基于结构路径的恶意PDF文档检测

陈 亮[1,2] 陈性元[1] 孙 奕[1,2] 杜学绘[1,2]
(信息工程大学 郑州 450001)[1] (数学工程与先进计算国家重点实验室 郑州 450001)[2]

摘 要 恶意PDF文档依然是网络安全中的威胁,甚至造成了许多重大的安全事故。现有检测方法主要分析恶意代码提取及仿真执行两个方面,检测效率不高,缺乏对PDF文档的针对性。在分析PDF文档结构特性的基础上,定义文档结构路径,提出了一种基于恶意和正常文档之间潜在的结构差异特性的检测方法。大量实验数据结果表明,本方法在检测准确率和检测速率方面都有不错的表现。

关键词 恶意软件检测,PDF文档,结构路径,决策树
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.2.019

Detection of Malicious PDF Based on Structural Path

CHEN Liang[1,2] CHEN Xing-yuan[1] SUN Yi[1,2] DU Xue-hui[1,2]
(The PLA Information Engineering University, Zhengzhou 450001, China)[1]
(State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China)[2]

Abstract Malicious PDF document is still a network security threat, and even causes a number of significant security incidents. The existing methods mainly analyse malicious code extraction and simulation execution. The detection efficiency is not high. On the basis of analyzing the structural properties of PDF documents structure, a structure path was defined and a detection method based on the structure of the potential difference between the characteristics of malicious and benign documents was proposed. A large number of experimental data results show that the method has a good performance on the detection accuracy rate and detection speed.

Keywords Malware detection, PDF documents, Structural path, Decision tree

1 引言

随着信息化及办公自动化的发展,PDF(Portable Document Format)文档日益成为流行的数据交换、信息共享的文件格式。2008年以前,针对PDF文档的恶意代码比较少,相应的文档漏洞也比较少,其检测方式还主要集中于特征码扫描阶段。随着PDF应用范围的扩大,其漏洞也逐渐增多,同时由于恶意office文档的检测技术已相对成熟,PDF已经代替office成为恶意代码的有效载体[1]。

尽管目前大多数的杀毒软件都支持恶意PDF文档的检测,但这种防护技术仍然是基于检验和计算或者启发式技术。启发式技能虽然在一定程度上能够识别部分已经存在的攻击的变种,但仍然不足以及时发现新的攻击手段,从而使得恶意PDF文档的检测准确率和效率一直不高,因此亟需快速有效的检测方法。

现有的针对恶意PDF文档检测的方法可以分为3种:动态分析、静态分析和动态与静态相结合的分析方法。动态分析方法可以划分为两种,一种是基于ShellCode的检测方法,2005年,P. Akritidis[2]利用软件仿真的方式,模拟受控制的ShellCode的运行;2010年,M. Polychronakis[3]在仿真ShellCode的基础上,结合启发式分析的方法,从而能够检测一定程度上的变种攻击。但是软件仿真的方法并不能够模拟所有的指令集合,因此通过规避模拟的指令集同样能够达到攻击目的。为了解决上述问题,2011年,K. Z. Snow[4]提出了一种基于硬盘虚拟化的ShellOS系统,ShellOS系统通过对操作系统内核的仿真能够有效地检测由应用程序指定的任意内存缓冲区的ShellCode,提高了检测正确率。然而,由于检测过程是在内存缓冲区执行的,这就要求应用程序在检测之前为其分配相应的缓冲区,从而检测效率比较低。另一种是基于JavaScript的检测方法,2010年,M. Cova[5]利用启发式技术来训练正常JavaScript模型以检测攻击技术;2010年,K. Rieck[6]则建立了专门的JavaScript沙盒,在沙盒中运行的PDF文档能够自动学习影响JavaScript编译器状态的一系列事件。

一般来说,动态分析技术依赖于恶意代码的执行,正因如此,它促进了静态分析技术的发展。之前的静态分析方法集中于n-gram分析[7,8],这种分析方法可以应用于所有文档内容的分析,并没有关注于PDF格式的特点,比如编码、压缩方

到稿日期:2014-03-06 返修日期:2014-06-23 本文受国家863项目高技术研究发展计划(2012AA012704),河南省科技创新人才计划(114200510001),信息保障技术重点实验室开放基金课题(KJ-13-110)资助。

陈 亮(1991—),男,硕士生,主要研究方向为网络与信息安全,E-mail:yixiu199151@sina.com;陈性元(1963—),男,博士,教授,博士生导师,主要研究方向为网络与信息安全;孙 奕(1979—),女,博士生,讲师,主要研究方向为网络与信息安全;杜学绘(1968—),女,博士,教授,博士生导师,主要研究方向为网络与信息安全。

式、加解密技术。2011 年，P. Laskov 和 N. Srndic[9] 提出了第一个关注于 JavaScript 内容的静态分析系统 PJSCAN，出于效率的考虑，PJSCAN 的 JavaScript 提取模块只是查找了 PDF 标准[10] 中描述的 JavaScript 应该存在的位置。这种提取策略安全性并不高，攻击者可以利用 JavaScript API 和像 eval() 的函数调用，将 JavaScript 放到任意的位置中去。2012 年，D. Maiorca[11] 等提出的 MALWARE SLAYER 系统，通过利用 PDFID 工具抽取 PDF 文档中文本关键，然后基于模糊识别的方法进行分析。该系统在检测率和误报率方面有良好的表现，但是它的表现取决于 PDFID 抽取函数的性能，并且不能提取目标流中的信息。

2011 年，Z. Tzermias[12] 等人提出了第一个用于分析恶意 PDF 文档的静态与动态分析相结合的 MDSCAN 系统。MDSCAN 静态分析模块是一个特定的语法分析程序，这个分析程序试图提取文档中的所有 JavaScript 信息，包括从交叉引用表中漏掉的目标以及潜在的变形目标，作为其动态执行的先决条件。提取的 Scripts 将在仿真的 Acrobat Reader 引擎中执行，在受控的执行过程中，所有的内存缓冲区都将被基于二进制仿真的 ShellCode 检测工具检查。同年，C. Curtsinger[13] 等人提出了一个针对浏览器内嵌 JavaScript 的分析系统 ZOZZLE。ZOZZLE 中的动态分析模块用于提取 Internet Exploerer 中 JavaScript 引擎中的 JavaScript 部分，而静态分析模块则利用贝叶斯分类方法，根据语法来分析检测到的 JavaScript 代码。

综上所述，动态分析系统虽然能够提供更好的检测正确率和误报率，但是这是基于大量时间开销以及虚拟化特定的指令系统的。目前大多的静态分析系统都是在提取分析其中的 JavaScript 内容基础上进行的，这与现在 PDF 文档漏洞现状相符合。但是，存在两个技术难题使得基于 JavaScript 内容的检测变得十分困难。第一个难题是 PDF 文档中 JavaScript 内容的定位。JavaScript 代码可能隐藏在 PDF 文档的逻辑结构中，甚至位于 PDF 标准所规定之外的其他位置。通过利用 eval() 函数，PDF 文档中的任何文本内容都能够解释成 JavaScript 语言，因此并不能阻止攻击者将大块的恶意 JavaScript 代码分散到文本中，等到其运行时再组合起来。第二个难题在于 JavaScript 语言的高程度表现力，这给攻击者提供了强大的代码混淆平台。尽管目前已提出了一些混淆的 JavaScript 代码的检测技术[14]，然而它们的实际应用效果并不明显。

2012 年，C. Smutz 和 A. Stavrou[15] 首先提出了一种不依赖于 JavaScript 内容的恶意 PDF 文档检测方法，该方法基于 PDF 文档元数据及结构特性，在大数据处理过程中有着出色的分类性能，但是同样不能提取目标流中的隐藏信息。2013 年，Nedim Srndic 和 Pavel Laskov[16] 提出了一种基于分级结构的恶意 PDF 文档检测方法，但是结构路径定义过程不够精确，结构路径元素的选取过于主观。本文在分析上述方法的基础上，提出了一种基于结构差异比对的恶意 PDF 文档检测技术，形式化定义 PDF 文档的结构路径，分析训练数据中抽取的正常文档和恶意文档结构路径的差异。大量实验数据结果表明，本方法在检测成功率和误报率方面都有不错的表现。

2 PDF 文档结构

PDF 文档的语法由以下 4 个元素组成：(1) 对象：组成 PDF 文档的基本元素；(2) 物理结构：指明了 PDF 文档中的对象分布及是否被修改过；(3) 逻辑结构：表明了组成 PDF 文档内容对象的逻辑分布；(4) 内容流：提供了文档内容的有效存储方式。

下面本文将对一个简单 PDF 文档的内容进行分析。

PDF 文档的物理结构如图 1 所示。每个 PDF 文档必须以标明其为 PDF 文档的一行代码（即幻数）开头，并且规定了用于描述此文档的 PDF 语言规范版本号：%PDF-1.1。其中 10 object 中，1 是对象序号，用来唯一标识一个对象；0 是生成号，根据 PDF 规范，若 PDF 文档被修改，则此数字累加，但在实际应用中很少使用这种方式修改 PDF，都是重新编排对象号；obj 和 endobj 是对象的定义范围。Catalog 字典对象表明为了装配这个文档，需要从哪里开始查找对象。Catalog 给出了在这个 PDF 中能找到的目录（Outlines）、页面（Pages）和动作执行（OpenAction）。其中 OpenAction 是一个典型的字典嵌套，包含两个元素：S，名字型对象，表明这是一个 JavaScript 执行字典；JS，字符串型对象，包含实际要执行的 JavaScript 脚本：alert('Hello!')。Outlines 和 Pages 都是间接对象，分别以对象标识符 20 和 30 表示。Outlines 字典对象包含一个整型对象 Count，表示这个 PDF 文档没有目录。Pages 字典对象包含整型对象 Count 和数组对象 Kids。虽然简单的 PDF 文档大致由上述对象组成，但是仅有这些内容还不足以使阅读程序读取和显示，还需要知道文档描述起始于哪个对象（即 root 对象）及每个对象的索引。交叉引用表就是为了使间接对象能够随机存取而设立的一个地址索引表。

```
%PDF-1.4
1 0 obj <<
 /Type /Catalog
 /Outlines 2 0 R
 /Pages 3 0 R
 /OpenAction <<
  /S /JavaScript
  /JS (alert('Hello!');)
 >>
>> endobj

2 0 obj <<
 /Type /Outlines
 /Count 0
>> endobj

3 0 obj<<
 /Type /Pages
 /Kids [4 0 R]
 /Count 1
>>endobj

4 0 obj<<
 /Type /Page
 /Parent 3 0 R
 /MediaBox [0 0 612 792]
 /Resources...
>> endobj

xref
0 4
0000000000 65535 f
0000000012 00000 n
0000000070 00000 n
0000000134 00000 n
trailer
/Size 4 /Root 1 0 R
startxref

530

%%EOF
```

图 1 PDF 文档物理结构

PDF 文档的逻辑结构大体上是一个树状结构。根结点为 Catalog 字典，这是 PDF 文档物理结构和逻辑结构的结合点，也是解析的起点。其结构如图 2 所示。

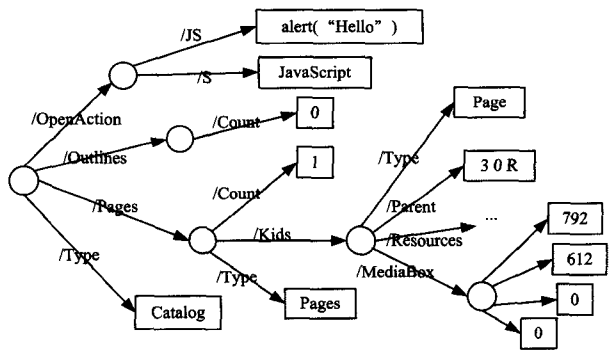

图 2 PDF 文档逻辑结构

3 基于结构路径的恶意 PDF 文档检测方法

基于结构路径差异的恶意 PDF 文档检测方法主要包括两大步骤：

1. 结构路径的抽取：分析 PDF 文档物理和逻辑结构，并从中抽取出具有代表性意义的结构路径，是检测是否为恶意文档的基础。

2. 学习检测过程：通过对训练数据的学习，构建检测恶意 PDF 文档的模型。然后将模型应用于实验数据判断模型的有效性。

系统检测流程如图 3 所示。

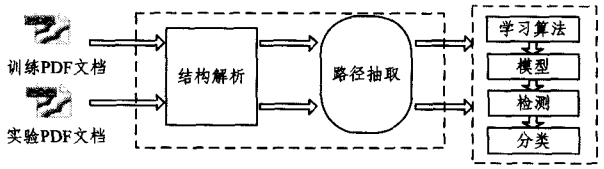

图 3　系统检测流程

3.1 结构解析及路径抽取

PDF 文档的树状结构能够用一系列的从根结点到叶子结点的路径表示。然而由于 PDF 文档中间接对象的大范围使用，不同的结构路径最终可能会指向相同的对象，甚至有可能形成环路。因此，结构路径选取的关键在于找出其中所有父结点和子结点之间的关系。在本文中定义结构路径的抽取过程满足以下规则：

规则 1　路径的抽取过程中元素个数及数目必须准确；

规则 2　算法必须是可重复和健壮的，即对于同一 PDF 文档，算法每次运行都能生成相同的结果；

规则 3　在可能存在的结构路径中，所选取的结构路径必须是最有意义的。

首先第一步需要完成的是对 PDF 文档的解析，在本系统中我们采取开源的 Poppler 作为结构解析工具。Poppler 内置对 PDF 文档的语法分析程序，能够访问文档中任何的对象元素。结构路径的抽取相当于对文档结构从 Catalog 字典对象到叶子结点的递归计数。

系统在结构路径的定义过程中采取广度优先搜索算法，算法自始至终一直通过已找到和未找到顶点之间的边界向外扩展，算法首先搜索与 S 距离为 k 的所有顶点，然后再去搜索与 S 距离为 $k+l$ 的其他顶点。因此，广度优先搜索算法能够遍历到结构中的每一个顶点，满足规则 1 的约束。规则 2 和规则 3 的满足依赖于算法对间接对象的处理，广度优先搜索算法中对已搜索过的顶点标记及出现环路的回溯能够很好地解决间接对象所引发的环路问题，从而满足规则 2 和规则 3 的约束。

根据第 2 节中 PDF 文档的物理结构和逻辑结构，可以定义出反映其特性的结构路径，如图 4 所示。

/Type:1	/Metadata
/OpenAction/S:1	/Type
/OpenAction/JS:1	/OpenAction/S
/Outlines/Count:1	/OpenAction/JS
/Pages/Count:1	/Outlines/Count
/Pages/Kids/Type:1	/Pages/Count
/Pages/Kids/Parent:1	/Pages/Kids
/Pages/Kids/...	...

图 4　PDF 文档结构路径

3.2 决策树算法

当结构路径的抽取完成之后，需要为给定的训练数据选取合适的学习算法训练模型，以及利用模型对未知的检测数据进行分类。常用的机器学习算法都能够在本实验中应用。本文不侧重于去比较哪种机器学习算法能够提供更好的检测性能。由于决策树是一种广泛应用、易于理解和实现预测模型，代表的是对象属性和对象值之间的映射关系，分类模型为树状结构，适合于训练数据集比较大的情况，因此本文选取决策树作为学习算法。典型的决策树算法示例如图 5 所示。

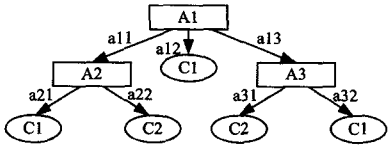

图 5　决策树算法示例

决策树算法有许多种形式，如 CART[21]，RIPPER[22]，C4.5[23]等。我们选取目前比较新的能够提供自动交叉验证、加权分类等实用功能的 C5.0 算法。

C5.0 算法以信息熵的下降速度作为确定最佳分支变量和分割阈值的依据。设 S 是抽取结构路径的样本集合，其中元数据类型变量 C 有 K 个分类，$freq(C_i,S)$ 表示属于 C_i 类的样本数，则结构路径集合 S 的信息熵定义为：

$$Info(S)=-\sum_{i=1}^{k}((freq(C_i,S)/|S|)\times\log_2(freq(C_i,S)/|S|))$$

由于 PDF 文档结构中间接对象的使用，对于结构路径中存在的属性变量 T，有 N 个分类，则属性变量引入后的条件熵定义为：

$$Info(T)=-\sum_{i=1}^{n}((|T_i|/|T|)\times Info(T_i))$$

属性变量 T 带来的信息增益为：

$$Gain(T)=Info(S)-Info(T)$$

而 C5.0 算法采取信息增益率来侵害结点，即

$$Gainration(A)=Gain(A)/Info(A)$$

其中，$Gain(A)$ 是在假设情况 A 下子结点的信息增益，$Info(A)$ 是子结点个数指标，分割后的子结点个数越多其值就会越大，相应的信息增益率则会减少。

利用上式对结构路径产生中所有的假设情况进行计算，按照信息增益率最大的情况产生子结点。完成树的生长后，C5.0 算法采用基于树规则的方法实现剪枝[24]。

4 特点分析

以上基于结构路径的恶意 PDF 文档检测方法兼顾了检测准确率和时间开销，检测模型的复杂度适中，具有较强的实用性，其特点主要有以下几个：

1）本方法将 PDF 文档结构作为分析的重点，与文献[7-11]相比，不需要对 ShellCode 及 JavaScript 代码定位，从而更加精确地描述正常文件与恶意文件之间的差异，因此具有更高的检测准确率。

2）同文献[4-6]动态检测方法相比，本方法不依赖于程序的动态执行，不需要分配相应的内存空间，因而时间开销小。

3）文献[15,16]从元数据分级结构特征出发，分析其元数据出现频率及重要程度判断是否为恶意文档。与本方法相比，缺乏对数据特征的统计，并且其元数据结构存在大量的冗

余数据，效率不高。

5 实验设计与结果分析

我们利用 VirusTotal[25] 收集的大量数据对以上检测方法的性能进行了实验。VirusTotal 是一个包含 47 个杀毒引擎、供免费的可疑文件分析服务的网站。与传统杀毒软件的不同之处是它通过多种杀毒引擎扫描文件。使用多种反病毒引擎可以令用户通过各杀毒引擎的侦测结果，判断上传的文件是否为恶意软件。实验数据按时间顺序分 3 批次获取，共包含 109064 份 PDF 文档，约 78GB，如表 1 所列。

表 1 PDF 文档实验数据集

	09. Oct. 2013		13. Nov. 2013		15. Dec. 2013	
	Malicious	begin	Malicious	begin	Malicious	begin
Dataset size	2. 7GB	20GB	2. 4GB	23GB	2. 9GB	27GB
Files in the dataset	22576	12453	21301	13458	25427	15849
Average file size	0. 12MB	1. 64MB	0. 11MB	1. 75MB	0. 12MB	1. 74MB

从表 1 中可以看出，恶意文档比正常文档的体积要小得多。恶意 PDF 文档一般不包含有意义的内容信息，从而会造成相应结构路径的缺失。在实验过程中，将获得的 3 个数据集合并，采取 10 层交叉检验的方式进行实验。10 层交叉检验就是把原始的数据随机分成 10 个部分。在这 10 个部分中，选择 1 个作为测试数据，剩下的作为训练数据。实验结果如表 2 所列。

表 2 交叉检验实验结果

	FP Rate	TP Rate	FP Count	TP Count
1	0	0. 9873	0	7892
2	0	0. 9974	0	8213
3	0. 0011	0. 9842	4	8034
4	0. 0023	0. 9924	10	7997
5	0. 0031	0. 9917	14	8122
6	0. 0017	0. 9984	7	8241
7	0. 0049	0. 9972	25	8023
8	0. 0099	0. 9963	50	7987
9	0. 0121	0. 9911	61	7964
10	0. 0203	0. 9884	103	10900

此外，我们利用以上实验数据对文献[4,9,11,12]中检测方法的性能进行了实验。表 3 给出了针对 5 种检测方法的实验结果。

表 3 5 种检测方法对应的实验结果

Performance Index	FP (%)	TP (%)	T(s)	f/p(s)
The Results of the Method in This Paper	0. 0001	0. 9923	1837	0. 023
The Results of the Method in Reference[12]	0	0. 8974	1564	0. 019
The Results of the Method in Reference[9]	0. 0011	0. 7284	2347	0. 029
The Results of the Method in Reference[4]	N/A	0. 8024	1932	0. 024
The Results of the Method in Reference[11]	0. 0251	0. 9955	2430	0. 030

由表 3 的实验结果可见，本文提出的基于结构路径的恶意文档检测方法检测准确率高于目前已知文献的检测方法。表中的时间开销是指实验中训练和检测所需要的时间，能在一定程度上反映检测的实时性。根据实验结果，本文中检测方法的计算时间多于文献[12]，少于其他文献的检测方法。因此，本文中检测方法的综合性能优于现有已知检测方法。

结束语 针对 PDF 文档可能夹带有恶意代码，提出了一种基于恶意文档和正常文档之间潜在结构差异特性的检测方法。通过对 PDF 文档结构的解析，定义结构路径，然后利用决策树 C5. 0 算法建立训练模型，并利用 VirusTotal[25] 提供的大量数据进行了实验。实验结果表明，本方法在检测准确率及时间开销上都有着很好的表现。

参 考 文 献

[1] 武雪峰. 恶意 PDF 文档的分析[D]. 济南：山东大学，2012

[2] Akritidis P, Markatos E, Polychronakis M, et al. STRIDE: Polymorphic sled detection through instruction sequence analysis[C]// 20th International Conference on Information Security. 2005: 375-392

[3] Polychronakis M, Anagnostakis K, Markatos E. Comprehensive shellcode detection using runtime heuristics[C]// Annual Computer Security Applications Conference (AC-SAC). 2010: 287-296

[4] Snow K Z, Krishnan S, Monrose F, et al. ShellOS: Enabling fast detection and forensic analysis of code injection attacks[C]// USENIX Security Symposium. 2011

[5] Cova M, Kruegel C, Vigna G. Detection and analysisof drive-by-download attacks and malicious JavaScript code[C]// International Conference on World Wide Web (WWW). 2010: 281-290

[6] Rieck K, Krüger T, Dewald A. Cujo: Efficient detection and prevention of drive-by-download attacks[C]// An-nual Computer Security Applications Conference (ACSAC). 2010: 31-39

[7] Li W-J, Stolfo S, Stavrou A, et al. A study of malcode-bearing documents[C]// Detection of Intrusions and Malware & Vulnerability As-sessment (DIMVA). 2007: 231-250

[8] Shafiq Z, Khayam S, Farooq M. Embedded malware detection using markov n-grams[C]// Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA). 2008: 88-107

[9] Laskov P, Srndi'c N. Static detection of malicious JavaScript-bearing PDF documents[C]// Annual Computer Security Applications Conference (ACSAC). 2011: 373-382

[10] PDF Reference[OL]. http://www. adobe. com/devnet/pdf/pdfreference. html, 2008

[11] Maiorca D, Giacinto G, Corona I. A pattern recognition system for malicious pdf files detection[J]. Lecture Notes in Computer Science, 2012, 7376: 510-524

[12] Tzermias Z, Sykiotakis G, Polychronakis M, et al. Combining static and dynamic analysis for the detection of malicious documents[C]// European Workshop on System Security (EuroSec). 2011

[13] Curtsinger C, Livshits B, Zorn B, et al. ZOZZLE: Fast and precise in-browser JavaScript malware detection[C]// USENIX Security Symposium. 2011: 33-48

[14] Kaplan S, Livshits B, Zorn B, et al. "nofus: Automatically detecting"+string. fromcharcode(32)+"obfuscated ". tolowercase()+"javascript code"[R]. Technical Report, Microsoft Research, 2011

[15] Smutz C, Stavrou A. Malicious PDF detection using metadata and structural features[C]// Annual Computer Security Applications Conference (ACSAC). 2012

[16] Detection of Malicious PDF Files Based on Hierarchical Document Structure[C]// Proceedings of the Network and Distributed System Security Symposicum(NDSS). 2013

[17] Lee W, Stolfo S, Mok K. A data mining framework for building intrusion detection models[C]// IEEE Symposium on Security and Privacy. 1999:120-132

[18] Mahoney M, Chan P. Learning rules for anomaly detection of hostile network traffic[C]// International Conference on Data Mining (ICDM). 2003

[19] Gu G, Porras P, Yegneswaran V, et al. BotHunter: Detecting malware infection through IDS-driven dialog correlation[C]// USENIX Security Symposium. 2007:167-182

[20] Canali D, Cova M, Vigna G, et al. Prophiler: a fast filter for the large-scale detection of malicious Web pages[C]// International Conference on World Wide Web (WWW). 2011:197-206

[21] Breiman L, Friedman J, Olshen J, et al. Classification and Regression Trees[M]. Wadsworth, 1984

[22] Cohen W. Fast effective rule induction[C]// International Conference on Machine Learning (ICML). 1995:115-123

[23] Quinlan J. C4. 5: Programs for Machine Learning[M]. Morgan Kaufmann, 1992

[24] Duda R O, Hart P E, Stok D G. 模式分类[M]. 李宏东, 姚天翔, 等译. 北京: 机械工业出版社, 2003

[25] https://www.virustotal.com/

(上接第 85 页)

注意到式(7)和定义 4(i),有 P' 亦处于终止状态。故由式(8)可知,R' 亦处于终止状态,与假设矛盾。

故 $Q' \parallel R'$ 处于终止状态。

(d)Q' 不处于终止状态,而 R' 处于终止状态。

注意到式(8),有 P' 亦处于终止状态,由定义 4(iv)知,Q' 亦处于终止状态,与假设矛盾。

故 $Q' \parallel R'$ 处于终止状态。

综合(a)—(d)得到,与 $Q \parallel R$ 是错误的假设矛盾。故,如果 $P \parallel R$ 是正确的,并且 $Q \Downarrow R$,那么也 $Q \parallel R$ 是正确的。证毕。

定理 2 用修改后的 CCS 进程 P 和 Q 表示 Web 服务 S_1 和 S_2。R 表示 Web 服务 S_1 的合成环境。如果 $P \parallel R$ 的服务或功能是成功进行的,并且 $Q \Downarrow \kappa R$,那么 $Q \parallel R$ 的服务或功能是 κ 可成功进行的。

证明:由定义 5 和引理 1 知,结论显然成立。证毕。

推理 1 假定 $Q \Downarrow \kappa R$。当 $\kappa > 0$ 时,那么 Q 对 P 的降级替换是有意义的。

证明:由 $\kappa > 0$ 及式(5)可知,存在 k_0,使得

$$\prod_{j=1}^{m_{k_0}} \mu_{kj} \cdot (1 - \chi_{t_{k_0 j}}(s_{k_0 j}^{\wedge})) > 0$$

即,至少存在一条路径可以实现原有服务系统中的一种服务或功能。由注 8 可知结论成立。证毕。

结束语 本文主要讨论了 Web 服务降级替换的一致性问题,并从量化角度分析了 Web 服务的降级替换:

· 以 3 种订票服务系统(正常系统、带延时的系统和某些服务不能提供的系统)为例,讨论了以原有进程代数研究服务系统的降级替换合成正确性时会碰到困难,而且原有的降级替换理论也无法精确地从量化角度给出服务降级替换的程度。于是,我们修改原有的进程代数,引入了超时处理算子和延时处理算子。

· 基于修改后的进程代数,给出了降级一致性定义和降级服务的替换度定义,保证了服务降级替换的合成正确性,进一步地从量化角度对 Web 服务的降级替换进行了讨论。

接下来,我们会在本文工作的基础上进一步考虑所给出的理论模型在具体实际中的应用。

参 考 文 献

[1] 刘克, 单志广, 王戟, 等. "可信软件基础研究"重大研究计划综述[J]. 中国科学基金, 2008(3):145-151

[2] Knight J C, Strunk E A, Sullivan K J. Towards a Rigorous Definition of Information System Survivability[C]// 3rd DARPA Information Survivability Conference and Exposition (DISCEX 2003). 2003, 1:78-89

[3] 余智华, 林思明, 陈海强. 网络安全——可生存性研究及网络建模[J]. 信息技术快报, 2005, 3(12):11-23

[4] Nagappan R, Skoczylas R, Sriganesh R P. Developing Java Web Services[M]. Wiley, 2002

[5] Dumas M, Yang Y, Zhang L. Improving Web Service Survivability Via Gracefully Degraded Substitution[C]// 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. Toronto, Ontario Canada, 2010: 597-600

[6] Liu F F, Shi Y S, Zhang L, et al. Analysis of Web Services Composition and Substitution Via CCS[C]// Proceedings of the DEECS'06. San Francisco, CA, USA, Springer-Verlag, Lecture Notes in Computer Science, 2006, 4055:236-245

[7] Bourouz S, Zeghib N. Verifying Web services substitute ability using open colored nets reduction techniques[C]// 2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO). IEEE, 2013:1-5

[8] 郭峰, 魏光. 基于 Petri 网的 Web 服务描述及其可替换性分析[J]. 计算机集成制造系统, 2013, 19(6):1423-1432

[9] Nakajima S. Safe Substitution of Components in Self-adaptive Web Applications[C]// 2013 20th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2013, 1:388-395

[10] 宋仲凯, 张晓容, 殷昱煜. QoS 驱动的服务动态替换方法[J]. 计算机应用与软件, 2012, 29(1):27-30

[11] 刘方方, 史玉良, 张亮, 等. 基于进程代数的 Web 服务合成的替换分析[J]. 计算机学报, 2007, 30(11):2033-2039

[12] 史玉良, 王海洋, 张亮, 等. Web 服务合成的相容性和替换性分析[J]. 计算机研究与发展, 2007, 44(11):1955-1961

[13] 刘莹, 张一川, 张斌, 等. 基于行为效果的服务可替换性分析[J]. 计算机研究与发展, 2010, 47(8):1442-1449

[14] W3C Working Group. Web Services Choreography Interface (WSCI) 1.0 [Z]. World Wide Web Consortium, W3C Note 8 Aug. 2002. URL. http://www.w3.org/TR/wsci

[15] Brogi A, Canal C, Pimentel E, et al. Formalizing Web Services Choreographies[J]. Electronic Notes in Theoretical Computer Science, 2004, 105:73-94