

一种基于描述语言的软件可信发布方法研究

李剑飞 徐开勇 金 雷

(信息工程大学密码工程学院 郑州 450004)

摘 要 目前对于高安全要求的内网环境,如何利用可信计算技术确保软件发布的安全与可靠具有重要的研究意义。由于软件可信发布具有动态性,即不同用户和平台对软件的要求不尽相同,而人为配置发布不仅效率低下而且无法确保其安全可靠,因此提出了一种用户、平台、软件之间相互选择的智能发布策略,其中的软件功能可信度算法可定量计算出软件功能符合用户需求的程度,智能匹配算法根据依赖关系生成安装序列以确保安装运行的顺利进行。同时为了描述发布过程中的软件信息及满足发布算法的需求,设计了一种基于 XML 的软件描述语言 SDDL(Software Distribution Description Language)。通过分析及实例证明了此发布方式确实能增强软件发布的可信性。

关键词 软件描述,软件可信发布,智能发布,可信计算

中图分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.048

Trustworthy Software Distributing Mode Based on Software Description

LI Jian-fei XU Kai-yong JIN Lei

(Institute of Security Engineering, PLA Information Engineering University, Zhengzhou 450004, China)

Abstract The present study of software distribution is mainly about the efficiency and integrity. For high security requirement of inner network environment, how to make use of trusted computing technology to ensure the safety and reliability of the software release has more important research significance. Because the software trust distribution is dynamic, namely the requirements of different users and platforms are different, and human configuration is inefficient and its safety and reliability can't be ensured, thus an intelligent release strategy among user, platform, software was put forward. The software function reliability algorithm can quantitatively calculate the degree of the software functions conforming with user requirements, and intelligent matching algorithm can generate installation sequence based on dependency to ensure running reliably. In order to describe the software information in the distribution process and meet the needs of distribution algorithm at the same time, we designed an SDDL (Software Distribution Description Language) based on XML. Analysis and application example demonstrate that the mode can indeed enhance the trust of software distribution.

Keywords Software description, Software trust distribution, Intelligent distribution, Trusted computing

软件发布研究已经进行了很多年,成果颇多^[1-4],但是大部分的研究旨在提高软件发布的效率与智能性,对发布安全的研究大多也仅限于软件传输过程及安装过程^[5],而忽视了作为发布过程重要参与者的用户的影响,同时用户作为软件最终的使用者,对软件可信性的影响持续于整个软件的应用周期。现有的访问控制技术虽然起到了一定的防护效果,但其技术特点对发布的效率影响较大,同时只是单纯对用户进行管控,并没有考虑发布的软件和应用平台的情况,可能会使合法用户无意识的合法操作导致软件或者平台被攻击或破坏^[6-9]。因此,一个综合用户、软件、平台三者的发布方式是很有研究价值的。

1 发布方式概述

本发布方式研究的区间是软件设计研发完成存入软件发布服务器后,从软件使用者发起请求开始,到软件使用者在平

台正常使用结束。其中参与的主体为使用者、软件发布服务器、可信计算平台,客体为软件,主体对客体实施的行为即为研究的重点,而根据实施行为的不同,在本方式中又具体分为软件自身行为、使用者行为、服务器发布行为、平台认证行为。这些主要因素的关系即发布流程如图 1 所示。

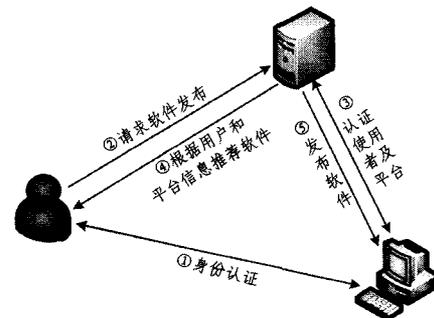


图 1 发布流程

到稿日期:2014-12-08 返修日期:2015-03-07 本文受国家自然科学基金项目:密码片上系统安全模型结构与验证方法研究(61072047)资助。
李剑飞(1991-),男,硕士生,主要研究方向为信息安全,E-mail:276346387@qq.com;徐开勇(1963-),男,博士,研究员,主要研究方向为信息安全、通信安全;金雷(1989-),男,硕士生,主要研究方向为信息安全。

①身份认证:指使用者与可信平台之间的认证,推荐使用 UKEY 认证的方式,因为 UKEY 可以保存用户的资料,包括基本资料以及权限资料,在登录平台时需要先认证使用者是否有登录本平台的权限,然后再通过口令等方式验证是否为本平台持有 UKEY,双向验证后使用本平台。这是对使用者第一步行为的控制,作用在于制止非法使用者接入网络,同时防止合法用户越权操作接入网络。

②请求软件发布:即使用者第二步行为,根据使用者需求和自身身份请求相对应的软件发布。首先也需要登录发布服务器,由于应用环境中肯定不止发布一个应用系统,为了方便使用者登录,可以另外建立一个认证服务器,采取统一登录的方式,此处不再赘述。

③认证使用者及平台:上一步和这一步即为一个完整的挑战应答循环。然而不同于普通的应答,由于可信计算模块有加密存储和报告的功能,因此在第一次用户和平台单独与发布服务器验证后,以后的验证均可由模块计算用户和平台的摘要值并进行储存,然后在服务器需要时报告。这样的好处是可以延长挑战应答间隔,减少系统开销,同时存储记录无法改变,可以做到有证可查。

④根据用户和平台信息推荐软件:用户登录发布系统后选取与自身身份相符合的软件集合,同时发布系统会提取用户和平台的信息,包括静态的数据信息与动态的行为信息,再根据发布算法对软件符合用户及平台预期的程度进行计算,然后根据环境所需的标准进行推荐,具体发布策略见下文。

⑤发布软件:用户根据功能需求选择软件进行下载,对于多个软件相互制约的情况,按照策略给出的安装序列进行安装即可。与普通软件发布不同,由于发布服务器与用户终端均为可信计算平台,因此可以采取可信计算平台的安全度量与汇报机制对软件的完整性进行验证,以确保软件传递过程的完整性与安装时的可靠性。

2 基于 XML 的软件发布语言设计

上述发布方式中需要对软件信息、用户需求、平台约束做统一的描述,进而才能进行对比分析,于是需要一种能够支持软件发布整个过程并能够描述软件可信要素的描述语言。在分析现有软件描述语言的基础上,提出一种基于 XML 的软件发布语言 SDDL (Software Distribution Description Language)^[10-12]。

2.1 现有软件描述语言分析

在现有的软件描述语言中,比如 Richard S. Hall 等人提出 DSD (Deployable Software Description)^[13,14] 语言用来描述软件系统及其复杂的内外部依赖关系; Arthur van Hoff 等人提出的 OSD (Open Software Description)^[15,16] 语言可以为打包软件描述软件组件、版本和内部结构等关系。信息管理格式 (Management Information Format, MIF)^[17,18] 是由桌面管理组织 (Desktop Management Task Force, DMTF) 创立的,它使用模型描述语言来描述不同运算系统的元素。总结其特点,如表 1 所列。

表 1 传统软件描述语言对比

软件描述语言	可扩展性	复杂度	对发布生命周期的支持
可部署软件描述 (DSD)	完全可扩展	较复杂	完全支持
开放式软件描述 (OSD)	部分可扩展	简单	部分支持
信息管理格式 (MIF)	部分可扩展	较复杂	完全支持

这些语言都可以用来描述软件,但是它们的着重点在于

软件发布的效率与智能性。对于可信软件而言,着重点应该在于描述软件的可信性特征,这里应包括软件的静态信息和软件运行时的行为信息。

2.2 SDDL 设计

对于一个可信软件而言,可以用 SDDL 语法来描述以下 5 个方面:

(1) 软件基本信息 (Software Info)

①传统信息:软件名称 (Name), 软件版本号 (Version), 软件描述 (Description), 制造商 (Vendor) 等。

②可信信息:制造商平台证书序列号 (VendorAIK), 大小 (Size), 数字签名 (Signature), 原始安装文件名 (Original file), 主要功能描述 (Functions) (其描述为针对用户的软件能提供的功能,如文档编辑、音频播放等)。

(2) 软件行为信息 (Software Behavior Info)

①软件行为的主体 (Subject)。其描述格式为:程序中映像名称 (一般等同于软件执行程序名) (Execution name), 软件版本号 (Version), 摘要值 (Sha1)。

②软件行为的客体 (Object)。软件行为的客体包括硬件与软件两类,而对硬件的操作通常可以用软件形式表现,如对内存的操作可以看作其中进程的产生与销毁,对磁盘的操作可以看作对其内容的读、写、存等,因此选取以下主要资源进行研究,分别是内存、注册表、文件系统以及网络资源的访问,其描述格式如表 2 所列。

表 2 系统资源的描述方式

对象	所属硬件	描述内容	描述方式
进程	内存	目录、文件名、类型(扩展名)	路径
注册表	硬盘	目录、项目名、类型	路径
文件	硬盘外置存储	目录、文件名、类型(扩展名)	路径
网络	网卡	IPv4/v6 地址、端口、协议	路径

③由于软件行为的目的是对资源的访问,复杂的业务逻辑的目的都可以归结为数据的输入与输出。结合软件行为可信要素以及可信行为描述的必要性与其可观测性,可将各类资源的访问动作分类如下。

进程:创建 (Create), 退出 (Exit), 加载映像 (Load Image);

文件:打开 (Open), 关闭 (Close), 读 (Read), 写 (Write), 其他 (Other);

注册表:打开 (Open), 关闭 (Close), 读 (Read), 写 (Write), 其他 (Other);

网络:连接 (Connect), 断开 (Disconnect), 发送 (Send), 接收 (Receive)。

基于软件对操作系统的调用,可将动作细分类,如表 3 所列。

表 3 基于系统调用的动作的分类

对象	动作描述
进程	Load Image/Thread Create/Thread Exit/Thread Profile/Process Create/Process Defined/Process Exit
文件	Create File/Read File/Write File/Lock File/Query Directory/Query Info...Set Info.../...
注册表	RegCloseKey/RegCreateKey/RegDeleteKey/RegDeleteValue/RegEnumKey/RegEnumValue/RegFlushKey/RegLoadKey/RegOpenKey/RegQueryKey/RegQueryKeySecurity/RegQueryMultipleValueKey/RegQueryValue/RegRenameKey/RegSetInfoKey/RegSetKeySecurity/RegSetValue/RegUnloadKey
网络	UDP Send/UDP Receive/TCP Send/TCP Connect/TCP Disconnect/TCPReceive/TCP Retransmit

(3) 平台硬件约束 (Platform Hardware Constraint)

〈Constraint〉描述软件部署在平台的硬件及基础软件约束条件。要确保软件在用户平台可以安全可靠运行,需要满足一定的条件约束,比如用户平台 CPU 主频〈Cpu〉、内存大小〈Memory〉、硬盘空间大小〈Harddisk〉、网络带宽〈Network〉、操作系统(OS)等。这些都可以用量化的方式在描述语言中定义,并在程序解析这个 XML 文件时提取出来与可信平台提供的硬件信息比较,其描述格式为:一个简单描述〈Description〉,一个可选的条件判别式〈Discriminant〉,一个约束条件〈Condition〉。

(4) 软件依赖约束 (Software Dependency Constraint)

〈Dependency〉描述软件的发布安装所要预安装的依赖软件约束信息,包括依赖子系统名称〈SoftwareName〉、可满足依赖的软件的描述文件〈Description_By〉以及可选的元素依赖类型〈Dependency_Type〉,其中依赖类型可以是以下值:“Pre_Requisite”表示必须在发布安装之前安装,“Requisite”表示必须在发布可用之前安装。

(5) 软件发布行为 (Software Deploy Behavior)

描述和定义软件的发布过程中操作系统需要做的行为〈Action〉。比如安装软件时需要预先结束掉一些相关程序的进程,或者安装软件后需要重新启动操作系统等。这些都可以通过在描述文件中定义来让程序去执行和实现,其描述形式为:一个动作名称〈Act_Name〉,一个简单描述〈Act_Description〉,一个动作内容〈Act〉,一个判别时间〈When〉。

3 软件可信发布方法

根据可信软件的定义^[19],即:如果软件的行为总能够与人们的预期一致,那么认为该软件是可以信任的。而在发布过程中的参与者包括软件开发者、软件使用者、软件发布管理员。开发者对软件行为的预期应为软件的功能满足设计需求,软件能够正常运行并没有安全威胁;软件使用者对软件行为的预期应该为软件能够满足个人应用需求,不存在隐藏的漏洞或者后门风险;作为软件发布管理员对软件行为的预期应该为他们能够根据平台与用户的需求及约束合理发布软件,使得软件能够正常安装运行。因此需要对三者的预期进行描述,而描述方式即为上述 SDDL 语言,只不过关注重点不同,开发者需要关注所有项目,用户需要关注基本信息中的软件功能描述,发布管理员需要关注发布约束及依赖。然而在最终发布时我们需要衡量所有预期并做出决策,人工计算不仅效率低下而且容易因为疏忽导致隐患,因此依据模糊计算的理论设计了一种可以定量计算软件功能可信度并能够自动匹配软件依赖及约束的软件可信发布方法,其流程如图 2 所示。

本发布方法主要分为两部分:第一部分的主要目的是根据管理员制定的基于用户身份的功能需求限制,来限定用户选择软件不会超出自身权限,同时确保不会安装多余功能而产生安全风险。具体的解决方法是设计了一种基于模糊计算理论的软件功能可信度定量计算算法,用户根据可信值选取所需功能软件即可。第二部分的目的是解决因软件安装时存在依赖与约束而人工配置效率低下且可靠性低的问题。对此,设计了一种基于运筹学理论的智能匹配算法,生成最优软件安装序列推荐给用户安装。

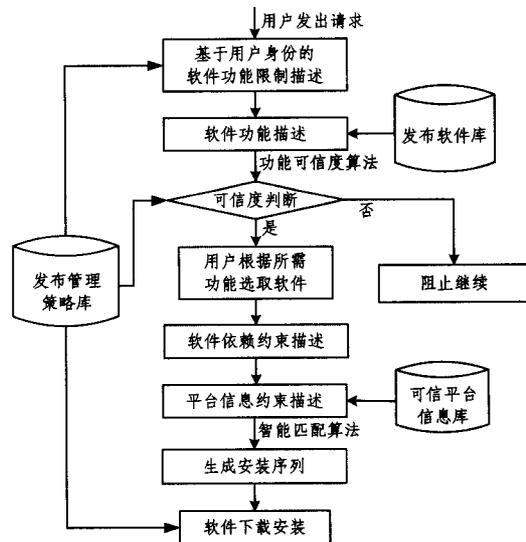


图 2 可信发布方法流程

3.1 软件功能可信度算法

由于软件开发者在提交软件时已经按照要求将软件的功能〈Funtions〉以声明的形式与软件一同交付于发布系统,因此发布系统中对于任意软件 S,其对应的功能声明集合:

$$F = [f_1, f_2, f_3, \dots, f_N]$$

定义 1 发布过程中在软件功能匹配时,存在的功能 f_N 数值置为 1,不存在的则保持初始状态为 0,即:

$$F = [f_1, f_2, f_3, \dots, f_N] (f_N = 1 \text{ or } f_N = 0)$$

定义 2 发布管理员针对不同身份用户设定所需功能限制,以策略形式存放于发布管理策略库中,用集合 RFB 表示,其表示规则为允许的功能将对对应 rfb_i 置为 1,不允许的功能置为 0,即:

$$RFB = [rfb_1, rfb_2, rfb_3, \dots, rfb_i] (rfb_i = 1 \text{ or } rfb_i = 0)$$

判断算法的目的即判断 RFB 与 F 的相似程度,主要步骤如下:

步骤 1 通过 SDDL 描述的 F 与 RFB 格式相同,只需提取〈Funtions〉中的字段进行匹配即可。根据模糊计算的原理,有:

$$Q_F = [fq_1, fq_2, fq_3, \dots, fq_i] \sum_{n=1}^i fq_i = 1$$

与 F 规则相同,当 f_{b_i} 存在时, f_{q_i} 被置为 1,其余仍保持 0 的状态,当匹配结束后统计共有多少个 1 在矩阵 Q_F 中,假设有 M 个 1,则根据条件 $\sum_{n=1}^i fq_i = 1$,将所有为 1 的替换为 $1/M$ 。

步骤 2 根据模糊计算理论中有关相似度的计算法则,求 RFB 与 F 的“和”,得到的是介于 0 与 1 的数,代表两者的相似度 D 。

$$D = RFB \cdot Q_F = [rfb_1, rfb_2, rfb_3, \dots, rfb_i] \cdot \begin{bmatrix} fq_1 \\ fq_2 \\ fq_3 \\ \dots \\ fq_i \end{bmatrix}$$

$$= \sum_{n=1}^i (rfb_i \times fq_i)$$

步骤 3 根据求解出的数字,服务器可以判定此软件功能是否满足用户功能需求权限,如果结果为 1,则说明此软件

功能等于或小于用户所具有的最大权限:如果小于 1,则说明存在超出给定功能权限的部分,将不予以推荐。

3.2 智能匹配算法

经过功能可信度计算后的软件以目录形式展现给用户供其选择,用户只需根据所需功能选择软件即可,之后软件发布系统利用智能匹配算法分析软件依赖及冲突,然后生成软件安装序列,客户端程序对软件进行自动下载安装。

定义 3(发布依赖) 一个软件发布依赖用一个二元组 (d_1, d_2) 表示,其中 d_1 是依赖 d_2 的软件的名称, d_2 是被依赖软件的名称,并且 $d_1 \neq d_2$ 。

定义 4(发布动作) 一次软件发布动作 P 用一个三元组 (D, M, L) 表示,其中 D 是发布软件的集合, M 是应用终端上已存在的软件集合, L 是发布依赖集合,其中 L 是反自反的, $\{d_1 | (d_1, d_2) \in L\} \subseteq D$ 。

匹配算法的目的即将 D 中软件除去 M 中已有软件后,根据 L 中的依赖关系,输出软件发布安装的序列。在实际过程中由于大部分系统软件安装时只能进行单一实例运行,因此我们在分析问题时也遵循这一规则。

步骤 1 用户选取软件后,通过匹配由 SDDL 描述的 \langle Dependency \rangle 信息,将所选软件和其依赖的软件一起提取,生成用户软件集合:

$$U = \{u_1, u_2, u_3, \dots, u_N\}$$

根据应用平台提交的信息,提取已存在的软件集合:

$$M = \{m_1, m_2, m_3, \dots, m_N\}$$

通过匹配 u_N, m_N 由 SDDL 描述的 \langle Dependency \rangle 与 \langle Constraint \rangle 信息去除 U 中包含的 M ,使得 $U \cap M = \emptyset$,生成最终待发布软件集合 D ,然后提取 D 中软件,由 SDDL 描述的软件依赖关系生成集合 L 。

步骤 2 根据 L 生成软件依赖关系树,此关系树为有向树,其根节点即为 D 中软件,其父节点为需要解决依赖的软件 d_1 ,其子节点为被依赖的软件 d_2 。然而由于具体发布时可能存在某子节点的安装依赖相邻父节点的问题,此时即成了有向图,需要将其转化为有向树,因此规定在生成依赖关系树时,需要将每个节点的依赖关系全部在此节点的子节点中表示,然后逐一生成即可。最终生成的依赖关系树应该满足如下要求:

- (1)在此有向树中有且仅有一个入度为 0 的节点;
- (2)除树根外的节点入度为 1;
- (3)从树根到任一节点有一条有向通路,即依赖关系 L ;
- (4)树中的有向通路都是单向的,不存在环路,即不存在软件循环依赖问题。

步骤 3 在生成依赖树后,应采取适当的方法对此树图进行遍历,然后生成相应的序列,在此根据本文生成序列所需特点,对树的遍历进行了对比研究,其结果如表 4 所列。

表 4 树的遍历算法对比

算法名称	生成序列源头	生成序列特点	可扩展性	实现难度
前序遍历	根节点	先根前序	弱	低
后序遍历	子节点	后根后序	弱	低
深度搜索	任意点	继承性	强	适中
广度搜索	任意点	层次性	强	适中

经对比可知,由于安装软件时的实际情况是子节点的软件安装成功后再安装父节点,即考虑的是安装层次的关系,因

此采取树的广度优先遍历算法,由根节点开始按层次遍历后生成遍历序列 S 。

步骤 4 生成遍历序列后,还需对序列做如下处理:

(1)由于采取从根节点开始的广度优先遍历,因此 S 是由软件 d_1 即用户选择的待发布软件开始的,然而实际安装时应从被依赖软件开始安装,于是需要将 S 做倒序处理,生成 S^{-} 。

(2)在生成依赖树过程中考虑到各层次依赖问题时,我们的解决方法是将重复的依赖关系也表示出来,则生成序列时会存在重复节点情况,此时只需将 S^{-} 中存在的重复节点只保留顺序最靠前的即可,代表安装时在最低层安装此软件。

按照上述步骤生成的 S^{-} 即为用户所需的最终安装序列。

4 安全性及实用性分析

在安全方面,本发布模式不仅应用了现有的可信计算平台作为硬件保障,采用软件行为声明的方式,从软件开发者提交软件信息开始即具有不可否认性;并引入可信计算的可信链传递的思想^[6],将 SDDL 描述的软件信息作为可信链传递的关键载体,通过从请求开始逐步验证其可信性,直到发布为止,这使得可信计算要求的“有据可查”在软件发布过程中得以实现。其与传统发布方式安全性的对比如表 5 所列。

表 5 安全性对比

对比项目	可信软件发布方式	传统软件发布方式
完整性认证	依托可信计算平台	普通协议认证
静态度量	依托 TPM 完成	普通 Hash 运算
动态度量	依托 TPM 完成	无
权限声明	依托行为声明	缺乏可信性
行为分析响应	依托行为声明	无
推荐策略	基于可信行为策略	基于需求推荐
应用场景	高可信要求环境	低可信要求环境

在实用性方面,由于现有的可信计算平台及相关技术已经十分健全,所需要的就是指导应用的理论,此发布方式用到的创新理论为软件行为声明,而在这方面的相关研究也有一定成果^[9-12]。在这两方面基础之上,本发布方式可以在各种系统上快速搭建,同时由于每个步骤都做到了“有据可查”,避免了因操作不当出现问题后的责任划分不清,因此减少了开发者的维护负担,保障了使用者应用软件的方便安全。

5 应用实例说明

由于前文提到安全和效率一直是对立地存在,而本文的发布方式在设置判定阈值时更着重于考虑可信性,因此本发布方式更适合应用于对安全性有更高需求的党、政、军机关或者对安全要求高的企业内网环境。在此以某企业会计部门为例,为了保障企业财务安全,其将部署新的可信计算平台,重建整个会计应用系统。对其操作流程进行说明如下:

(1)先将可信计算平台合理安装及部署,内部网络连通,平台及人员信息在认证服务器端进行认证及保存,形成系统安装的硬件基础。

(2)系统部署人员使用 UKEY 或者生物特征等方式登录可信计算平台,平台通过验证登录口令和身份后确定其权限,方可让其继续操作。

(3)通过口令直接或者登录统一验证服务器登录的方式进入发布服务器,选择所需要的软件。假设功能安全策略库规定其所需功能有:文字编辑、会计应用、多媒体播放、图片编

辑,不能使用的功能有:编程操作、密码管理。根据软件功能可信度算法有:

①若任一软件 S_1 具备功能 $F_1=[\text{文字编辑,会计应用,多媒体播放}]$,则 $Q_{F_1}=[1/3,1/3,1/3]$ 。

由于 $RFB=[\text{文字编辑,会计应用,多媒体播放,图片编辑,编程操作,密码管理}]=[1,1,1,1,0,0]$,得:

$$D=RFB \cdot Q_{F_1}=[1,1,1,1,0,0] \cdot \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \\ 0 \\ 0 \\ 0 \end{bmatrix} = 1$$

即 S_1 这个软件功能没有超出会计的权限,可以进行推荐发布,供用户选择。

②若任一软件 S_1 具备功能 $F_1=[\text{文字编辑,会计应用,多媒体播放,编程操作}]$,则 $Q_{F_2}=[1/4,1/4,1/4,1/4]$ 。

表6 软件下载目录格式

软件名称 <Name>	软件版本号 <Version>	软件描述 <Description>	制造商 <Vendor>	大小 <Size>	功能描述 <Functions>	软件标识存储位置 <identification-address>	软件存储位置 <software-address>
----------------	--------------------	-----------------------	-----------------	--------------	---------------------	--------------------------------------	------------------------------

图3表示用户选取了所需软件 u_1, u_2 , 根据其 SDDL 描述的 <Dependency> 信息,发现依赖软件 u_3, u_4, u_5, u_6 ,同时提取终端已安装软件信息进行比较,发现 u_6 与 m_3 相同,即已存在于终端之上,亦即排除在待发布软件集合 D 外,则 $D=[d_1, d_2, d_3, d_4, d_5]=[u_1, u_2, u_3, u_4, u_5]$ 。在 D 生成的同时需要考虑 <Constraint> 中的相关平台约束信息,平台信息会利用可信计算模块的完整性汇报功能进行提供,将集合 D 中的软件按照平台硬件要求进行筛选,生成最终的待发布集合 D 。

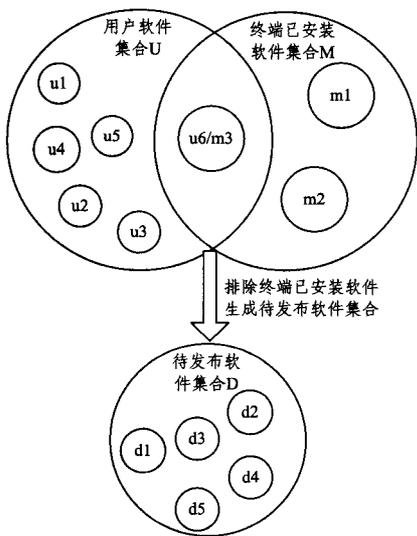


图3 待发布软件生成示意图

假设 D 中各软件的依赖关系 L 如图4(a)所示,则最终生成的依赖树如图4(b)所示。

根据依赖树,采取广度优先遍历算法输出遍历序列 $S=d_1, d_2, d_3, d_4, d_3, d_5, d_5, d_3, d_5$, 则 $S^- = d_5, d_3, d_5, d_5, d_3, d_4, d_3, d_2, d_1$ 。

S^- 中保留最前端,顺序剔除重复元素后得到最终发布安装序列: $S^- = d_5, d_3, d_4, d_2, d_1$ 。

终端采取自动安装或人工安装的方式,按照 S^- 对软件进行安装即完成最终发布。

由于 $RFB=[\text{文字编辑,会计应用,多媒体播放,图片编辑,编程操作,密码管理}]=[1,1,1,1,0,0]$,得:

$$D=RFB \cdot Q_{F_2}=[1,1,1,1,0,0] \cdot \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 0 \\ 1/4 \\ 0 \end{bmatrix} = 3/4$$

即 S_2 这个软件功能不满足会计身份的权限要求,则不予以推荐进入下一步发布。

(4)通过软件功能可信度计算后的软件将以列表形式呈现给用户。本发布系统在用户登录服务端根据所需功能选择软件时,服务器提供一个软件标识中相关内容组成的目录,其本质为服务器中存储标识的数据库的表,用户只需根据功能 <Functions> 需要选择相应软件即可,具体格式如表6所列,选择完成后采用智能匹配算法进行运算。

整个应用系统按照此方式即可安全可信地完成全部平台的软件部署。

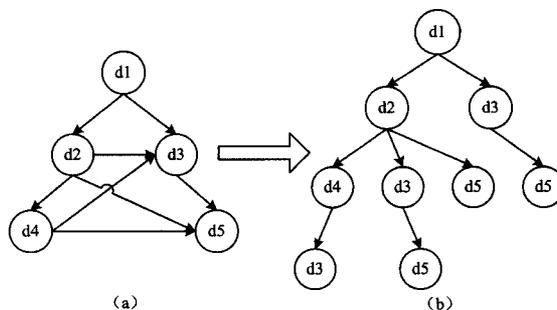


图4 依赖树生成

结束语 本文从软件发布现状入手,分析了现有架构和技术的优缺点,以软件的描述信息作为可信传递的载体,给出了一种综合了用户、发布服务器、可信计算平台3方面的可信发布方式,设计了用于判断软件功能可信性的算法,以及用于生成软件发布安装序列的算法,最终结合具体应用案例说明系统的研究价值和在可信性上相对于传统系统的提升。在以后的工作中需要改进的方面有以下两点:(1)开发者行为验证方面还需要进一步研究;(2)对于用户及平台信息提取的实现还可以进一步细化。目前的发布方式还只是模型阶段,具体的实施还有待进一步研究,未来成熟的系统将会给高安全需求的用户带来更完善的软件发布可信性保障。

参考文献

[1] 赵雨水,左青,杨立,等.软件发布机制体系结构研究[J].计算机工程与设计,2010,31(4):700-705
Zhao Yu-shui, Zuo Chun, Yang Li, et al. Architecture for software distribution[J]. Computer Engineering and Design, 2010, 31(4): 700-705

[2] 刘甜.软件发布机制的研究与应用[D].石家庄:石家庄铁道大学,2014

(下转第262页)

- [6] Hosek P, Migliavacca M, Papagiannis I, et al. SafeWeb: A middleware for securing Ruby-based Web applications[C]// Proceedings of the 12th International Middleware Conference. International Federation for Information Processing, 2011:480-499
- [7] Migliavacca M, Papagiannis I, Eysers D M, et al. DEFCON: High-Performance Event Processing with Information Security[C]// USENIX Annual Technical Conference. Boston, MA, 2010: 88-102
- [8] Enck W, Gilbert P, Chun B G, et al. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones[C]// OSDI. Berkeley, CA, USA; USENIX Association, 2010:255-270
- [9] Rodero-Merino L, Vaquero L M, Caron E, et al. Building safe PaaS clouds: A survey on security in multitenant software platforms[J]. Computers & Security, 2012, 31(1):96-108
- [10] Pappas V, Kemerlis V P, Zavou A, et al. CloudFence: Data Flow Tracking as a Cloud Service[M]// Research in Attacks, Intrusions, and Defenses. Springer Berlin Heidelberg, 2013:411-431
- [11] 刘鹏. 云计算[M]. 北京:电子工业出版社, 2011
Liu Peng. Cloud Computing [M]. Beijing: Publishing house of electronic industry, 2011
- [12] Bello L, Russo A. Towards a taint mode for cloud computing Web applications[C]//7th Workshop on Programming Languages and Analysis for Security. New York, ACM, 2012, 7: 1-7, 12
- [13] McDonald S. [EB/OL]. (2012-11-18)[2014-01-20]. <http://bitbucket.org/stephenmcd/cartridge/>
- [14] Johnson N. [EB/OL]. (2010-03-12)[2014-03-25]. <http://googleappengine.blogspot.com/2010/03/app-engine-community-update.html>

(上接第 228 页)

- Liu Tian. Research and Application of Software Distribution Mechanism[D]. Shijiazhuang: Shijiazhuang Tiedao University, 2014
- [3] 王华, 刘焕敏, 冯朝阳, 等. 一种分布式软件发布部署系统[J]. 计算机系统应用, 2012, 21(2):1-4
Wang Hua, Liu Huan-min, Feng Zhao-yang, et al. Distributed Software Release and Disposition System[J]. Computer Systems & Applications, 2012, 21(2):1-4
- [4] 陈伟, 魏峻, 黄涛. W4H: 一个面向软件部署的技术分析框架[J]. 软件学报, 2012, 23(7):1669-1687
Chen Wei, Wei Jun, Huang Tao. W4H: An Analytical Framework for Software Deployment Technologies [J]. Journal of Software, 2012, 23(7):1669-1687
- [5] Malek S, Medvidovic N, Mikic-Rakic M. An extensible framework for improving a distributed software system's deployment architecture[J]. IEEE Trans. on Software Engineering, 2012, 38(1):73-100
- [6] 张娴. 基于可信计算的软件安全下载设计与实现[D]. 西安:西安电子科技大学, 2010
Zhang Xian. Design and Implement of Software Secure Downloading Based on Trusted Computing[D]. Xi'an: Xidian University, 2010
- [7] Virvilis N, Gritzalis D. Trusted Computing vs Advanced Persistent Threats: Can a defender win this game? [C]//2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing, and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC). 2013:396-403
- [8] 蒋泽. 可信网络中用户行为可信评估的研究[D]. 重庆:重庆大学, 2011
Jiang Ze. Study on Evaluation of User Behavior Trust in Trusted Network[D]. Chongqing: Chongqing University, 2011
- [9] 林闯, 田立勤. 可信网络中用户行为可信评价的研究[J]. 计算机研究与发展, 2008, 45(12):2033-2043
Lin Chuang, Tian Li-qin. Research on User Behavior Trust in Trustworthy Network[J]. Journal of Computer Research and Development, 2008, 45(12):2033-2043
- [10] 温博为. 可信计算平台技术应用研究[D]. 西安:陕西师范大学, 2013
Wen Bo-wei. Application Research on Trusted Computing Platform Technology[D]. Xi'an: Shaanxi Normal University, 2013
- [11] 王志勇. 基于行为声明的动态可信度量技术研究[D]. 北京:北京工业大学, 2013
Wang Zhi-yong. Dynamic Trustworthiness Measurement Technology Based on Behavior Declaration[D]. Beijing: Beijing University of Technology, 2013
- [12] 曾梦歧, 卿显, 谭平璋, 等. 一种基于标识认证的信任链建立方法[C]//第一届中国可信理论与实践学术会议论文集. 北京:清华大学出版社, 2009
Zeng Meng-qi, Qing Yu, Tan Ping-zhang, et al. Establishment Method of Trust Chain Based on Identity Authentication[C]// Proceedings of the first China credible theory and practice conference. Beijing: Tsing Hua University Press, 2009
- [13] 孙迪. 软件行为可信技术研究[D]. 北京:北京工业大学, 2013
Sun Di. Research on Software Behavior Trustworthy Technology[D]. Beijing: Beijing University of Technology, 2013
- [14] Ruiz J L, Duenas J C, Usro F, et al. Deployment in dynamic environments[C]//DECOR04 (2004). 2004:85-98
- [15] Hall R, Heimbigner D, Wolf A L. Specifying the Deployable Software Description Format in XML: CU-SERL-207-99[R]. Software Engineering Research Laboratory, University of Colorado, 1999
- [16] Hall R S, Heimbigner D, Wolf A L. Evaluating Software Deployment Languages and Schema[C]//14th IEEE International Conference on Software Maintenance (ICSM'98). 1998:177
- [17] Van Hoff A, Partovi H, Thai T. The Open Software DescriptionFormat (OSD)[OL]. <http://www.w3.org/TR/NOTE-OSD>
- [18] Distributed Management Task Force. Common Information Model (CIM) Specification (Version 3. 0)[OL]. http://www.dmtf.org/spec/cim_spec_v30
- [19] Coupaye T, Estublier J. Foundations of enterprise software deployment[C]//Proc. of the Euromicro Conf. on Software Maintenance and Reengineering. Zurich: IEEE Computer Society, 2000
- [20] 王怀民, 唐扬斌, 尹刚, 等. 互联网软件的可信机理[J]. 中国科学 (E辑), 2006, 36(10):1156-1169
Wang Huai-min, Tang Yang-bin, Yin Gang, et al. Trustworthiness Mechanism of Internet Software[J]. Science in China (Series E), 2006, 36(10):1156-1169