doi:10.3969/j.issn.1000-436x.2013.04.013

基于控制流序位比对的智能 Fuzzing 测试方法

王颖，杨义先，钮心忻，谷利泽
（北京邮电大学 信息安全中心，北京 100876）

摘 要：在国际前沿技术 EFS（evolutionary fuzzing system）的研究基础上，提出基于控制流序位比对算法的智能 Fuzzing 测试方法。根据遗传算法的内在属性演算得到基于序列比对的适应度函数，并有效地计算出需要搜索的程序逻辑空间。最后给出了该方法与 2 种传统 Fuzzing 方法的测试性能的实验结果比对，证明了该方法能够充分利用遗传算法属性中并行性进行智能地程序逻辑学习，具有逻辑覆盖面广、搜索导向性强的优点，能够提高漏洞挖掘能力。

关键词：智能 Fuzzing；控制流；遗传算法；漏洞
中图分类号：TP 393.08 **文献标识码：**B **文章编号：**1000-436X(2013)04-0114-08

Smart Fuzzing method based on comparison algorithm of control flow sequences

WANG Ying, YANG Yi-xian, NIU Xin-xin, GU Li-ze
(Information Security Center, School of Computer, Beij ng University of Posts and Telecommunications, Beijing 100876, China)

Abstract: Flowing the way introduced in the research of evolutionary fuzzing system (EFS), a smart fuzzing method was proposed based on the node comparison algorithm among control flow sequences. Through mapping program execution flow sequences onto the control flow sequences, the isomorphism relationship between dada search space and program logic space was established. The analyzed results prove that the method is capa of mining a mass of information from group data effectively, and is able to fully util ze the parallelism of genetic algorithm to guide the fuzzing test.
Key words: smart Fuzzing; control flow; gene algorithm; vulnerability

1 引言

Fuzzing 测试是通过向程序输入大量的、非预期的输入，完成对程序潜在漏洞的发现，其自动化程度高、适应性广的特点使其成为漏洞挖掘较有效的方法之一。然而，传统的 Fuzzing 测试属于随机黑盒测试，产生的测试数据对程序的逻辑适应性低，导致测试数据冗余程度高，因此研究具有导向性的“smart fuzzer”成为 Fuzzing 测试技术的发展方向，并产生了下面 3 个主要的研究分支。

1) 文件格式 Fuzzing 测试，其基本思想是首先分析目标软件要解析的文件或协议的格式，在理解文件格式或协议的基础上，构造符合文件或协议格式的测试数据。例如，2007 年 Choi YoungHan 等开发的基于标签识别的 Fuzzing 测试工具(TAFT)[1]、2008 年刘奇旭等[2]针对 TFTP 服务器进行的 Fuzzing、2011 年姚洪波等[3]开发的针对 Windows 内核的 Fuzzing 工具中应用的测试方法均是这个思想。

2) 符号执行的 Fuzzing 测试，其基本思想是通过在程序的控制流图上进行约束求解，指导 Fuzzing 测试数据集的生成。目前国际上相关研究组织在这一领域取得了一系列的研究成果，DART[4] (directed automated random testing)、SAGE[5] (scalable, automated, and guided execution)和 TaintScope[6]工程中实现的方法都采用了这个思想。

3) 进化 Fuzzing 测试，其基本思想是利用遗传

收稿日期：2012-07-04；修回日期：2012-10-18
基金项目：国家自然科学基金资助项目（61121061）
Foundation Item: The National Natural Science Foundation of China (61121061)

算法来构造 Fuzzing 测试数据。该思想起源于 2007 年的黑帽大会，并被 DeMott J D 等命名为 evolutionary fuzzing system(EFS)[7]方案。

以上 3 种方法在如何增强 Fuzzing 测试的导向性问题上，分别从不同角度进行了有意义的探索，各具优点，但在总体上进行比较，进化 Fuzzing 测试方法具有较大优势，主要体现在以下 2 个方面。

1) 基于“学习”的智能导向性

文件格式 Fuzzing 测试方法通过分析文件格式使测试具有导向性，其相应的代价是测试中对文件格式的分析主要依靠人工完成，工作量大，且通用性差；符号执行 Fuzzing 测试方法通过符号执行增强测试的导向性，其相应的代价是使测试过程中计算的开销随着程序规模的扩大而急剧增大。进化 Fuzzing 是通过遗传算法为测试过程提供导向，这种导向性由于遗传算法本身自组织、自学习的特性，从而具有了“启发式学习”的智能，这对解决程序测试中 NP 完全问题非常有效。

2) 导向性和随机性的有机结合

随机性是 Fuzzing 测试的基本思想，对其增加导向性的同时仍要保证其随机性，这是一对矛盾，而遗传算法[8]可以很好的解决这个矛盾，它借鉴了达尔文的进化理论，即在生物随机变异的过程中通过选择作用保证进化的导向性，从而进化 Fuzzing 方法可以有机的将随机性和导向性相结合，因此将进化算法应用于 Fuzzing 测试，是在“智能”Fuzzing 测试研究探索上迈出的重要一步，也是一个重要的里程碑。不过进化 Fuzzing 仍存在部分关键问题需要进一步分析研究，主要表现在以下 2 个方面。

① 个体基因模型问题。遗传算法应用于具体问题时，首先需要建立能够解决问题内在实质或能够表达个体内在实质的描述模型，这如同自然生物外在的不同性状体现了其内部染色体中不同的遗传基因一样。目前，国际上现有的相关方法缺少对这一方面的研究，例如 EFS 系统中就没有对个体基因模型的描述方法。

② 适应度函数的问题。进化目标是遗传算法计算所得到的最优解，因为遗传算法在搜索最优解的过程中，仅以适应度函数为依据，不利用外部信息[8]，所以适应度函数的设计至关重要。目前国际上在这方面还没有深入的研究和可喜的成果，比如 EFS 中只是简单地用代码覆盖率来做适应度函数。尽管代码覆盖率在一定程度上可以说明测试的充分性，但代码覆盖不能等同于逻辑覆盖，而程序漏洞和程序的逻辑却密切相关，因此 EFS 的适应度函数的设计存在缺陷。

为了解决传统 Fuzzing 测试的不足，本文在进化 Fuzzing 思想的基础上，提出了基于控制流序位比对的智能 Fuzzing 方法。

2　基于控制流序位比对算法的智能 Fuzzing 方法

基于控制流序位比对的智能 Fuzzing 方法的核心问题是采用控制流分级目标划定策略，设计出控制流序位基因型和基于控制流序位比对的适应度函数。

2.1　方法概述

基于控制流序位比对的智能 Fuzzing 方法的总体思想是在对程序进行静态预分析[9]、标记预设[10]和动态控制流跟踪[11]的条件下，首先针对每个测试用例驱动程序进行跟踪，用个体遗传基因标记产生的执行节点序列，其次设计用于挖掘每代群体中控制序列流差异信息的适应度函数，筛选出优势测试数据个体，然后将选出的优势个体进行交叉、变异等操作产生下一代测试个体，并且不断循环迭代这一过程，最后通过进化选择，智能地产生各种适应程序不同逻辑的测试数据，直到最终发现潜在漏洞。这一方法的整体框图如图 1 所示。

在实现基于控制流序位比对的智能 Fuzzing 方法的过程中，关键要解决好 2 个环节。一个是基于控制序位基因模型的设计，另一个是控制流序位差异的适应度函数的设计，这 2 个环节分别解决了进化 Fuzzing 中存在的个体遗传基因型描述和适应度函数的缺陷问题。

2.2　控制流序位基因模型

控制序位基因模型的主要构架的内容是将 Fuzzing 测试中产生的不同测试数据作为相互独立的个体，把每个测试个体在测试过程中产生的控制流序列作为其内在遗传基因。在动态执行跟踪的过程中，以程序的基本块或者抽象函数为跟踪节点，待每一个测试数据跟踪执行完成后，产生一个由跟踪节点构成的有序序列，然后根据序列中所包含的长度，位置等要素，对该序列进行编码，得到对应每个测试用例的唯一的遗传信息编码 P_s。P_s 通常是一个有序序列，其中，包含了测试数据个体驱动程序的执行路径、调用模块和函数以及基本块的名称、数量、次数、顺序等信息。P_s 中序列的长度用

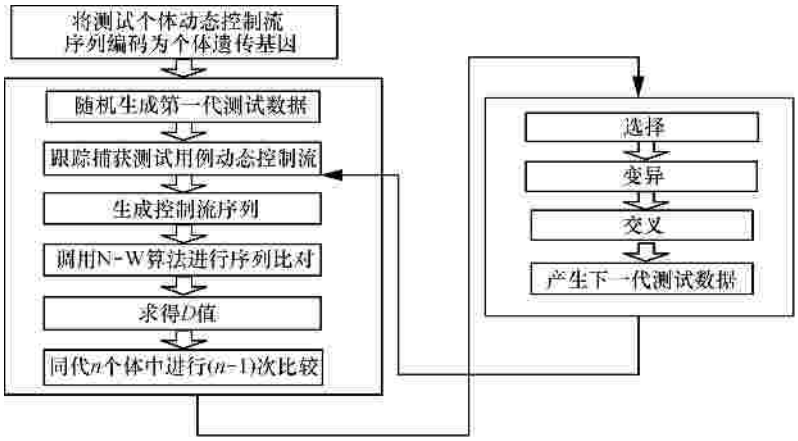

图 1 基于控制序位比对算法的智能 Fuzzing 方法

$|P_s|$ 表示，控制流序列 P_s 中的第 i 个跟踪点以 $P_s[i]$ 表示，$P_s[i]\in N$，N 是程序控制流图中跟踪点的集合。

建立该描述模型的主要原因是软件漏洞与程序的逻辑密切相关，程序的内在逻辑中存在的缺陷是产生漏洞的主要原因，而对应于不同的程序逻辑缺陷，潜在的漏洞只会在某种特定输入条件下才能被触发，那么找到这种特定的输入数据就找到了错误的逻辑。如果将每个测试数据都看作进化进程中的个体，测试数据的结构只是个体的外在表现，而测试数据个体的内在遗传基因是它驱动程序产生的控制逻辑流程，因此可用控制流序位来对这个内在基因进行编码。

下面通过分析对比 3 种基本的控制流与软件程序的基本逻辑关系，来说明控制流节点间的序位关系能够反映出测试数据驱动程序所执行的逻辑。

1) 顺序逻辑

程序执行的顺序逻辑对应控制流序列上的连续的 2 个节点。

2) 分支逻辑

对图 2 中 2 条不同的测试用例 P_s^1 和 P_s^2 之间相对应的一段片断进行对比后可发现，在以相同顺序出现 n_1、n_2 2 个节点之后，2 条序列分别出现了彼此不同的第 3 个节点 n_3 与 n_4。这说明了通过序列比对，可以分析出测试用例驱动了程序中 2 个不同的分支逻辑，能够反映出 2 条序列在分支逻辑上所具有的差异性。

3) 循环逻辑

分析图 3 中的控制流序列片段，在连续出现 n_1、n_2、n_3、n_4 节点之后再次出现节点 n_2、n_3，说明这段序列片段中包含循环控制逻辑。

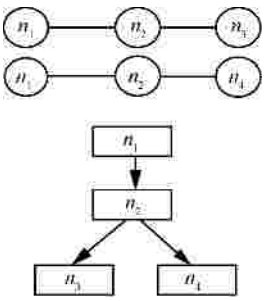

图 2 控制流序列对应的分支逻辑

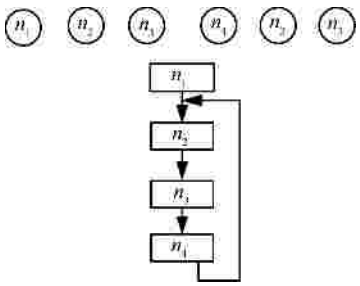

图 3 控制流序列对应的循环逻辑

由于无论多么复杂的程序逻辑都是由顺序、分支、循环 3 种基本控制逻辑构成，因此控制流序列可以反映程序的内在逻辑，并且控制流序列与程序的所有可能执行逻辑之间具有一一对应的同构关系，所有反映预期或非预期的程序逻辑的控制流序列组成了一个搜索空间。如果以测试数据个体驱动程序所产生的控制流序列作为测试个体的遗传基因时，就可以建立起程序逻辑空间和测试数据空间之间的映射关系，然后再利用适应度函数对测试数据的逻辑适应性作出检测与评估，就能确定出有效的搜索空间。

2.3 基于控制流序位比对的适应度函数

在建立控制流序位基因模型的基础上，就可以

开始构建基于控制流序位比对的适应度函数。

2.3.1　控制流序位差异

在获得基本序列和基本逻辑对应关系的基础上，通过比对这些序列位置，可以分析不同的测试用例用于程序逻辑分析上的差别。此时引入个体遗传信息编码 P_s，用来表示跟踪点之间的控制流关系，那么不同的测试用例之间对程序逻辑的适应性差异就可通过对比它们各自 P_s 之间的距离差异获得。假设有 2 个测试用例的路径编码分别为 P_s^1、P_s^2，如果 2 个测试用例使程序执行了相同的逻辑，则可表示为

$$\left|P_s^1\right| = \left|P_s^2\right| \tag{1}$$

$$P_s^1[i] = P_s^2[i], \quad 0 < i \leq \left|P_s^1\right| \tag{2}$$

在式(1)和式(2)的基础上，本文给出计算控制流序位比对算法的演算公式

$$D = \left(P_s^1, P_s^2\right) \tag{3}$$

D 表示任意 2 个序列之间的相似度距离，D 越大就说明序列之间逻辑的差异越大，D 越小就说明序列之间逻辑的相似性越大。

2.3.2　适应度函数描述

构建合适的适应度函数就是要将 Fuzzing 测试的目标转化为进化搜索的目标，下面对分析、转化过程和获得适应度函数进行分别阐述。

由于程序中存在的潜在漏洞所对应的程序逻辑是超出设计者设计范围的非预期逻辑，这种逻辑在正常情况下不会执行，因此进化测试的目标就转变为搜索得到软件中尽可能多的互斥的程序逻辑，其中，还包括各种不同的程序逻辑分支以及各类深层循环逻辑等。但是，要全面获得执行逻辑在工程上实现比较困难，因为程序庞大的逻辑空间具有不可穷尽性，而且事先无法对所有可能执行的逻辑子空间进行穷举分析，甚至不能对进化目标预设数量和范围，使得适应度函数设计中面对的是一个未知空间的探索问题，导致设计出有效的适应度函数的难度较大。

为了解决这个难题，关键的步骤是如何获得优势个体。通常可以通过分析遗传算法本身固有的自学习、自组织特性，充分利用测试完毕的群体数据作为已知信息对其他未知空间实施搜索，然后筛选出本代群体中的优势个体，并通过分析已知个体的控制流序列得到程序中部分逻辑。根据实践经验，为了实现对程序控制流图中的更多逻辑覆盖，在优化选择中通常可以用 2 类测试用例作为构造下一代新的测试用例的父代。一个是在每一代测试用例中，与其他测试用例相比较使程序执行了更复杂的逻辑、更多的代码的测试用例；另一个是在每一代的测试用例中，执行了和其他测试用例执行逻辑具有较大差异的测试用例。按照这个设计原则，本文的适应度函数定义为

$$fitness = \left|P_i(Z)\right| + \sum_{i=1}^{n-1}\left[D_i(Z) + N_i(loop)\right] \tag{4}$$

其中，Z 表示简化循环的路径长度；$\left|P_i(Z)\right|$ 表示一个测试用例控制流序列的 Z 路径的长度，$D_i(Z) + N_i(loop)$ 表示当前测试用例和本代其他测试用例两两比较后产生的控制流序列的差异值，其中，$D_i(Z)$ 是指 Z 路径所覆盖的控制流距离，在计算 $D_i(Z)$ 时，循环被限制为一次，$N_i(loop)$ 表示 2 个测试用例执行路径上的相同循环体的循环次数的差异值；累加求和的值表示个体测试用例在本代整个进化种群中路径差异度的情况，其中，n 表示本代测试用例共有 n 个互异个体，当前个体要和本代中的其他个体进行 $n-1$ 次比较。从式(3)中可以看出，适应度函数不仅能对 Z 路径的覆盖差异进行计算，而且能够对执行路径上的循环差异做出评估。适应度函数选择进化个体的算法如下所示。

1) 利用适应度函数计算得到适应值。

2) 根据适应值对每代测试数据的个体按适应性的高低进行排序，并设定相应的阈值。

3) 当个体测试用例的适应性超过阈值时，就被选择作为生成下一代测试数据的父代个体，然而适应性小于阈值的个体将被淘汰出进化周期。

4) 超过阈值且适应度高的个体最终会被优先选择作为父代个体。

式(4)中采用 $D_i(Z)$ 和 $N_i(loop)$ 分别作为差异值的比较部分，而没有直接求包括重复循环在内的整个序列的差异值 $D_i(S)$，主要是因为程序中的多次重复循环会急剧增加序列的长度，从而会导致增加求 D 值这一计算过程的时间和空间复杂度。另外，阈值 T 的计算过程如下所示。

1) 设适应度函数是随机变量 Y，即 Y 的取值按概率分布

$$Y = fitness(x) \tag{5}$$

2) 取每一代测试用例的适应度的均值作为阈值 T，如式(6)所示。

$$T = \mathrm{E}(Y) = \mathrm{E}(\mathit{fitness}(x)) \tag{6}$$

这样在适应度函数的指导下，一方面不断增加搜索的广度和深度，提高测试覆盖能力，另一方面，通过世代优化选择，减少了无效的重复的测试路径，反而增加了覆盖导向控制流图中那些较少走过的程序执行路径，从而提高了发现程序漏洞的几率。

2.4 应用序列比对算法求解动态控制流差异

为了计算控制流序列差异 D 值，文章参照生物信息学中比较生物分子序列相似性和差异性的方法，提出了序列比对算法。由于程序的执行路径是由跟踪点组成的序列，因此其与生物分子序列的抽象表现形式具有同构关系，从而可以运用序列比对算法来比较程序的控制流序列。

序列比较算法分为双序列比较算法和多序列比较算法。最简单的双序列比较算法就是汉明距算法，这种算法不够灵活，而且对于不等长度的序列就不能使用汉明距离来比较了。通常解决办法是在不等长度的序列中进行插入和删除操作来使 2 个序列成为等长序列，得到扩展汉明距离，但是如何查找最佳的插入和删除位置又成为解决的难题，在这种情况下就产生了 N-W[12]、S-W 等序列比对算法。

由于程序可执行路径的复杂多样性，在测试过程中跟踪到的动态控制流序列的长度和基本块调用的情况均表现复杂，例如 2 条控制流序列中间个别基本块调用存在差别，但是其他执行路径却相似性很高。解决方法是在进化选择算法中充分考虑路径的整体差异度，在求解 2 条路径差异时通过采用反映序列全局相似性的变长双序列比对算法 N-W 算法来完成。N-W 算法过程如下所示。

1) 设两序列 $s(0 \llcorner m), t(0 \llcorner n)$，其中，$s_i, t_i$ 分别表示序列 s, t 中的第 i 个字符。

2) 按下面的公式递归计算规则最大匹配矩阵 $\boldsymbol{D}[m][n]$。

$$F(s_i, t_j) = \min\begin{cases} F(s_{i-1}, t_{j-1}) + \partial(s_i, t_j) \\ F(s_i, t_{j-1}) + \partial(-, t_j) \\ F(s_{i-1}, t_j) + \partial(s_i, -) \end{cases} \tag{7}$$

其中，$F(s_{i-1}, t_j)$、$F(s_{i-1}, t_{j-1})$、$F(s_i, t_{j-1})$ 分别是 $s(0:i-1)$ 与 $t(0:j-1)$、$s(0:i-1)$ 与 $t(0:j)$、$s(0:i)$ 与 $t(0:j-1)$ 的最优比对得分；$\partial(s_i, t_i)$ 表示 s_i 与 t_i 之间的替换权值，$\partial(-, t_i)$ 表示在 t 序列中进行删除和插入等编辑操作时计分函数 ∂_{op} 的操作权值，$\partial(s_i, -)$ 表示在 s 序列中进行插入和删除等编辑操作时计分函数 ∂_{op} 的操作权值。式(8)[11]给出了计分函数的定义。

$$\partial_{\mathrm{op}} = \begin{cases} \partial(s_i, t_j), s_i = t_j \\ \partial(s_i, t_j), s_i \neq t_j \\ \partial(s_i -) = \partial(-, t_j) \end{cases} \tag{8}$$

由于序列整体的最优比对得分 $F(s, t)$ 位于矩阵右下角的位置，因此要确定 2 条序列的最优比对方式，需要从得分矩阵右下角的位置开始，按照计算矩阵时相反的方向完成回溯，即沿着产生最高序列比对得分的路径逆向移动，同时记录相应位置的序列字符，最终得到经过编辑操作的等长最优序列。

2.5 基于控制流等级的分级目标划定策略

从对个体的遗传基因编码到基于控制流比对的适应度函数设计的整个过程，都需要以测试数据个体驱动程序产生的控制流信息为基础条件，因此在实现方法中加入了基于控制流等级的分级目标划定策略，以致能够对进化搜索的目标区域进行较灵活的控制和调节。控制流等级就是指把控制流序列中抽象节点的来源等级由高到低依次划分，这一过程相当于程序由粗粒度到细粒度的逻辑划分。通常等级划分分为模块级、函数级和基本块级 3 种类型，如图 4 所示。

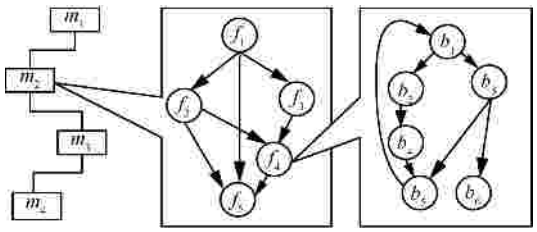

图 4 控制流的分级关系

通过层次划分，在进化程序逻辑搜索时就可以根据控制流的等级对程序的逻辑空间进行分级搜索，同时基于该策略设计出的适应度函数也能够产生不同的调控幅度，形成对搜索过程的灵活调控，达到不仅能对搜索空间进行逐级递进搜索，而且能将进化目标设定为某个区域、某个模块或某个入口函数，甚至涉及内存危险操作的某个函数等。

3 方法实现过程

下面以一个基本块的控制流序列比对分析的实例，说明方法实现中的几个关键环节。首先，解

析程序中的基本块，并将基本块作为程序动态控制流序列的基本节点，其次，将当前基本块的首地址作为该节点的编码。对于图 5 中所示程序结构，第一个基本块的首地址是 0x00401020，其对应节点的编码为 0x00401020。根据这种算法，图 5 中程序片段被解析为 6 个基本块，其所对应的控制流节点序列如图 6 所示。

```
mov   eax, dword_424110      movesi,                    push  edi
sub   esp, 8                 offset unk_424130          call  sub_402550
push  esi                    mov   ecx, [esi]           mov   ecx, [edi]
push  edi                    mov   edx, [esi+4]         add   esp, 4
movedi, [esp+10h+arg_0]      lea   eax, [esp+10h+var_8] push  ecx      ; int e
push  edi                    push  0                    call  _ftell
mov   [edi+28h], eax         push  eax                  add   esp, 4
mov   ecx, dword_424114      push  edi                  mov   [edi+4], eax
mov   [edi+2Ch], ecx         mov   [esp+1Ch+var_8], ecx push  0Offset
mov   edx, dword_424118      mov   [esp+1Ch+var_4], edx push  edi; int
mov   [edi+30h], edx         call  sub_4024D0           call  sub_4023C0
mov   eax, dword_42411C      add   esp, 0Ch             add   esp, 8
mov   [edi+34h], eax         test  eax, eax             xor   eax, eax
call  sub_402560             jl    short loc_4024BD     loc_4024BD:
add   esp, 4                 add   esi, 8               pop   edi
test  eax, eax               cmp   esi, offset unk_424168 pop  esi
jnz   short loc_4024BD       jb    short loc_40246D     add   esp, 8
                                                        retn
                                                        sub_402430 endp
```

图 5　示例程序片段

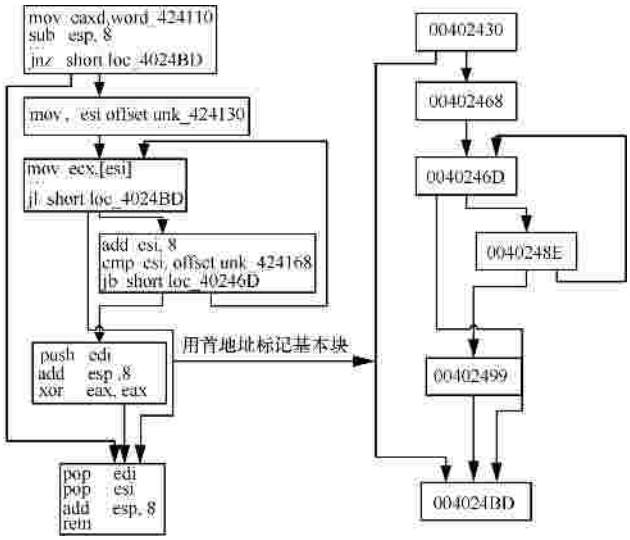

图 6　程序基本块解析并抽象成相应节点序列

最后跟踪不同测试过程获得的程序动态执行轨迹。图 7 所示 3 条序列分别表示 3 条执行轨迹，每个序列节点以该节点（基本块）的首地址标记。

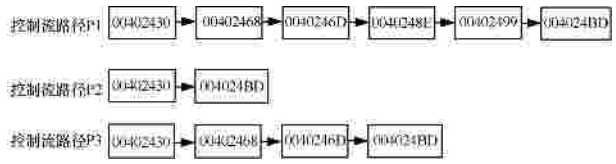

图 7　动态跟踪得到的 3 条控制流序列

利用 N-W 序列比对算法求解路径差异度，并且根据优先选择差异度较大序列的原则计算得到最优比对排列序列。在本例中，获取计分函数权值的计算公式为

$$\partial_{\text{op}}=\begin{cases}\partial(s_i,t_j)=0, s_i=t_j\\ \partial(s_i,t_j)=1, s_i\neq t_j\\ \partial(s_i-)=\partial(-,t_j)=1\end{cases}\tag{9}$$

以对比 P_1 和 P_3 的路径序列差异为例，将计分函数代入式(6)递归计算生成的得分矩阵如图 8 所示。

P_3 \ P_1	0	00402430	00402468	0040246D	0040248E	00402499	004024BD
0	0	1	2	3	4	5	6
00402430	1	0	1	2	3	4	5
00402468	2	1	0	1	2	3	4
0040246D	3	2	1	0	1	2	3
004024BD	4	3	2	1	1	2	2

图 8　用 N-W 算法对比路径序列生成的得分矩阵

图 8 中箭头指示的路径是沿着矩阵生成的路径逆向回溯得到的最优匹配路径。在计算最优匹配路径的过程中，如果回溯到的节点的位置在对角线方向上，则记录下当前节点的编码，否则就在 2 条序列中进行插入操作。

图 9 是按上述规则得到的 P_3 和 P_1 序列，其中列到这个程序片段时的执行轨迹可表示为一个由基本块组成的有序序列，经过 NW 算法两两比较重新进行编辑操作后得到的最优比对排列序列。

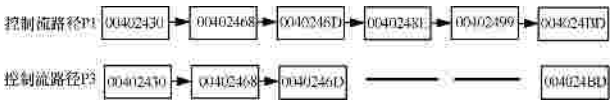

图 9　P_1 和 P_3 2 条控制流序列的比较过程

矩阵右下角的值是 P_1 和 P_3 序列整体比对的距离值，即 $D(P_1,P_3)=2$。同理可求得 P_1 和 P_2 的最优比对序列，如图 10 所示。其距离值为 $D(P_1,P_2)=4$。

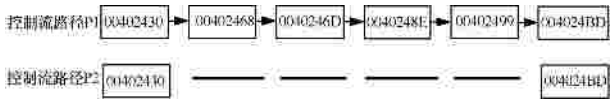

图 10　P_1 和 P_2 2 条控制流序列的比较过程

再进一步循环计算，最后根据式(3)中适应度函数计算适应值作出选择判定，产生下一代测试数据，并重复循环上述动态控制流跟踪之后的计算过程。

4 智能 Fuzzing 测试方法特性分析与性能验证

4.1 方法特性分析

基于控制序位比对的智能 Fuzzing 测试方法具有以下 3 个特性。

1) 该方法具有普适性，对源码、二进制或固件程序等不同形式的目标程序形式均适用。

2) 该方法能够达到逻辑覆盖的要求，可以对分支覆盖、循环覆盖进行评估，相对于 EFS 方法的代码覆盖更具优势。

3) 设计基于动态控制流序列差异的适应度函数，使遗传算法内在的并行性优势得以充分发挥。例如在第 3 节中提到的测试用例中，如果每代测试数据产出 100 个测试个体，独立分析每代测试数据个体的动态控制流序列内在的程序逻辑信息，只能得到 100 条信息。根据该方法，将测试用例中每条序列和其他 99 条序列两两比对差异，再利用 N-W 算法求解，其中，个体的遗传基因序列的长度 l 为 6，节点在序列中出现计为 1，不出现计为 0，然后用 $\sum_{i=1}^{99} D_i(Z)$ 进行比对差异累加，得到的适应值将最多包含 100×2^6 条信息。以上分析证明应用本文提出的方法得到的信息量远远大于从 100 个独立测试个体中所获得的信息，这相当于基于信息指导的测试扩大了 2^6 倍的并行搜索的能力。

4.2 方法性能的实验验证

符号执行 Fuzzing 测试方法和 EFS 的 Fuzzing 测试方法是当前国际 Fuzzing 测试领域中 2 种主流测试方法，本文通过实验将智能 Fuzzing 测试方法与这 2 种方法的测试性能进行比较，以验证出方法的有效性。Libpng 是一款 PNG 图像文件处理库，常用操作系统或使用广泛的应用软件在处理 PNG 格式文件时都要引用 Libpng 库文件，其运行安全性受到社会各应用领域的普遍关注。

本文以 1.4.2 版本的 Libpng 库文件（下载地址：http://sourceforge.net/projects/libpng/files/libpng14/older-releases/1.4.2/）作为测试对象，根据国际上 CVE、CWE 和 NVD 等国际上研究软件漏洞的权威组织提供的数据，1.4.2 版本的 Libpng 库文件中只在 2010 年 6 月发现一个缓冲区溢出漏洞，其 CVE 编号为 CVE-2010-1205，相应的 CWE 编号为 CWE-119。该漏洞存在于 Libpng 库文件源代码安装包的 pngpread.c 文件中，可以引发与“执行任意代码”或“拒绝服务”相关的安全威胁。

下面就以 CVE-2010-1205 漏洞进行实验。首先假设该漏洞的存在为未知信息，以 Libpng 库文件的性能标准为参考，根据上述 3 种测试方法各自的算法，任意给定测试数据集合。测试过程是应用 3 种方法分别对 Libpng 库文件进行 50 次 Fuzzing 测试，其中，每一次测试中使用的数据量逐步递增，然后计算统计 3 种方法各自的测试性能，得到每种方法整体性能的统计度量值，最后对比分析本文提出的方法与其他 2 种方法的测试性能，再进行整体上的评估。

如图 11 和图 12 指出了智能 Fuzzing 测试方法与符号执行 Fuzzing 测试方法、EFS 的 Fuzzing 测试方法整体性能的对比。

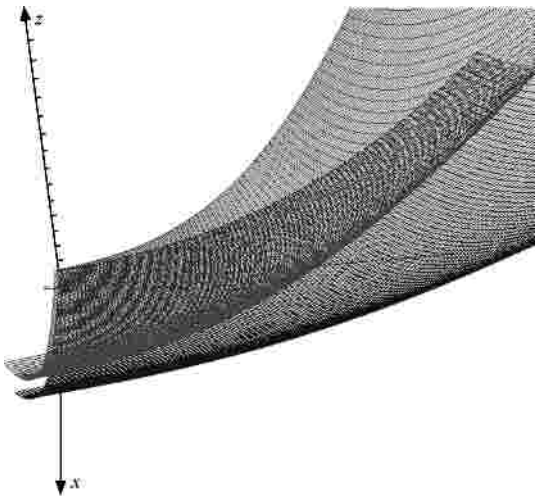

图 11 与符号执行测试方法整体性能的比较

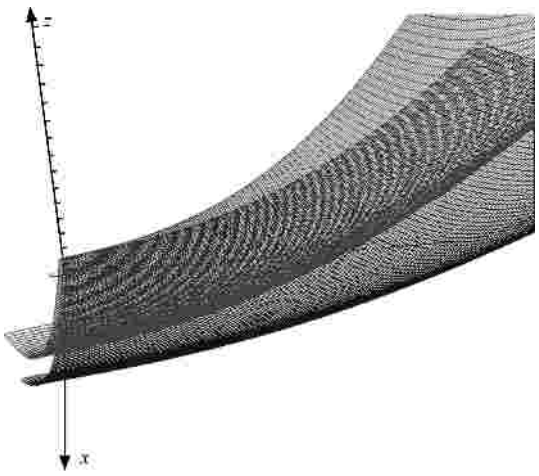

图 12 与 EFS 测试方法整体性能的比较

其中，x 坐标轴表示测试数据数量，y 坐标轴表示测试数据的递归进化次数，z 坐标轴表示需要测试的代码行的数量或称搜索空间的大小。图 11 中

下方的曲面是智能 Fuzzing 测试方法中搜索空间的整体统计度量值的解空间，图 11 中上方曲面代表符号执行 Fuzzing 测试方法中搜索空间的整体统计度量值的解空间，

图 12 中上方曲面代表 EFS 的 Fuzzing 测试方法中搜索空间的整体统计度量值的解空间。可以看到，两个图中智能 Fuzzing 测试方法所需搜索空间的整体统计度量值分别小于 Fuzzing 测试方法和 EFS 测试方法所需搜索空间的整体统计度量值，且它们之间的差值随着测试数据数量和繁殖次数的增加而加大。

根据上述实验结果，笔者认为智能 Fuzzing 测试方法的整体性能优于现有的符号执行 Fuzzing 测试方法和 EFS 的 Fuzzing 测试方法，能够提高软件程序中发现漏洞的概率，并且具有操作简单、效率高等优点。

5　结束语

本文的方法首先是建立控制序位基因模型，使得测试数据空间能够映射到程序逻辑空间，从而将测试数据的构造问题转化为空间搜索问题，其次比对每代测试数据的控制流差异，挖掘其中潜在的群体逻辑信息来构造适应度函数，并进一步使用该函数完成进化搜索，最后利用适应度函数比对群体间控制流的差异为深度搜索提供程序控制逻辑信息。

使用此方法的优势是获取的信息量远远大于对相同数量个体单独获得信息量的总和，说明基于控制流序位比对的智能 Fuzzing 测试方法能够调动遗传算法内在并行性，达到智能学习程序逻辑，加强搜索导向性的目的，从而使测试过程扩大了程序控制逻辑的覆盖范围，提高了发现漏洞的概率。

参考文献：

[1] CHOI Y H, KIM H C, LEE D H. Tag-aware text file testing for security of a software system[A]. Procedings of International Conference on Convergence Information Technology[C]. I E Press, 2007. 2254-2259.

[2] LIU Q X, ZHANG Y Q. TFTP vulnerability finding techniq based on fuzzing[J]. Computer Communications. Elsevier, 2008. 31(14): 3420-3426

[3] 姚洪波, 尹亮, 文伟平. 基于 FUZZING 测试技术的 Windows 内核安全漏洞挖掘方法研究及应用[J]. 信息网络安全, 2011, (12):9-16.
YAO H B, YIN L, WEN W P. Based on the FUZZING lead to new mining method based on windows kernel vulnerability[J]. Netinfo Security, 2011, (12):9-16.

[4] GODEFROID P, KLARLUND N, SEN K. DART: directed automated random testing[A]. Proce of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation[C]. 2005. 40-6.

[5] FROID P G, EVIN L M Y D, *et al*. Automated whitebox fuzz testing[A]. Procof Network and Distributed Systems Security(NDSS)[C]. 2008. 151-166.

[6] WANG T L, WEI T, GU G F, *et al*. TaintScope: a Checksum-aware directed fuzzing tool for automatic software vulnerabi ity detection[A]. The 31st IEEE Symposium on Security and Privacy[C]. Berkeley, California, USA , 2010.

[7] DEMOTT J, ENBODY R, PUNCH B. Revolutionizing the field of grey-box attack surface testing with evolutionary Fuzzing[EB/OL]. http://www.blackhat.com/html/bh-usa-07/bh-usa-07-speakers.html#Demott, 2012.

[8] COELLO C A, LAMONT G B, VELDHUIZEN A V. Evolutionary Algorithms for Solving Multi-Objective Problems[M]. New York: Springer-Verlag, 2007.

[9] SPARKS S, EMBLETON S, *et al*. Automated vulnerability analysis: leveraging control flow for evolutionary input crafting[A]. Proc of Computer Security Applications Conference[C]. 2007.477-486.

[10] DALLMEIER V, KNOPP N, MALLON C, *et al*. Automatically generating test cases for specification mining[J]. IEE Transactions on Software Engineering, 2012. 38(2):243-257.

[11] KANG M G, MCCAMANT S, POOSANKAM P, *et al*. DTA++: dynamic taint analysis with targeted control-?ow propagation[A]. Network and Distributed Systems Security(NDSS)[C]. 2011.

[12] NEEDLEMAN S B, WUNSCH C D. A general method applicable to the search for similarities in the amino acid sequence of two proteins[J]. Journal of Molecular Biology, 1970, 48(3):443-453.

作者简介：

王颖（1978-），女，吉林长春人，北京邮电大学博士生，主要研究方向为信息安全、网络安全。

杨义先（1961-），男，四川盐亭人，博士，北京邮电大学教授、博士生导师，主要研究方向为信息安全、网络安全、编码密码学、数字信号处理、神经网络等。

钮心忻（1963-），女，浙江湖州人，博士，北京邮电大学教授、博士生导师，主要研究方向为信息安全、网络安全、信息隐藏与数字版权管理等。

谷利泽（1965-），男，辽宁营口人，博士，北京邮电大学副教授，主要研究方向为数字签名技术与应用。