一种基于环境约束的异常程序行为模型

何加浪[1] 徐 建[1,2] 张 宏[1]
(南京理工大学计算机科学与技术学院 南京 210094)[1]
(南京大学计算机软件新技术国家重点实验室 南京 210093)[2]

摘 要 在控制流模型中加入影响程序运行的环境因素,结合静态分析方法的优点,建立了程序异常行为分析模型,用以标记函数调用指令,并在动态运行时进行返回值一致性约束,从而解决了一般方法回避函数指针导致的间接调用问题。同时根据程序的局部运行原理,将分析范围限定在函数范围内。实验结果表明模型具有较好的精确性和较低的性能影响。

关键词 控制流,运行环境,程序行为
中图法分类号 TP309.7 **文献标识码** A

Abnormal Behavior Model Based on Environment Constraint

HE Jia-lang[1] XU Jian[1,2] ZHANG Hong[1]
(Institute of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China)[1]
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)[2]

Abstract Including environmental factors for the procedure operation in the control-flow model, combining with the advantages of static analysis methods, this paper established the model for analysis of procedure behaviors. It marked function call instructions, using consistency constraint to return value to overcome the indirect call problems that the general method produces for avoiding the function pointer in a dynamic run-time. At the same time, using the local principle of the procedure limited the scope of the analysis in the functions. The experimental results show that the model has good accuracy and lower performance impact.

Keywords Control flow, Operating environment, Program behavior

1 引言

随着信息技术的飞速发展,对软件的依赖性越来越强。而软件的规模不断扩大,复杂性不断提高,由于程序本身的缺陷和外界攻击导致软件产生异常,从而引发损失甚至灾难的案例在不断增加。目前对可信软件的研究是一个热点,但是对于保证软件按照人们期望的行为运行的理论和方法还不成熟,因而对软件进行动态行为分析,及早发现软件运行时的异常,减轻或避免损失和灾难的发生就变得十分重要。

传统的方法通常对程序运行后留下的日志、审计数据等进行分析,检测程序异常,这类方法实时性较差。近年来的研究较为集中在对程序行为的建模和监控方面,如系统调用序列模型、控制流模型、数据流模型等,该类方法的特点是可以动态地监控运行时刻的程序,及时发现异常。系统调用是程序和操作系统的交互,程序运行时会留下特定的系统调用轨迹,对表征程序的行为十分有意义,如神经网络模型、Markov模型等都是围绕着系统调用进行建模分析程序运行时的行为。

控制流模型通过学习获取软件运行的控制流信息,对于软件运行时的控制流进行监控,通过静态分析源码生成控制流模型或者通过分析程序运行时刻信息生成模型,然后约束程序实际执行流程和生成的控制流模型相一致。这类方法实时性很强,但是对于一些不改变程序控制流的异常不能很好地监控,而基于数据流的模型可以在一定程度上检测到这些异常。但是现在的控制流模型除静态分析源码的方法外,都具有学习阶段模型的输入不具有一般性、覆盖率和收敛速度不稳定等问题,因而存在很高的误报。本文提出一种基于标记的控制流程序行为异常分析模型,通过在学习阶段输入约束模型获取较高的覆盖率和收敛速度,提高模型的表达能力。

本文第2节是相关工作;第3节介绍模型和相关算法;第4节是实验和分析;最后是总结和下一步工作。

2 相关工作

Forrest[1]最早根据生物免疫原理通过"自我"和"非我"的区分来分析程序的异常行为,指出系统调用序列对于刻画程序的"自我"非常有意义。接着有很多研究者对基于系统调用的方法做了重要的改进。在相关文献中[2,3],通过Markov模型、神经网络模型在很大程度上改进了学习的泛化程度和学

到稿日期:2009-07-01 返修日期:2009-09-30 本文受国家自然科学基金重大研究计划项目(No. 90718021)资助。
何加浪(1984—),男,博士生,主要研究方向为软件分析,E-mail:hejl@mail.njust.edu.cn;徐 建(1979—),男,博士,讲师,主要研究方向为软件工程、可信软件;张 宏 男,教授,博士生导师,主要研究方向为信息安全。

习精度，使模型具有很强的表达能力。部分研究者在尽量保证模型精度的同时，在减少系统开销方面也做了重要的工作[4]。而一部分研究者开始注意到程序的函数调用关系是描述程序行为的重要信息。Wagner[5]通过分析源码获取程序的控制流保证了控制流的精确性，然而由于其分析的开销很大，无法获取程序运行时由于动态输入产生的控制流关系，同时存在数据版权方面的问题。Gao[6,7]等采用灰盒的方法动态获取程序的执行图，通过获取程序运行时调用栈中的返回地址构建程序执行图来逼近程序真实控制流图，具有不依赖源码和可执行二进制文件的特点，但是由于是在系统调用点采集调用栈中的信息，因此存在遗漏部分函数调用信息的问题，无法获取完整的程序执行轨迹。在文献[8]中，比较了相关的模型，证明了控制流模型具有更精确的表达能力。

本文在考虑程序运行时控制流的的同时，考虑到程序的局部性原理，引入相对于环境的约束，建立程序运行时分析模型，在很大程度克服了上述了模型的缺点，并通过实验验证模型的有效性。

3 模型和算法

本文提出环境约束的控制流模型 ECCFM(Environment Constraint Control-Flow Model)。Sharif[8]等证明了对于任何基于系统调用的程序分析模型，总存在更精确的控制流模型，同时指出将系统调用序列和控制流相结合的混合模型存在冗余检查。因此，本文根据 Sharif 等的结论，提出了一种基于环境约束的控制流模型，其新颖性主要体现在以下方面。

1)考虑程序运行环境，利用控制流节点和特定资源的关联特征对特定环境进行标记，以此适应软件运行环境的动态变化；

2)考虑到实际应用中软件数据流的相关性，通过对系统调用参数进行控制流图节点关联，同时根据程序的局部性原理，程序在时间和空间上都具有局部性，因而将参数的关联性限定在一个局部内，提出局部学习算法来获取较好的收敛速度和覆盖率。

3.1 ECCFM 模型的建立

程序控制流的变化主要是由于跳转指令引起的。对于函数调用指令，由于函数执行结束后会返回调用点，本文不把其归为跳转指令，而作为一种特殊的标记指令在 PDA 模型中进行处理，具体作法在后面章节中讨论。为便于描述，引入如下定义。

定义 1(基础块) 不含有跳转指令的顺序执行的程序片段，对于 CALL 指令，视为顺序执行指令进行标记。

首先对单个函数 f_i 建立 NFA 模型：

$NFA_{f_i}=\{K,\Sigma,\delta,q_0,F\}$

式中，K 为有限的状态集合，Σ为有限输入字母表，q_0 是 K 中的一个开始状态，$F\subseteq K$ 为结束状态集合，δ 为 $K\times\Sigma\rightarrow K$ 的一个多值映射，即 $\delta(q,\alpha)=\{p_1,p_2,\cdots,p_k\},k\geqslant 1\wedge k\in N$。

在建立单个函数 NFA 时，状态集合元素为基础块，用该基础块的第一条指令地址标识；输入字母用跳转指令及其地址标识，开始状态为函数的第一个基础块，结束状态为函数 ret 指令所在基础块。为了不改变控制流信息的异常，本文引入了 NFA 代价和 NFA 参数的约束。

定义 2(NFA_{fi}代价) 对应于 f_i 执行所占系统资源，用于标记 NFA_{fi}，记为 $g(NFA_{fi})$，即 $g:Q\rightarrow M$。其中 $Q=\{x|x=NFA_{f_i},i\geqslant 1\wedge i\in N\}$，$M=\{\vec{m}|m=(k_1,k_2,\cdots,k_i),i\geqslant 1\wedge i\in N\}$，$k_i$ 为不同的系统资源对应量。

ECCFM 模型是一个八元组$(Q,\Sigma,\Gamma,\Delta,\delta,\lambda,q_0,Z_0)$，其中 $Q=\{x|x=NFA_{f_i},i\geqslant 1\wedge i\in N\}$是状态集合；$\Sigma$为有限输入字母表，其元素为每个 NFA_{f_i} 中标记的 CALL，记为 target@addr，target 为跳转目的地址，addr 为 CALL 标记地址；Γ 为栈内符号集合，形式如 target@addr；Δ 为输出字母表{control_alarm，args_alarm，$g(NFA_{fi})$_alarm，…}；δ 是 $Q\times\Sigma\times\Gamma\rightarrow Q\times\Gamma^*$ 的状态转换函数；λ 是 $Q\times\Sigma\times\Gamma\rightarrow\Delta$ 的映射；q_0 为初始状态；Z_0 为初始栈底元素。

3.2 学习及检测阶段

第一阶段学习：通过分析源代码或二进制文件生成单个函数的 NFA 已经很成熟。本文使用 EEL 库[4]来生成各个 NFA_{fi}。同时对每个 NFA_{fi} 建立一个 Hash 表，保存跳转指令的源地址和目的地址。类似地，在静态分析阶段获得的函数调用关系保存在另外一个 Hash 表里。对于目的地址不能确定的跳转或者函数调用，仅标记为 NULL，在动态监控时考虑其一致性。图 1 是例子代码和相应的控制流图，图 2 说明了函数 fun1 对应的 NFA_{fun1}。

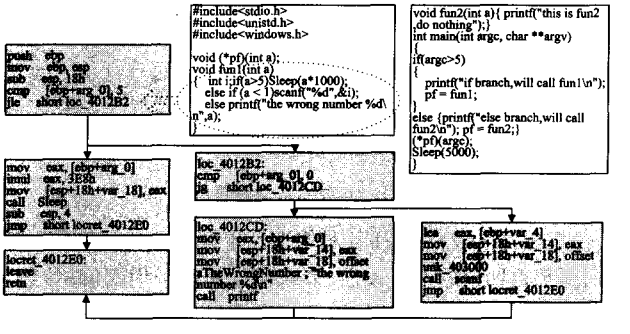

图 1 例子 fun1 的控制流图

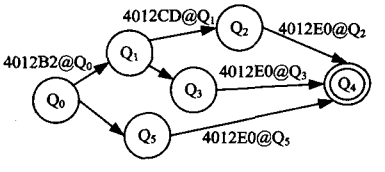

图 2 fun1 对应的 NFA

第二阶段学习：在形成各个函数的 NFA 以后，需要对函数 NFA 的整体进行约束。本文主要从 3 个方面进行约束：NFA 代价约束、出口点约束和参数约束。代价约束主要约束单函数的消耗资源，由于函数调用深度的关系，当前函数代价不应该包含被调函数的代价，算法如图 3 所示。

```
i=0;
WHILE(NFAfi)
{IF(NFAfi is base function)
    g(NFAfi)=Mfi;
ELSE g(NFAfi)=Mcurr- Σg(NFAfi) ;
//fj is the function called by function fi
i++;
}
```

图 3 $g(NFA_{fi})$的算法

在算法 3 中，由于系统内部函数和标准的库函数一般经过很好的开发和测试，因此本文不对其进行内部分析，称不含

有其他用户自定义函数调用的函数为 base function。

出口点约束主要是为了保证函数调用返回地址不被篡改，因而在函数调用发生时将返回地址压栈，在函数出口处验证其一致性。本文采取动态学习，即在程序运行时记录函数调用指令处的地址，从而避免了由函数指针导致的间接调用和虚函数等无法静态获取目的地址的缺点。

对于参数约束的学习，根据程序局部性原理，本文只考虑单 NFA 内部系统调用参数的关系约束，学习算法如图 4 所示。

```
RELATION=NFAfi
{EQUAL, INSAMEDIR, CONTAIN}
i=0
WHILE(NFAfi)
{
  For(eack x,y)xy. relation=0;
  WHILE(each args X)
  { WHILE(each args Y)
    { switch(Relation)
      {case EQUAL: xy. equcnt++;break;
          case INSAMEDIR: xy. dircnt++; break;
          case CONTAIN: xy. concnt++; break;
          default:xy. relation = NULL;
      }// switch end
    }// WHILE(each args Y)
  } WHILE(each args X)end
i++;
} //WHILE(NFAfi) end
```

图 4　参数关系学习算法

在检测阶段，有以下 4 种情况产生异常：控制流的不一致性；函数返回值的不一致性；节点代价超过平均阈值；参数关系不一致性。对于前两种情况产生的异常，由于是严格的行为不一致，将直接产生报警。对于后面的两种情况，本文引入累积因子 a。产生一次此种异常时，a 自增；当 a 超过一定的阈值后产生报警。

4　实验与分析

本实验在 Intel(R) Core™2 CPU 6320@1.86GHz，1.97GB RAM Linux 9.0 环境下进行，利用改编的 EEL 库[4]和程序动态分析工具 DYNINST[9]分别对 gzip，vsftpd 进行分析和监控。$g(NFA_{fi})$中的资源，如 IO 资源、网络吞吐率、文件读写操作、CPU 使用率、RAM 使用率等，为了简化实现，实验中仅选取 CPU 使用率。

4.1　稳定性分析

实验考虑 $g(NFA_{fi})$的收敛和参数学习算法的收敛速度。图 5 显示了 gzip，vsftpd 的各两个函数的 $g(NFA_{fi})$的收敛速度，从该图中可以看出，gzip，vsftpd 的 $g(NFA_{fi})$是比较稳定的。更进一步的统计结果得出约 79% vsftpd 的函数具有较好的稳定性。gzip 的稳定性更好，达到 84%。这表明在程序运行中各函数对资源的消耗具有很大的稳定性，程序处理问题的规模的增长只与少部分函数关系比较密切。图 6 显示了参数关系的稳定性，约 68%的参数之间的关系是稳定收敛于 RELATION 集合中的一种关系。对于参数关系波动比较大的，本文不把其作为检测的依据，即认为该参数之间不存在 RELATION。

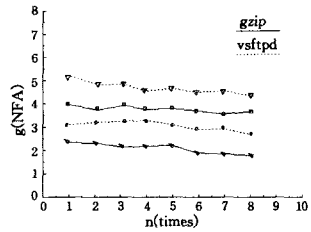

图 5　$g(NFA_{fi})$的稳定性

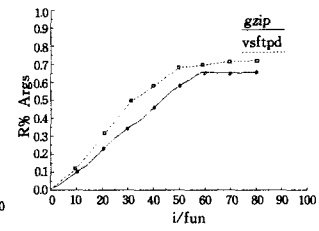

图 6　参数关系的稳定性

4.2　检测能力

在检测能力方面，对 vsftpd 的漏洞进行模拟典型攻击和 Mimicry 攻击[10]，并与 EMPDA[11]进行对比。图 7 表明，在开始阶段，ECCFM 的检测能力高于 EMPDA，这主要是因为该模型的静态分析阶段获取的是基本信息。随着学习的充分，两者的差别不是很明显，但是在漏报方面本文的模型具有明显的优势。在充分学习以后，单独进行 10 次 Mimicry 攻击，EMPDA 仅检测出 3 例，而本文提出的模型检测出 8 例。进一步分析表明，导致这种结果的原因是本文的模型充分利用程序的运行环境及参数之间的关系等相关信息，对 Mimicry 攻击具有更强的检测能力。我们无法使测试具有完备性，但是结果表明系统对 Mimicry 攻击检测的有效性。

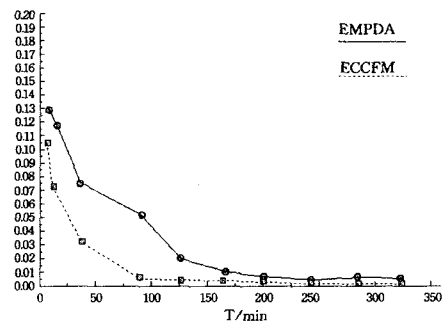

图 7　vsftpd 模型的误报

4.3　性能分析

表 1 表明，系统性能仅下降约 9 个百分点，优于 EMPDA 模型，这主要是由于本模型仅对局部程序进行分析。但是对程序运行时间的影响高于 EMPDA 模型，这主要是因为监控程序需要获取更多的信息。

表 1　系统性能和程序运行影响

	Gzip%	Vsftpd%	OS%
EMPDA	7.53	9.79	16.83
ECCFM	11.23	13.78	9.21

结束语　本文提出的 ECCFM 模型是自动机模型的改进，充分利用程序运行时环境相关信息，结合系统调用参数之间的相关性约束。用实验初步验证了模型的有效性和精确性。同时考虑到性能方面的影响，利用程序局部性原理对程序仅作局部性分析，降低了对性能的影响。

下一步工作是对模型做一些改进，利用强学习和弱监督的思想，降低其对性能方面的影响，并对模型本身的安全性做进一步的形式化验证。

参 考 文 献

[1] Forrest S, Hofmeyr S A, Somayaji A, et al. A sense of self for Unix processes[C]// SP'96: Proceedings of the 1996 IEEE Symposium on Security and Privacy. Washington, DC, USA, IEEE Computer Society, 1996

（下转第 142 页）

定义 9(非键属性部分正确性,α_{Q_S}) 即关系 S 非键属性部分占总属性部分的比率,表示为

$$\alpha_{Q_S}=\frac{1}{m}\sum_{j=k+1}^{m}(1-E_j') \tag{11}$$

关系 S 正确性显然由键属性和非键属性两部分正确性组成,即 $\alpha_S=\alpha_{K_S}+\alpha_{Q_S}$。

对于完整性(定义略),亦有键属性 A_j 的完整性 $\beta_j=1-\chi_S$,键属性部分有 $\beta_{K_S}=\frac{k}{m}(1-\chi_S)$,关系完整性有 $\beta_S=\beta_{K_S}+\beta_{Q_S}$。

由式(8)、式(10)和式(11)可得

$$\alpha_S=\alpha_{K_S}+\alpha_{Q_S}=\frac{1}{m}(1-\gamma_S)(k+\sum_{j=k+1}^{m}(1-E_j-N_j)) \tag{12}$$

$$\beta_S=\beta_{K_S}+\beta_{Q_S}=\frac{1}{m}(1-\chi_S)(k+\sum_{j=k+1}^{m}(1-N_j)) \tag{13}$$

由式(12)、式(13)可以看出,关系正确性(完整性)评价计算与误属性(缺失性)和错误(空值)的分布相关,这样就可以利用统计抽样方法推断获得(空值率可以自动计算),从而有利于评价的实施。

结束语 本文是在文献[14-16]研究基础上的进一步深入和总结。属性粒度数据质量评价模型从正确性映射和完整性映射两个方面分别分析和描述了元组的质量类型。正确性指标定义包含数据项错误和元组误属两个方面的影响,完整性指标定义包含数据项空值和元组缺失带来的影响。通过分析空值即不正确也不完整,在模型中建立了正确性和完整性指标相互联系,进而引入属性量化前后错误(空值)率,进一步量化定义评价指标。

笔者还将以数据质量评价模型为基础继续就关系代数运算对正确性和完整性评价指标的传递影响的理论方面进行更为深入的研究。

参 考 文 献

[1] Aebi D, Perrochon L. Towards Improving Data Quality[C]// Proceedings of the International Conference on Information Systems and Management of Data. 1993:273-281

[2] Kon H B. A process view of data quality (TDQM working paper)[M]. Total Data Quality Management Research Program. Sloan School of Management, Massachusetts Institute of Technology, 1993:17

[3] Kon H B, Madnick S E, Siegel M D. Good Answers from Bad Data: A Data Management Strategy[R]. Massachusetts Institute of Technology (MIT), Sloan School of Management, 1995:1-16

[4] Motro A, Rakov I. Estimating the Quality of Data in Relational Databases[C]// 1996 Conference on Information Quality. Cambridge, Massachusetts, 1996:94-106

[5] Motro A, Rakov I. Estimating The Quality of Databases[C]// The 3rd International Conference on Flexible Query Answering Systems (FQAS). Cambridge, MA, 1998:298-307

[6] Motro A, Rakov I. Not All Answers Are Equally Good: Estimating the Quality of Database Answers[M]. Kluwer Academic Publishers, 1997:1-21

[7] Reddy M P, Wang R Y. A Data Quality Algebra for Estimating Query Result Quality[C]//CISMOD, Conference. Bombay: 1996

[8] Wang R Y, Ziad M, Lee Y W. Data Quality [M]. New York: Kluwer Academic Publishers, 2002

[9] Parssian A, Sarkar S, Jacob V S. Assessing Data Quality for Information Products: Impact of Selection, Projection, and Cartesian Product[J]. Management Science, 2004, 50(7):967-982

[10] Parssian A, Sarkar S, Jacob V S. Assessing Information Quality for the Composite Relational Operation Join[C]// The 7th International Conference on Information Quality. MIT Cambridge, MA USA, 2002:225-237

[11] Parssian A, Sarkar S, Jacob V S. Assessing data quality for information products. [C]// ICIS-99. 1999:428-433

[12] Scannapieco M, Batini C. Completeness in the Relational Model: a Comprehensive Framework[C]// 9th International Conference on Information Quality. Cambridge, MA, USA, 2004:333-345

[13] Ballou D P, Chengalur-Smith I N, Wang R Y. Sample-based Quality Estimation of Query Results in Relational Database Environments[J]. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(5):639-650

[14] 陈卫东,张维明. 数据质量模型及选择运算中的质量传播研究[J]. 计算机工程与应用,2007,43(27):1-3

[15] 陈卫东,张维明. 投影运算对数据库数据质量的传递影响研究[J]. 计算机应用研究,2008,25(9):2751-2753

[16] 陈卫东,张维明. 笛卡尔积运算对数据库数据质量的传递影响[J]. 计算机科学,2008,35(6):210-212

(上接第 114 页)

[2] Gao D, Reiter M K, Song D. Behavioral distance measurement using hidden markov models[C]// Zamboniand D, Kruegel C, eds. Research Advances in Intrusion Detection, LNCS 4219. Berlin Heidelberg: Springer-Verlag, 2006:19-40

[3] Ghosh A, Schwartzbard A. A study in using neural networks for anomaly and misuse detection[C]// Proceedings of the 8th USENIX Security Symposium. 1999

[4] Giffin J T, Jha S, Miller B P. Efficient context-sensitive intrusion detection[C]// Proceedings of Symposium on Network and Distributed System Security. 2004

[5] Wagner D, Dean D. Intrusion detection via static analysis[C]// Proceedings of the 2001 IEEE Symposium on Security and Privacy. May 2001:156-168

[6] Gao D, Reiter M K, Song D. Gray-box extraction of execution graphs for anomaly detection[C]// Proceedings of the 11th ACM Conference on Computer & Communication Security (CCS 2003). 2003

[7] Gao D, Reiter M K, Song D. On gray-box program tracking for anomaly detection[C]// Proceedings of the 13th USENIX Security Symposium. 2004

[8] Sharif M S, Singh K, Giffin J, et al. Understanding precision in host based intrusion detection[C]// Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID). 2007:21-41

[9] Buck B, Hollingsworth J K. An API for Runtime Code Patching [J]. The International Journal of High Performance Computing Applications, 2000(14):317-329

[10] Parampalli C, Sekar R, Johnson R. A practical mimicry attack against powerful system-call monitors[C]// Proceedings of the 2008 ACM Symposium on information, Computer and Communications Security. March 2008:156-167

[11] Lu Wei, Zeng Qingkai. A control flow based program behavior extended model[J]. Journal of software, 2007, 18(11):2841-2850