基于文件格式的漏洞挖掘技术研究

陈 韬 孙乐昌 潘祖烈 刘京菊

(解放军电子工程学院网络工程系 合肥 230037)

摘 要 文件格式漏洞的严重威胁性和挖掘复杂性使基于文件格式的软件漏洞挖掘技术成为信息安全领域的一个研究热点。总结了文件格式漏洞挖掘技术的发展历程,重点分析了当前该技术在实际应用中存在的不足,最后提出下一步可能的研究方向。

关键词 文件格式,漏洞挖掘

中图法分类号 TP391 **文献标识码** A

Research on Software Vulnerability Mining Technique Based on File-format

CHEN Tao SUN Le-chang PAN Zu-lie LIU Jing-ju

(Electronic Engineering Institute of PLA, Hefei 230037, China)

Abstract Software vulnerability mining technique based on File-Format becomes more important in information security filed for the file-format vulnerability's serious damage and complexity discovery. Summarized the development of this technique, and mainly analyzed three disadvantages when applied to a special file-format. At last, proposed the further improvement directions.

Keywords File-Format, Vulnerability mining

1 引言

文件格式漏洞指系统或应用程序在处理文件过程中产生的诸如内存读写、缓冲区溢出或整数溢出等错误,这些错误可能导致软件崩溃、信息泄露、未授权或越权访问、远程控制等安全问题。近年来,微软发布了大量影响 Windows 系统的文件格式漏洞。2008 年微软发布的安全公告[1]中描述的漏洞 MS08052 几乎影响所有使用 Microsoft Windows GDI+组件的软件,涉及多种常见的图片文件格式。攻击者利用该漏洞将木马藏于图片中,用户查看图片即可触发漏洞,进而执行攻击者指定的任意代码如下载执行木马、格式化硬盘等。2010 年以来,几乎每个月都有新发现的文件格式漏洞[2],表 1 列举了 2010 年部分文件格式漏洞的名称及其影响的文件格式类型。

计算机使用的文件格式种类繁多,FILEXT 网站统计现有的文件格式大约有 14000 种,而其中大多数文件格式未对外公开其格式文档[3]。文件格式漏洞的严重威胁性和挖掘复杂性使得文件格式漏洞自动挖掘与分析技术成为了信息安全领域的一个研究热点[4]。本文主要介绍了基于文件格式的漏洞挖掘技术研究进展,总结了当前文件格式在漏洞挖掘中存在的 3 大问题,即文件格式知识获取问题、测试用例生成问题和异常监控问题,最后给出了下一步可能的研究方向。

表 1 2010 年部分文件格式漏洞

漏洞编号	漏洞名称	文件类型
MS10-005	Microsoft Paint 中的漏洞可能允许远程执行代码	JPEG
MS10-013	Microsoft DirectShow 中的漏洞可能允许远程执行代码	AVI
MS10-028	Microsoft Visio 中的漏洞可能允许远程执行代码	Visio
MS10-062	Microsoft MPEG-4 编解码器媒体文件解析远程代码执行漏洞	MPEG
MS10-078	OpenType 字体格式驱动程序中的漏洞可能允许特权提升	OTF
MS10-080	Microsoft Excel 中的漏洞可能允许远程执行代码	Excel
MS10-082	Windows Media Player 中的漏洞可能允许远程执行代码	Mp3
MS10-087	Microsoft Word RTF 文件解析栈溢出漏洞	RTF
MS10-088	Microsoft PowerPoint 解析缓冲区溢出漏洞	PPT
CVE-2010-0209	Adobe Flash Player 内存破坏漏洞	SWF
CVE-2010-2216	Adobe Reader 内存破坏代码执行漏洞	PDF

2 基于文件格式的漏洞挖掘技术发展历程

按照测试用例生成过程中知识的运用程度将基于文件格式的漏洞挖掘技术分为 3 个发展阶段,即传统文件格式漏洞挖掘阶段、简单文件格式漏洞挖掘阶段和高级文件格式漏洞挖掘阶段。

2.1 传统文件格式漏洞挖掘阶段

传统的文件格式漏洞挖掘以手工测试[5]和模糊测试[6,7]

陈 韬(1987—),男,硕士生,主要研究方向为网络安全,E-mail:oatnehc0924@126.com;孙乐昌(1951—),男,教授,博士生导师,主要研究方向为分布式操作系统、网络安全;潘祖烈(1976—),男,博士,主要研究方向为网络安全;刘京菊(1974—),女,硕士,副教授,主要研究方向为网络安全。

为主。手工构造测试用例的方式不仅工作量大、测试周期长，且极容易出现测试遗漏。早期模糊测试技术[8]可以追溯到1989年，Barton Miller教授和他的高级操作系统课题组开发和使用了一个原始的模糊器，用来测试UNIX应用程序的健壮性。模糊测试技术采用人工或随机的方法构造大量正常样本数据并利用动态技术监控目标程序加载样本数据的过程，如果程序运行过程中出现异常，则说明程序存在潜在的漏洞。

专用于文件格式漏洞挖掘的工具[9]产生于2004年以后，微软发布的MS04-028漏洞公告详细描述了负责处理JPEG文件的引擎中的一个缓冲区溢出漏洞。尽管这并非第一个被发现的有关文件格式的安全漏洞，但由于许多流行的Microsoft应用程序都共享这段有漏洞的代码，该漏洞还是吸引了人们的注意力。此后文件格式漏洞的发掘和利用一直受到广泛关注，出现了FileFuzz[10]、ffuzzer、fuzzer、SPIKEfile[11]、notSPIKEfile[12]等一系列应用于文件格式的Fuzzing测试工具。文献[13]介绍了文件格式漏洞挖掘技术，并分析比较了多种测试工具的优缺点。FileFuzz和notSPIKEfile分别应用于windows和Linux平台，它们使用基于变异技术(mutation-based)的方法，每次变异样本文件时按照存储顺序改变样本文件的一个字节、字或双字或字符串。相比于随机生成的正常测试用例，基于变异的方法产生的测试数据大多为畸形数据，由于输入型程序产生漏洞的主要原因是程序员在代码实现时没有对用户的输入数据进行有效的检查[14]，因此不符合正常文件规范的畸形数据能够更有效地触发程序的异常。传统文件格式漏洞挖掘阶段所使用的方法理论较简单，可以实现较高的自动化，不用大量的理论推导和公式计算，不需要分析具体的文件格式信息或额外的知识支撑，并且在该方法提出初期就取得了较好的成果。但大多数畸形数据无法突破文件格式中的校验和、固定字段或长度等检查限制，在程序运行过程中被迫提前退出，导致代码的覆盖率不高，存在漏洞挖掘效率低的缺点[15]。

2.2 简单文件格式漏洞挖掘阶段

简单文件格式漏洞挖掘阶段最主要的特点是漏洞挖掘过程中利用文件格式知识指导测试用例的生成。图1所示为简单文件格式漏洞挖掘技术的整体流程，其中包括文件格式知识应用于漏洞挖掘技术的测试用例生成过程、测试过程和监控记录过程。

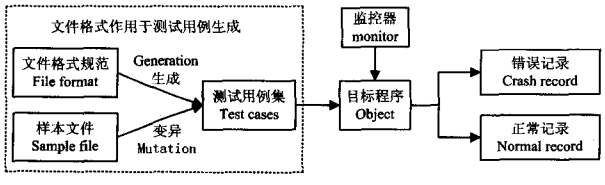

图1 简单文件格式漏洞挖掘技术的整体流程

简单文件格式漏洞挖掘阶段的测试用例生成方法可以分为基于生成技术(generation-based)的方法、基于变异技术的方法和两者相结合(如工具Peach[16])的方法[15]。基于生成技术的测试用例生成方法[17]通常需要给出文件格式具体的描述规则。依据文件描述规则产生测试用例需要用户对文件格式有非常深入的了解，因此需要大量的人工参与。该方法产生的数据可以有效地越过应用程序中对固定字段、校验和、长度的检查，从而使测试用例的有效性大大加强。基于变异技术的测试用例生成方法[18]通常是在对文件格式有所了解的基础上，对样本数据中的某些域进行随机变化，从而产生新的变异数据。该方法对初始样本有着很强的依赖性，不同的初始样本数据会带来不同程度的代码覆盖率，从而产生差异很大的测试效果。文献[19]首次提出利用基于块的协议知识生成测试用例，大大增强了Fuzzing测试用例的有效性，其效率相比于手工测试或随机模糊测试有了非常大的提高。文献[20]比较了两种方法的优缺点并通过PNG实验表明，基于生成的测试用例的代码覆盖率比基于变异的测试用例更高。

简单文件格式漏洞挖掘产生测试用例的方法仍采用随机模糊测试。通常针对某一特定属性需要维护一张模糊数据列表，例如模糊器预先制作字符串数据模糊列表(见表2)针对字符串属性的生成或变异，这样的列表包含常见的字符串编码类型集、不同长度的字符串集、含有特殊字符的字符串集等。模糊数据列表的建立能够有针对性地对特定数据类型进行构造或变异，并且用户可以根据获得的特殊知识(如程序分析知识、文件格式信息等)向列表添加可能触发漏洞的字符串。但与此同时，模糊列表的建立限制了测试用例的数量，当列表中的数据全部完成测试用例生成或变异后，测试就结束了；当模糊列表规模较小时，很可能会减少漏洞的发现数量，而创建大规模的列表则需要使用者非常熟悉能有效触发漏洞的特殊数据。

表2 字符串数据模糊列表

字符串类型	实例
单个字符	'\x08','\n','C'等256个单个字符
不同类型的字符串	"ABCDEFGHIJKLM" "2,147,483,649" "\xef\x83\xb0\xef\x83\xb0" "A\x00A\x00A\x00A\x00"
不同长度的字符串	"\u40b2\u7175\u1c45\ud17f"等 各类长度的字符串集
含特殊字符的字符串	"..\\..//…\./.\\.a.\\..\\../.\..\\1.txt" "tA<mzY\r<Y@o\r&g}@cd,.rZ_/" "%n%s%n%s%n%n%"等

简单文件格式漏洞挖掘产生测试用例的方法其优点是执行效率比较高、应用范围广、通用性强，缺点是需要大量的人工参与来进行文件格式知识的获取并实现这些知识到测试用例的转化。

2.3 高级文件格式漏洞挖掘阶段

高级文件格式漏洞挖掘的特点是在挖掘过程中融入了更多的知识，产生的测试用例更加具有针对性，即测试用例的生成更加针对程序中可能产生潜在漏洞的代码。文献[21]中的vulnerable points、文献[22]中的attack points和文献[23,24]中的vulnerable statements实质上都指可能产生漏洞的代码，这里统一称为不安全代码。高级文件格式漏洞挖掘技术通过分析程序中不安全代码的特征，有导向地产生测试用例，可以更加高效地发掘潜在的漏洞。

文献[25]应用静态分析技术标识和定位目标软件中不安全函数的位置，这些不安全函数既包括库中的标准函数(如strcpy())，也包括由程序员自己定义实现的一些不安全函数。文献[26]提出了使用输入输出分析技术寻找文件节点与不安全代码之间的映射关系的方法，该方法得到的结果准确率较高，通过查找到的不安全代码，有导向地针对文件中相应节点进行生成或变异技术，实现对不安全代码的测试。

为了准确到达需要测试的代码，通常采用遗传算法等启发式方法生成测试用例以覆盖不安全代码，文献[21]基于控制依赖图应用遗传算法覆盖包含不安全函数的语句；文献[23,24]以提高代码覆盖率、脆弱语句覆盖率、嵌套深度以及缓冲区的占用率来发现程序中的漏洞，为导向设计遗传算法产生覆盖不安全函数。

遗传算法是一种全局智能搜索算法，图 2 所示为遗传算法针对预定目标生成测试用例的流程图。

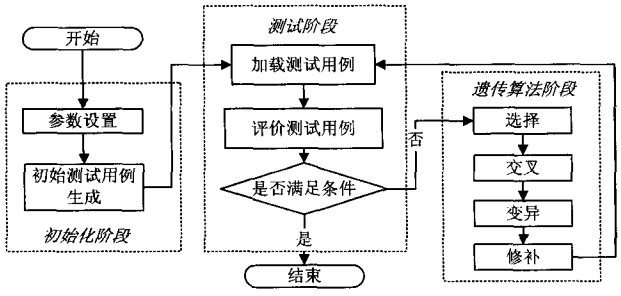

图 2　遗传算法应用于测试用例生成

遗传算法用于文件格式漏洞挖掘分为 3 个阶段。初始化阶段随机产生初始测试用例集；测试阶段通过分析加载过程中测试用例与预期目标之间的差距评价测试用例的优劣，如果测试用例满足预期的目标，则完成针对该目标的测试，否则进入遗传算法阶段。经过遗传算法的选择、交叉和变异操作重新生成新的测试用例，并通过修补操作使测试用例满足文件格式的要求，重新测试新生成的测试用例，直至满足所设定的条件为止。遗传算法由于在迭代过程中伴随信息交换的进行，优良的品质被逐渐保留并加以组合，从而不断产生出更佳的个体，实现对测试用例的优化，最终能够找到较优的解来满足预期的目标。神经网络算法和退火免疫遗传算法通常与遗传算法结合使用，以防止遗传算法早熟和局部收敛差的自身缺陷。

利用启发式算法针对不安全代码生成测试数据，其测试效率要高于纯粹的简单 Fuzzing 技术。然而不安全代码定义的完整性和查找的准确性都影响到漏洞挖掘的效率，并且针对复杂不安全代码的适应度函数设计比较困难，不同的不安全代码需要设计不同的适应度函数。因此该方法还无法大规模应用于各种漏洞的挖掘。

3　基于文件格式的漏洞挖掘技术面临的问题

3.1　文件格式知识获取问题

文件格式分析是软件测试和漏洞挖掘的一个重要组成部分，它关系到测试用例的生成，直接影响到软件的漏洞挖掘效果。大量漏洞挖掘和分析实验[27-30]表明，知识的获取程度直接关系到漏洞挖掘效果的好坏，而一般漏洞挖掘活动中在文件格式分析上的时间几乎占整体漏洞挖掘活动周期的 80%。因此，提高文件格式知识获取的分析效率，将有效提高整体漏洞挖掘过程的效率，大大缩短每次漏洞挖掘的活动周期并减少人力投入。

大量漏洞挖掘实验利用文件格式信息产生测试用例，但所使用到的文件格式知识获取方法都停留在公开文档的分析或手工分析阶段，并没有提出具体高效的文件格式分析技术。以下从公开文档分析技术、文件比较分析技术和逆向分析技术三个方面总结当前文件格式分析技术在漏洞挖掘应用中的不足之处。

1)公开文档分析技术

部分文件格式具有公开的、不同程度规范的格式文档。格式文档描述了数据如何编码、如何排列和数据的各个属性结构，或者规定了是否需要特定的计算机程序读取或处理。因此一般从设计文档中可以分析得到较为详细的文件格式信息。但由于设计或维护等因素导致其中一部分文档不规范或不完整，因此针对公开文档的分析受到其规范程度的影响。并且不同的软件会对应不同的文件处理方法，程序员在实现软件功能时不一定会严格遵守已发布的格式标准。例如 PNG 文件格式文档[31]中规定对数据块进行校验和计算，但使用微软的图片管理器或使用 windows 图片和传真查看器查看 png 图像时不会对辅助数据块进行校验和计算。因此仅仅依靠分析公开的格式文档得到的文件格式信息并不能真实地反映出软件处理文件的过程。

2)文件比较分析技术

文件比较是格式分析方法中比较直观的方法。它直接收集大量的样本文件，通过比较这些文件中的共同点得到文件小部分比较明显的固定格式。例如文件头的标志信息，文件的作者、版权、备注等明文信息。通过使用 010Editor 的文件比较功能可以方便地找到两个二进制样本文件的相同点和不同点。

虽然使用文件比较的方法分析文件格式比较直观，但这种方法只能够对相对简单的文件格式进行分析，并且得到的文件格式往往不精确、不完整，而且无法得到各节点之间的关系。

3)逆向分析技术

通过逆向分析同样可以得到较为详细的文件格式信息。对于有经验的逆向工程师，只需要提供可执行文件和足够的时间就可以从二进制代码中推断出比较准确的文件结构属性和各个结构的意义。Eldad Eilam 的经典逆向工程书籍[32]介绍了逆向工程的各类相关知识。其中在破译未公开的文件格式一部分中主要讲解了如何使用数据逆向工程及技术获得与专业数据格式相关的未公开信息。刘颖东等人[33]介绍了通过分析游戏资源压缩包文件格式的方法从游戏中提取图像、音乐等资源文件。文献[34]通过跟踪程序执行流程，提取程序输出结果的格式信息。

二进制程序的逆向分析技术主要分为静态分析技术[35]和动态分析技术[36-38]。静态分析技术指通过反编译器将二进制的可执行文件转换为可供阅读的汇编代码或其他易于理解的表现形式。文献[32]介绍了如何使用逆向技术深入分析研究未公开的程序数据，主要用静态分析的方法详细讲解了 Cryptex 软件所生成的 CRX 文件格式。动态调试分析技术通过调试器加载应用程序，监控程序在加载文件时的运行状态，记录程序处理文件的过程，从而分析得到文件的格式信息。文献[39]通过动态分析 AdobeReader 进程内存堆的内容发现 PDF 树结构；在逆向分析中通过动态跟踪对相应节点数据的处理代码，可以进一步确定软件中代码的功能。

静态分析方法是对代码的全局分析，因此可以得到比较完整的文件格式信息，但这种方法需要及其丰富的反汇编知识和大量的经验积累，花费的时间较长，导致文件格式分析的

效率不高。另外一旦应用程序被加壳处理或被加密，则无法通过静态分析的方式直接分析其代码数据。动态分析技术虽然可以得到文件被处理的过程，但这种方法受样本文件的影响较大，是对代码的局部分析，无法全面地分析得出完整的文件格式信息。通常在实际文件格式分析中往往将静态分析技术和动态调试技术两种方式相结合，发挥各自的优势。文献[40]分析了漏洞挖掘系统对数据输入结构的需求。文献[41]同时利用静态分析和动态执行技术通过跟踪和分析堆栈、反汇编代码、寄存器等信息自动分析二进制文件的域属性。文献[42,43]描述了针对协议输入数据的格式分析技术。

综合以上文件格式分析方法的现状可以发现，当前针对文件格式的分析大多依靠人工对文档或可执行文件的手工分析，这不仅需要较多的经验积累，而且需要大量的时间，导致文件格式分析的效率非常低。因此如何提高文件格式分析的效率已成为当前漏洞挖掘的一大重要问题。

3.2 基于文件格式的测试用例生成问题

测试用例生成技术是漏洞挖掘的关键技术。测试用例的数量和质量是影响漏洞挖掘效率的主要因素。若测试用例过多，则测试时间长，挖掘效率低；测试用例过少则可能无法触发软件潜在的漏洞。而测试用例质量越高，表明测试用例所覆盖到的代码越多，能发现漏洞的概率越大。如何尽可能地减少测试用例的数量和提高测试用例的质量成为测试用例生成技术的重点和难点。

测试用例的生成有两种方式，一种为预先生成方式，一种为动态生成方式。理论上，长度为 n 字节的文件能够产生 2^{8n} 个测试用例，但其中大部分数据都为冗余数据，因此预先生成所有测试用例是不可行的。文献[17]基于文件的规范，抽象地描述了文件推导规则，定义了文件模糊测试模板，设计了文件模糊变异模型。在规范描述下生成不同类型的文件，然后对每类文件进行变异模糊测试，有效地减少了大量无效测试。文献[27]对 mp3、m3u、pls 文件格式进行了分析，并根据文件格式设定了基于生成方式的测试用例生成策略，即选择不断增加字符串长度的方法来处理字符串参数。该文章对文件格式信息的使用比较简单，只使用了字符 A 来填充字符串。文献[38]通过改变文件头的结构如增加、循环或改变数值等方式进行测试，这种变异的方法对数据生成没有指导性的意义，仍然属于随机测试的范畴。文献[28]研究了目标文件脆弱点，即选择一个文件格式之后通过了解该文件格式，找到文件的脆弱点。对于影响脆弱点数据的变异，该方法已初步地利用格式信息来指导测试用例生成。但必须对文件格式有很深的了解，脆弱点的选择仍然需要人工分析得出。为了减少测试用例的数量和提高其质量，采用启发式的方法生成测试用例成为一种重要手段，贪心算法[45]、遗传算法[46,47]和模拟退火算法[48]等有导向的生成测试用例针对性更强，效果更好。但这些启发式算法本质上属于搜索算法，针对不同的目标需要不同的搜索策略。文献[49]介绍了多维测试用例生成方法，采用遗传算法有导向地针对脆弱点产生测试用例，该方法可以挖掘出多个因素触发的漏洞，但脆弱点的查找与判断仍然建立在文件格式分析的基础上，因此适应度函数的设计需要对脆弱点的特性进行人工分析。

当前的测试用例的生成方法仍然无法做到测试用例集的精简和有效，如何做到质量和数量之间的平衡仍然是一个急需解决的问题。

3.3 异常监控问题

异常监控是漏洞挖掘技术效果的判断依据，漏洞挖掘效果的好坏通常以触发异常的数量来评判。按图 1 所示，早期的监控器监控目标程序，一旦程序崩溃，则认为程序存在潜在的漏洞。但事实上，漏洞的触发不一定会引发程序的崩溃。根据漏洞的种类可以将异常分为应用程序崩溃、内存读写错误、占用系统资源、越权访问等。文献[50]中介绍了异常处理的概念和处理异常的方法，描述了各种异常的鱼骨图结构。其中某些异常难于检测，例如程序发生越权访问错误，Fuzzing 测试可以发现软件系统中缓冲区溢出等安全漏洞，但是无法检测出用普通用户的访问身份成功地访问了只能管理员才能访问的功能；或者程序发生了缓冲区溢出，却并没有导致程序崩溃；或者程序陷入死循环、信息泄露、大量占用系统 CPU、内存资源等情况。现在大多数的 Fuzz 工具对错误检测的手段还处在一个比较初级的阶段。主要解决方法是通过分析应用程序维护的日志和系统维护的日志来进行错误检测。这种方法能够检测到的错误种类非常有限而且准确性不高，无法检测到应用程序内部被异常处理例程所处理的错误并且需要大量的人工参与，自动化和智能化程度不高。

结束语 基于文件格式的漏洞挖掘技术相对于网络协议的测试更加复杂，影响甚大，但当前的文件格式漏洞挖掘技术仍存在诸多不足，下一步的发展趋势主要体现在下面几个方面：

1) 提高文件格式分析技术的自动化水平。文件格式分析消耗时间长，效率低。而知识的详实程度直接影响到测试用例的生成。当前文件格式逆向分析技术已有大量的实际经验可循。下一步可以将这些经验转化成自动化的分析工具，以提高整个漏洞挖掘的效率。

2) 提高测试用例的有效性，防止组合爆炸问题。测试用例生成技术是整个漏洞挖掘中的关键技术，测试用例集必须同时保证质量和数量，才能高效地挖掘出软件的潜在漏洞。融合了知识的测试用例生成方法自动化程度高，通过融合文件格式知识、程序静态知识、动态知识等方式可以加强测试用例生成的有效性。

3) 建立一套行之有效的异常监控机制。异常监控在挖掘过程中非常重要，它直接关系到漏洞挖掘的成效，是真正体现漏洞挖掘的价值所在。从已有的漏洞挖掘实际中总结异常类型，分析其表现形式，弥补当前异常监控的不足之处。

参 考 文 献

[1] Microsoft 安全公告[OL]. http://support. microsoft. com. kb/954593,2009-10

[2] NSFOUCS 绿盟科技安全小组[OL]. http://www. nsfoucus. net,2011-03

[3] FILEXT The file Extension Source[OL]. http://filext. com, 2011-04

[4] 唐彰国，钟明全，李焕洲，等. 基于 Fuzzing 的文件格式漏洞挖掘技术[J]. 计算机工程，2010,36(16):151-153

[5] 迟强，罗红，乔向东. 漏洞挖掘分析技术综述[J]. 计算机与信息技术，2009,(Z2):90-92

[6] Sutton M, Greene A, Amini P. Fuzzing: Brute Vulnerability Discovery [M]. New York: Addison-wesley professional, 2007: 21-

28
[7] Oehlert P. Violating Assumptions with Fuzzing[J]. IEEE Computer Society,2005,3(2):58-62
[8] Miller B P,Fredrisksen L,So B. An empirical study of the reliability of Unix utilities[J]. Communication of ACM,1990,33(12):32-44
[9] Fuzzing Tools[EB/OL]. http://www. threatmind. net/secwiki/FuzzingTools,2007-12
[10] Sutton M,Defense I. FileFuzz[CP/OL]. http://labs. idefense. com/software/,2010-10
[11] Green A,Defense I. SPIKE[CP/OL]. http://labs. idefense. com/software/,2011. 04
[12] Green A,Defense I. notSPIKE[CP/OL]. http://labs. idefense. com/software/,2011-04
[13] Sutton M,Greene A. The art of file format fuzzing[C]//USA: Black hat USA Conference. 2005:1-20
[14] Warnock M. Look Out! It's the Fuzz! IAnewsletter,2007:1-4
[15] 吴志勇,王红川,孙乐昌,等. Fuzzing 技术综述[J]. 计算机应用研究,2010,27(03):829-832
[16] Peach[CP/OL]. Web page: http://www. peachFuzzer. com, 2011-04
[17] 沈亚楠,赵荣彩,王小芹,等. 基于规范生成的文件模糊测试[J]. 计算机工程与设计,2010,31(16):3591-3594
[18] 陈莹莹. 变异技术在软件测试中的研究及应用[J]. 科协论坛(下半月),2008(12):67-68
[19] Aitel D. The Advantages of Block-Based Protocol Analysis for Security Testing[R]. Immunity,2002
[20] Miller C,Peterson Z N J. Analysis of Mutation and Generation-Based Fuzzing [EB/OL]. http://securityevaluators. com/files/papers/analysisFuzzing. pdf,2009-07
[21] Liu Guang-hong,Wu Gang,ZhengG Tao,et al. Vulnerability analysis for x86 executables using genetic algorithm and fuzzing [A]//Third 2008 International Conference on Convergence Hybird Information Technology (ICCIT)[C]. 2008:491-497
[22] Ganesh V,Leek T,Rinard M. Taint-based Directed Whitebox Fuzzing[C]//Proceedings of the 31st International Conference on Software Engineering. Washington,DC,2009:474-484
[23] Del Grosso C,Di Penta M,Antoniol G,et al. Improving network applications security: a new heuristic to generate stress testing data[A]//Proceedings of the genetic and evolutionary computation conference[C]. Washington,DC,USA,2005:1037-1043
[24] Del Grosso C,Di Penta M,Antoniol G,et al. Detecting buffer overflow via automatic test input data generation[J]. Computers and Operations Research,2008,35(10):3125-3143
[25] 任华. 基于不安全函数的缓冲区溢出发现技术研究[D]. 解放军信息工程大学,2007
[26] Schroeder P J, Korel B. Black-box test reduction using I/O analysis[C]//Proceedings of the 2000 ACM SIGSOFT international symposium on Software testing and analysis (ISSTA '00). Portland,Oregon,2000:173-177
[27] 魏瑜豪,张玉清. 基于 Fuzzing 的 MP3 播放软件漏洞发掘技术[J]. 计算机工程,2007,33(24):158-160
[28] 贺拓. Flash 应用程序漏洞挖掘与利用[D]. 西安:西安电子科技大学,2010
[29] 高峻,徐志大,李健. 针对复合文档的 Fuzzing 测试技术[J]. 计算机与数字工程,2008,36(12):116-119
[30] 夏建军,孙乐昌,吴志勇,等. 基于 Fuzzing 的 PNG 漏洞挖掘技术[J]. 计算机与数字工程,2010,38(03):92-96
[31] RFC2083 PNG Portable Network Graphics Specification Version 1. 0[EB/OL]. http://www. portal. acm. org,2011-03
[32] Eilam E,Chikofsky E. Reversing: Secrets of Reverse Engineering[M]. Indiana:Wiley Publishing,2005:199-242
[33] 刘颖东. 揭秘数据解密的关键技术[M]. 北京:人民邮电出版社,2009:87-97
[34] Lim J,Reps T,Liblit B. Extracting Output Formats from Executables[C]//Proceeding of the 13th Working Conference on Reverse Engineering(WCRE'06). Benevento,Italy,2006:167-178
[35] Livshits V B,Lam M S. Finding security vulnerabilities in java applications with static analysis[C]//Proceedings of the 14th conference on USENIX Security Symposium. Volume 14,2005:18
[36] Newsome J,Song D. Dynamic taint analysis for automatic detection,analysis,and signature generation of exploits on commodity software[C]//Network and Distributed System Security Symposium (NDSS). 2005:1-38
[37] Suh G E,Lee J,Devadas S. Secure Program Execution via Dynamic Information Flow Tracking[M]. Cambridge: Computer Science and Articial Intelligence Laboratory,2004:1-14
[38] Xu Wei,Bhatkar S,Sekar R. Practical Dynamic Taint Analysis for Countering Input Validation Attacks on Web Applications
[39] 曹旭,张一宁. 基于逆向工程和 Fuzzing 技术的 AdobeReader 漏洞发掘技术研究[J]. 信息工程大学学报,2009,10(02):204-206
[40] 郑亮. 基于输入触发的漏洞挖掘模型[J]. 计算机工程与设计,2009,30(18):4227-4230
[41] Kim H C,Choi Y H,Lee D H. Efficient file fuzz testing using automated analysis of binary file format[J]. J Syst Architect,2010,57(3):259-268
[42] Cui W,Peinado M,Chen K,et al. Tupni: Automatic reverse engineering of input formats[C]//15th ACM Conference on Computer and Communications Security. New York,2008
[43] Caballero J,Yin Heng,Liang Zhen-kai,et al. Polyglot: Automatic Extraction of Protocol Message Format using Dynamic Binary Analysis[C]//Proceedings of the 14th ACM conference on computer and communications security ACM. 2007:317-319
[44] Lewis C,Rhoden B,Sturton C. Using Structured Random Data to Precisely Fuzz Media Players. 2007
[45] 单锦辉,高友峰,刘明浩,等. 一种新的变异测试数据自动生成方法[J]. 计算机学报,2008,31(06):1025-1034
[46] 吴云,胡小娟,邱宁佳,等. 基于遗传算法的测试用例生成技术研究[J]. 长春理工大学学报:自然科学版,2010,33(03):137-139
[47] Berndt D,Fisher J,Johnson L. Breeding software test cases with genetic algorithm[J]. System Sciences,2003,2(1):6-9
[48] 李小青,张文祥. 基于退火免疫遗传算法的测试用例生成研究[J]. 计算机仿真,2008,25(05):171-174
[49] 吴志勇,夏建军,孙乐昌,等. 多维 Fuzzing 技术综述[J]. 计算机应用研究,2010,27(08):2810-2813
[50] 姜淑娟,徐宝文. 异常处理—一种提高软件健壮性的方法[J]. 计算机科学,2003,30(9):169-172