标就是跟踪程序流程，查看进程或线程堆栈的利用情况，查看关键函数的调用情况等。大多数的调试器都提供跟踪程序流程的功能，如，在 Windows 下，SoftICE 的，Ollydbg 的，另外一些虚拟机，如 bocus，也提供跟踪程序流程的功能。在 Linux 系统下，主要是利用 ptrace 系统调用。但这些工具的问题在于，无法自定义跟踪，只能不分主次所有的执行语句都记录，这样跟踪数据就会很大，甚至可以说是海量的，从中难找到关键的信息，另外运行速度也是一个很大的问题。

四、缓冲区溢出检测技术

缓冲区溢出漏洞的主要检测方法是缓冲区边界检查和指针完整性检查法。缓冲区边界检查的出发点在于只要缓冲区不能被溢出，溢出攻击也就不可能实现了，所以它通过检测每次对缓冲区的读写操作有没有超过缓冲区边界来防止缓冲区溢出。而指针完整性检查则是检测指针内容有没有被破坏，而不去预先阻止。当检测到指针内容被修改后，再决定如何处理。有些方法是提示出错，有些是做纠错处理，使程序正常继续。这两种方法主要都是基于编译器实现的，核心的思路都是对返回地址或函数指针作一定的处理，来达到目标。

4.1 缓冲区溢出边界检查

缓冲区边界检查中一个比较重要的技术是引用对象（referent-object）技术[1]。这

种技术的主要出发点是任何由界内指针计算得到的地址都与原指针对应同一个引用对象（即某个缓冲区对象）。因此，其设计思路是建立一个缓冲区对象表，记录每一个静态、堆或栈缓冲区对象的基地址和缓冲区长度。判断一个由界内指针计算而得的地址是否在界内，首先搜索对象表，得到界内指针对应的参考对象，然后根据其基地址及长度判断当前地址是否在参考对象规定的范围内。这种方法可以对栈、堆及 BSS 段的缓冲区边界进行检查。

它采取了红黑树（Red/Black Tree）的方法来存储缓冲区对象，因为它的性能可以达到 O(logn)，比一般的二叉树要好。另外，在每一个分配的堆缓冲区前加一个特殊结构的数据 meta data 以便于在缓冲区对象表中实现插入、删除操作。它使用“目标代码插入”技术，对程序每次的内存访问操作都插入一段检测指令。所以它不仅可以检测缓冲区溢出，也可以检查一般的内存泄漏等问题。通过用这种方法检查，可执行代码在执行的时候数组的所有引用来保证其合法性。

4.2 指针完整性检查

指针完整性检查技术是在栈的返回地址后加一个附加结构，称为“canary”。如果有溢出攻击，“canary”就会被破坏，从而就会发现有攻击存在。为了避免“canary”被攻击者伪造，设计者设计了两种生成“canary”的方法。一种是利用在 C 语言中的终止符号，如 0(null)，CR，LF，-1(EOF)等不能在常用的字符串函数中使用的符号，因为这些函数一旦遇到这些终止符号，就结束函数过程了。另一种是利用随机符号，用一个在函数调用时产生的一个 32 位的随机数来实现保密，使得攻击者不可能猜测到附加字节的内容。而且，每次调用，附加字节的内容都在改变，也无法预测。

Stack Guard 是针对栈返回地址溢出攻击的，而 Point Guard 则是在所有的指针后都加了“canary”，这样就可以面向所有的缓冲区溢出攻击[2]。

它是通过在函数调用时保存返回地址的值，在函数返回时比较当前返回地址的值与预先保留的值，如果不相同说明有溢出攻击。它可以检测基于栈内返回地址的溢出攻击[1]。

4.3 攻击数据行为检测

这种方法主要是从 Shellcode 的角度来检查溢出攻击。一般来说，远程缓冲区溢出攻击需要传输大量的数据，以便将 Shellcode 植入目标主机，因此可以通过检测输入数

据，并对其进行分析，来判断是否有攻击存在。这种方法与入侵检测系统的设计思路是相同的，在实现上也多依赖于入侵检测系统。分析输入数据是检测的核心，方法有很多，主要有建立攻击数据签名特征[3]，利用沙盒（sandbox）模拟运行攻击代码[5]等。

4.4 编写正确的代码

这是从程序员的角度，也是从源头避免缓冲区溢出的最根本的方法。编写正确的代码是一件非常有意义但耗时的工作，特别象编写 C 语言那种具有容易出错倾向的程序(如:字符串的零结尾)，这种风格是由于追求性能而忽视正确性的传统引起的。尽管花了很长的时间使得人们知道了如何编写安全的程序，具有安全漏洞的程序依旧出现。毕竟再优秀的程序员也会有犯错的时候，因此人们开发了一些工具和技术来帮助程序员编写安全正确的程序。

最简单的方法就是用 grep 来搜索源代码中容易产生漏洞的库的调用，比如对 strcpy 和 sprintf 的调用，这两个函数都没有检查输入参数的长度。事实上，各个版本 C 的标准库均有这样的问题存在。另外，静态分析工具也是程序开发人员检测缓冲区溢出的很有用的工具。

不过，由于开发时间、代码规模或其它原因，不一定所有的代码都能被检测到，同时对于已经存在的程序，还需要有其它的工具来检测和防范缓冲区溢出漏洞。

所有的缓冲区溢出漏洞都源于 C 语言缺乏类型安全。如果只有类型-安全的操作才可以被允许执行，这样就不可能出现对变量的强制操作。如果作为新手，可以推荐使用具有类型-安全的语言如 Java 和 ML。

但是作为 Java 执行平台的 Java 虚拟机是 C 程序，因此通过攻击 JVM 的一条途径是使 JVM 的缓冲区溢出。因此在系统中采用缓冲区溢出防卫技术来使用强制类型-安全的语言可以收到意想不到的效果。

4.5 不可执行的缓冲区

这是从操作系统的角度来实现的预防缓冲区溢出攻击的手段。通过使被攻击程序的数据段地址空间不可执行，从而使得攻击者不可能执行被殖入被攻击程序输入缓冲区的代码，这种技术被称为非执行的缓冲区技术。事实上，很多老的 Unix 系统都是这样设计的，但是近来的 Unix 和 Windows 系统由于实现更好的性能和功能[4]，往往在在数据段中动态地放入可执行的代码。所以为了保持程序的兼容性不可能使得所有程

序的数据段不可执行。

另外，有些攻击是直接利用 C 中的库函数或其它的系统调用来进行的，所以这种方法对这些攻击都是无效的。

五、小结

移动互联网应用的开放性、移动性给人们带来了极大的方便，但是，同时也存在一些不可忽视的安全问题。由于移动互联网应用的多样性，导致可能存在大量安全漏洞，对这些移动互联网应用的承载终端造成威胁，并对最终用户的个人信息和数据产生危害。由于移动互联网应用程序往往与个人隐私信息等相关，因此其造成的危害超过一般计算机中的应用软件。对移动互联网应用中的安全漏洞检测技术进行研究是当今移动互联网发展中的一个热点问题。

本文阐述了移动互联网各类程序中漏洞产生的原理、漏洞的危害，对漏洞检测技术进行了研究，特别对危害极大的缓冲区溢出漏洞进行了详细探讨，希望能为移动互联网时代信息安全问题的研究提供参考。

参考文献

Olatunji Ruwase,Monica SLam.A Practical Dynamic Buffer Overflow Detector[C].In:Proceedings of the11th Annual Network and Distributed System Security Symposium,2004:159 ~ 169

Crispin Cowan, Calton Pu, David Maier, Heather Hinton, Peat Bakke, Steve Beattie, and Jonathan Walpole. Buffer Overflows: Attacks and defense for the vulnerability of the Decade[C]. DARPA Information survivability

Stig Andersson, Andrew Clark, George Mohay. A Framework for Detecting Network-based Code Injection Attacks Targeting Windows and UNIX[C], Proceeding of the 21st Annual Computer Security Application

江魁著，网络安全体系结构，人民邮电出版社，2005

Stig Andersson, Andrew Clark, George Mohay. Detecting Network-based Obfuscated

Code Injection[C], Proceeding of the 21st Annual Computer Security Application Conference 2005: 75~85.

作者简介

赵鑫，男，1982 年生，河北人，硕士研究生，中国移动北京公司，中级工程师，主要研究方向是信息安全与网络安全技术。

朱东来，男，1976 年生，北京人，本科，中国移动北京公司，中级工程师，主要研究方向是信息安全管理。

杨宏，男，1982 年生，北京人，本科，中国移动北京公司，中级工程师，主要研究方向是网络安全技术。