软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(1):135−159 [doi: 10.13328/j.cnki.jos.005131]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

信息流控制研究进展*

吴泽智[1,2,3], 陈性元[1,2,3], 杨 智[1,3], 杜学绘[1,3]

[1](解放军信息工程大学,河南 郑州 450001)

[2](密码科学技术国家重点实验室,北京 100094)

[3](河南省信息安全重点实验室,河南 郑州 450001)

通讯作者: 陈性元, E-mail: chxy302@vip.sina.com

摘 要: 信息流控制能够保证数据与隐私端到端安全,一直是信息安全领域研究的重点和难点.为介绍信息流控制相关的研究现状和进展,首先,从基于格、安全类型系统、安全进程代数和自动机这 4 个方面介绍了信息流控制的基本理论与模型;其次,从计算机层次结构由下而上出发,综述了基于硬件、操作系统、虚拟机、高级语言、低级语言、数据库和网络的信息流控制实现方法,并对比了各类研究的特点;然后,结合当今时代前沿技术,分析了信息流控制在云计算、移动互联、大数据和物联网等新技术下的应用;最后,总结了当前信息流控制相关研究中存在的问题,并针对今后该领域的研究趋势进行了展望,对下一步研究工作有一定的参考价值.

关键词: 信息流控制;安全模型;安全类型系统;进程代数;动态污点跟踪

中图法分类号: TP309

中文引用格式: 吴泽智,陈性元,杨智,杜学绘.信息流控制研究进展.软件学报,2017,28(1):135−159. http://www.jos.org.cn/1000-9825/5131.htm

英文引用格式: Wu ZZ, Chen XY, Yang Z, Du XH. Survey on information flow control. Ruan Jian Xue Bao/Journal of Software, 2017,28(1):135−159 (in Chinese). http://www.jos.org.cn/1000-9825/5131.htm

Survey on Information Flow Control

WU Ze-Zhi[1,2,3], CHEN Xing-Yuan[1,2,3], YANG Zhi[1,3], DU Xue-Hui[1,3]

[1](PLA Information Engineering University, Zhengzhou 450001, China)

[2](State Key Laboratory of Cryptology, Beijing 100094, China)

[3](He'nan Province Key Laboratory of Information Security, Zhengzhou 450001, China)

Abstract: Information flow control has been a hot and difficult research topic in providing end-to-end data security. This article presents an overview of the field of information flow control. First, the basic theory and models for information flow control are introduced from the perspectives of lattice, security type system, security process algebra and automata machine. Next, working from the bottom up of the computer hierarchy, the implementation methods of information flow control on hardware, operating system, virtual machine, high-level language, low-level language, database and network are introduced, and a comparison among various studies is provided. Then, combining the new technologies of the current era, the applications of information flow control in cloud computing, mobile internet, IoT (internet of thing) and big data are analyzed. Finally, the current problems and the future trends of information flow control are discussed.

Key words: information flow control; security model; security type system; process algebra; dynamic taint tracking

当今时代,信息技术发展日新月异.移动互联网、物联网、云计算、大数据等信息技术不断得到普及应用,即时通信、社交网络、电子商务、互联网金融等领域内的新技术、新产品和新应用层出不穷.同时,随着经济

* 基金项目: 国家高技术研究发展计划(863)(2015AA016006, 2012AA012704)
Foundation item: National High Technology Research and Development Program of China (863) (2015AA016006, 2012AA012704)
收稿时间: 2015-09-01; 修改时间: 2016-03-18; 采用时间: 2016-09-08; jos 在线出版时间: 2016-10-11
CNKI 网络优先出版: 2016-10-12 16:26:52, http://www.cnki.net/kcms/detail/11.2560.TP.20161012.1626.019.html

社会信息化日臻成熟,人们的生活水平和生活方式得到了不断的提高和改变.然而,新技术、新产品和新应用在方便生活的同时也带来了巨大的安全问题.国际上,斯诺登曝光美国间谍活动,警示全球云服务和社交网络的监听风险.苹果公司 iCloud 服务被黑客攻破,造成数百家喻户晓的名人私密照片被盗,再次敲响了移动互联与云服务的安全性警钟.韩国信用卡信息泄漏、索尼影业被黑和朝鲜网络瘫痪事件仍在持续发酵.在国内,携程信息"安全门"事件敲响了电商网站和在线平台的安全警钟.阿里云遭受互联网史上最大规模的 DDoS 攻击.小米 800 万用户数据泄露,12306 网站、智联招聘和考研招生个人信息的泄漏,引发了全社会对于个人隐私的高度关注.中央网络安全和信息化领导小组的成立,宣告网络与信息安全问题已然提升至国家高度.

要实现对数据与隐私的安全保护,加密、访问控制和信息流控制是最有效的方法.加密与访问控制方法是如今最成熟的数据安全技术.加密是指以某种特殊的算法和密钥改变原有的信息数据,使得未授权的用户即使获得了已加密的信息,也无法知悉信息内容.加密机制可保证数据在存储与通信过程的安全性,但不能保证端到端的安全性,因为数据一旦被解密之后,再无相应保护机制能够保证解密后的数据被安全地使用.访问控制是通过限制用户对数据信息的访问能力及范围,保证信息资源不被非法使用和访问.广泛应用的模型包括访问控制矩阵模型和基于角色的访问控制模型等.但访问控制也不能保障端到端的安全需求.例如,若主体 A 允许读取数据 a,在访问控制点 A 读取数据 a 后,A 可以任意使用所读取的数据 a,而系统失去了对数据 a 的控制权限.因而,访问控制并不能有效地控制信息在系统内的传播和间接污染.因而,需要研究可提供端到端安全保证的信息流控制机制,与加密和访问控制一起构成数据与隐私安全的坚实的城墙.

信息流是指信息在系统内部和系统之间的传播和流动,信息流控制是指以相应的信息流策略控制信息的流向.信息流控制策略一般包括数据机密性策略和完整性策略,机密性策略是防止信息流向未授权获取该信息的主体,完整性策略是防止信息流向完整性高的主体或数据.信息流控制机制实现的核心思想是:将标签(污点)附着在数据上,标签随着数据在整个系统中传播(数据派生出的对象也将会继承原有数据标签),并使用这些标签来限制程序间的数据流向.机密性标签可以保护敏感数据不被非法或恶意用户读取;而完整性标签可以保护重要信息或存储单元免受不可信或恶意用户的破坏.为实现信息系统的信息流控制,各国研究人员、政府研究机构、大学高校和互联网企业都展开了大量的研究工作,顶尖的国际会议,如 S&P,CCS,OSDI,NDSS,PLDI,SOSP,SIGCOMM,SIGMOD 和 SS 几乎每年或每两年都有数篇论文研究信息流控制理论及其实现与应用.在与信息流控制相关的期刊和会议学术论文中,较为经典和有影响力的文献总和已经达到数千计.特别是近年来新技术的不断创新与发展,信息流控制研究工作又有了新的理论基础、新的应用领域和新的实现方式.鉴于当前国际和国内还未有信息流控制工作的相关综述文章,本文介绍了信息流控制相关研究工作的研究现状和进展概况,特别侧重于介绍具有代表性的信息流控制系统的结构、组成和实现方法,希望为未来的研究提供一定的参考与帮助.

本文首先给出信息流及信息流控制的定义.第 1 节从基于格、安全类型系统、安全进程代数和自动机这 4 个方面介绍信息流控制的基本理论与模型.第 2 节从计算机层次结构由下而上出发,综述基于硬件、操作系统、虚拟机、高级语言、低级语言、数据库和网络的信息流控制实现方法并对各类研究的特点进行对比.第 3 节分析信息流控制在云计算、移动互联、大数据和物联网等新技术下的应用.第 4 节对信息流控制研究工作有待深入研究的难点和发展热点进行分析与展望.最后是结束语.

1 信息流控制理论与模型

信息流的严格定义是基于熵[1]的形式化定义:系统在 s 状态下执行动作序列 c 后到达的状态 t,x_s 表示系统在状态为 s 时 x 的取值,y_s,y_t 分别表示系统在状态为 s 和 t 时 y 的取值,$H(x_s|y_t)$表示条件熵,如果 $H(x_s|y_t)<H(x_s|y_s)$,则称动作序列 c 使得信息从 x 流向 y.定义可理解为:在系统状态变迁后,由 y 的值可推导出变迁之前 x 值的信息,则说明信息由变量 x 向变量 y 流动.例如,程序语句 if (x) y=0;else y=1,若其 x 等概取值为 0 和 1,则 $H(x_s)$=1,$H(x_s|y_t)$=0.由于 $H(x_s|y_t)$=0<$H(x_s|y_s)$=$H(x_s)$=1,因此信息从 x 流向 y.

为实现有效和正确的信息流分析与控制,需要制定系统的信息流安全策略.信息流安全策略定义什么样的

信息流动行为是合法的,是系统安全的最高级抽象.无干扰策略是一种抽象出安全策略本质的策略,直观上可理解为:若受保护的数据与不受保护的数据之间不存在相互干扰,则认为该系统是安全的.

1982 年,Goguen 和 Meseguer[2]首次提出信息流的无干扰策略,该策略是面向确定性系统的,并且系统的输入与输出是同步的.McCullough[3]扩展了无干扰策略至非确定性系统,分析了异步系统中广义无干扰策略.而后,众多无干扰策略相继提出,包括不可演绎策略[4]、不可推断策略[5]、解析策略、隔离策略[6]和受限策略等,并讨论了各策略在确定性系统与非确定性系统下可组合[7]、可传递[7]、可修正等性质.

为实现信息流安全策略,安全模型的设计是最关键的部分,安全模型用于精确和形式地描述系统的安全特征.根据信息流安全模型描述方法的不同,从基于格的信息流模型、基于安全类型系统的信息流模型、基于进程代数的信息流模型和基于自动机的信息流模型分别加以描述.

1.1 基于格的信息流模型

Denning[8]建立了基于格的信息流模型,通过格结构来形式化地描述信息流动策略、系统状态和各状态之间的转换关系.格的定义如下:如果$(L,\leqslant)$是一个偏序集合,L 中每一对元素 a 和 b 都存在最大下界与最小上界,则称二元组$(L,\leqslant)$是格.基于格可以构建系统的信息流策略,将系统中信息按照其受保护程序可以划分为不同安全等级,每一个级别的信息可以形成一个安全类 SC(security class),则系统中信息流动策略都可以用有限格$(L,\leqslant)$来描述,其中,L 表示安全类的集体.例如,令 $A,B\in L$ 且 $A\leqslant B$.依据机密性策略要求,只能允许信息在一个安全类 A 内部或向高级别安全类 B 流动.

最早基于格的信息流模型是提供数据机密性保护的 BLP 模型,其基本思想仍然是从访问控制的角度研究如何既保证主体有效地访问客体,又使得系统的安全性不致遭到破坏.BLP 模型的基本安全规则是“下读上写”.

- “下读”可描述为:主体 s 允许读客体 o,当且仅当主体安全级 $level(s)$可以支配(dom)客体安全级 $level(o)$;
- “上写”可描述为:主体 s 可以写客体 o,当且仅当客观安全级别 $level(o)$可以支配(dom)主体安全级别 $level(s)$.

模型规则要求所有数据只能按照安全级别从低到高的流向流动,从而保证了敏感数据安全.同理,与 BLP 模型完全对偶的 Biba[9]模型提供了数据完整性保护.由于以上模型要求绝对的信息单向流动,影响了系统的可用性和灵活性,后续基于格的模型引入了主体安全级范围概念、客体安全级范围概念、主体安全级动态调整、可信主体概念以及环策略、低水标策略和高水标策略等.

基于格的信息流模型存在隐蔽通道和降密困难问题.

- 对于隐蔽通道问题,基于格的机密性信息流模型通常采用信道来刻画信息的流动,具体在实际系统中表现为读或者写.然而,信息除了通过主信道之外,仍可以通过其他的旁路信道进行流动,而往往这些旁路信道在基于格的信息流控制模型中并不容易被正确描述和集成,例如 BLP 模型;
- 第二,安全模型中一个重要的研究内容是模型中降低安全等级策略问题,这涉及到模型在现实生活中的可用性和灵活性.基于格的信息流模型一般设置一个降密器,高密级信息可以通过降密器流向低密级信息.然而,这一性质并不满足格的传递性,即,高密级信息不能直接流向低密级信息.

1.2 基于安全类型系统的信息流模型

类型系统用于定义编程语言中的数值和表达式的不同类型以及如何操作这些类型和类型间如何互相作用,类型可以表达具有特定意义和目的一个值或者一组值.简单类型化λ演算为程序类型系统研究提供了一个基本框架,其研究的基本内容包括类型和项的构造、上下文、类型化规则及性质、约简规则及性质等.随着面向对象思想的广泛应用,又产生了多态λ演算[10]、带子类型系统和对象演算系统等.安全类型系统则在已有类型系统中加入安全类型,并建立相应的安全类型规则和安全子规则,如果程序在安全类型规则下是良类型(well-typed)的,那么安全类型系统的可靠性保证了程序一定满足系统安全策略.其安全性证明一般采用无干性证明方法.

Volpano[11]首先将类型系统引入程序安全领域,他提出的安全类型系统包括 3 种不同类型:第 1 种是安全类

型τ,第 2 种是表达式和命令类型π,第 3 种是短语类型ρ.安全类型满足偏序($\leqslant$)关系,其语法如下所示:

- $\tau::=s$;
- $\pi::=\tau|\tau\ proc(\tau_1,\tau_2\ \text{var},\tau_3\ acc)|\tau\ cmd$;
- $\rho::=\pi|\tau\ \text{var}|\tau\ acc$.

然后给出了安全类型系统的类型陈述规则和类型判断规则,如$\lambda;\gamma\vdash n:\tau$表示 n 在安全类型系统环境下具有安全类型τ.类型判断规则形如:

$$\text{(rulename)}\quad \frac{Premise_1;Premise_2;...;Premise_n}{Conclusion}.$$

上式左边为规则名,横线上方为一组前提条件,下方为结论,前提和结论都以判断式形式给出.例如,

- 显式信息流赋值(ASSIGN)规则,为保证显式流 e'到 e 的安全,e'的安全级别必须大于等于 e 的安全级;
- 隐式信息流条件(IF)规则,为保证隐式流 e 到 c 或 c'的安全,c 和 c'的安全级必须大于等于 e 的安全级.

$$\text{(ASSIGN)}\quad \frac{\lambda;\gamma\vdash e:\tau_{var},\ \lambda;\gamma\vdash e':\tau}{\lambda;\gamma\vdash e:=e':\tau_{cmd}},$$

$$\text{(IF)}\quad \frac{\lambda;\gamma\vdash e:\tau,\ \lambda;\gamma\vdash c:\tau_{cmd},\ \lambda;\gamma\vdash c':\tau_{cmd}}{\lambda;\gamma\vdash \text{if } e \text{ then } c \text{ else } c':\tau_{cmd}}.$$

因此,程序开发者只需根据系统自身安全需求构造出安全类型系统并遵循这些安全类型系统规则编写程序,就能实现程序变量级别的信息流控制,满足安全策略的系统.此后,相关研究将类型系统扩展到多线程的命令语言、面向对象的顺序语言等.

基于安全类型系统的信息流控制模型存在完备性问题和精确性问题.

- 第一,安全类型系统往往是不完备的,因为安全类型系统与编程语言紧密相关,不同的编程语言采用不同的类型系统;而且在编程语言中增加了一些新的表达式后,需要扩展已有安全类型系统以处理新的表达式;
- 第二,安全类型系统可以有效地实现静态安全类型检查,但存在精确性较差的问题[12].例如 $L:=H;L:=0$,其中,L 是低安全级变量,H 是高安全级变量.在处理语句$L:=H$时,安全类型系统将拒绝该语句,因为赋值存在从高安全级到低安全级的信息流.然而,该程序显然满足不干扰,因为后一语句消除了此信息流.但基于安全类型系统的方法将拒绝任何带有不安全的子程序的程序.

1.3 基于进程代数的信息流模型

进程代数是一类使用代数方法研究通信并发系统的理论泛称,是并行和分布式系统理论中十分重要的研究领域.进程代数能够完全以代数的形式来描述系统的行为,有效地刻画并发系统的非确定性、通信递归抽象分流和死锁等行为.Foley[13]最先在 CSP(communication sequential process)框架下研究了信息流的无干扰模型,定义了基本无干扰属性和广义无干扰属性.随后,Focardi 在 CCS(calculus of communication systems)框架下研究了在内信息流安全性质,并提出安全进程代数(security process algebra,简称 SPA)[14],将安全概念与系统等价这一概念联系起来,以进程代数作为工具,对安全性质的理解进一步深化.SPA 将 CCS 有名动作集划分为两个不相交的高级动作集 Act_H 和 Act_L,可描述多级安全系统中不同安全级别之间的信息流关系.其语义和语法如下所示:

表达式	::=m,n	信道名
	| x,y	数据值
	| P,Q	进程、系统资源
系统事件	::=$m(x)$	从公开信道 m 发送 x
	::=$m\langle x\rangle$	从公开信道 m 接收 x

	::=^$n(y)$	从内部信道 n 发送 y
	::=^$n\langle y\rangle$	从内部信道 n 接收 y
	::=P/x	x 事件在 P 中隐藏
	::=$P\backslash x$	x 事件在 P 中禁止
	::=t	系统内部不可见事件
操作结构	::=$a.P$	发生 a 后执行 P
	::=$[a=x]$	a 满足条件 x 继续执行
	::=$P\|Q$	P 与 Q 并发执行
	::=$P+Q$	P 与 Q 选择执行

在安全进程代数框架内分析模型无干扰性质时,其实质表现为系统或进程某个动作子集上的行为与另一个动作子集上行为的相关性.由此,在迹语义下可定义非确定性无干扰(non-deterministic non interference,简称NNI):$(E\backslash_I Act_H)/Act_H \approx_T E/Act_H, E/Act_H$ 表示将进程 E 中所有的高级别动作都隐藏起来,是系统的低级观察视图;$E\backslash_I Act_H$ 表示禁止进程 E 中所有的高级别输入动作的执行.非确定性无干扰性质的基本思想是:高级进程是通过高级输入来影响系统输出的,如果高级的输出动作与高级的输入动作相关,那么在保护了进程高级输入动作集的同时也就保护了相应的高级输出动作集;如果高级的输出动作与高级的输入动作无关,那么即使低安全级的进程能够从低级观察视图中获得高级进程输出动作的信息,也还是不能获得任何有关高级进程行为的信息.其而后又定义了强非确定性无干扰(string NNI,简称 SNNI)、基于互模拟非确定性无干扰(bisimulation NNI,简称BNNI)、基于互模拟强非确定性无干扰(bisimulation SNNI,简称 BSNNI)、基于互模拟组合不可演绎性无干扰安全属性(bisimulation-based non deducibility on compositions,简称 BNDC)、基于互模拟组合不可演绎性强无干扰安全属性(strong bisimulation-based non deducibility on compositions,简称 SBNDC).为验证系统是否满足相应的安全属性,众多自动化验证工具被开发出来,应用较为广泛的工具包括 CoSec[15],CVS,CoPS[16],MWB[17]和FDR2[18]等.

Aldini[19]将进程代数扩展到概率系统,形成概率安全进程代数,揭露了由系统概率行为引起的信息泄漏.文献[20,21]在此基础上进一步分析了各种基于复合的不可演绎的概率无干扰信息流属性,以解决信道过滤和安全降密等安全问题.

基于进程代数的信息流控制模型能够直接刻画系统中的信息流控制与无干扰问题,但存在安全性验证困难问题和系统完全实现困难问题.首先,在采用自动化工具验证用进程代数描述的安全模型时,所产生的系统状态图动辄包含上万个节点和上万条边,导致自动化验证工具不响应或者验证时间过长,因而,此类方式只适应于安全模型局部性的验证;其次,安全模型在实际实现时,由于受技术和实际物理条件的限制,并不能保证安全模型中所考虑到的隐蔽通道都被关闭.

1.4 基于自动机的信息流模型

有限状态自动机(finite stateantomata,简称 FS)简称有限自动机,是具有离散输入/输出系统的数学模型,广泛应用于程序语言的设计和实现中.同时,在安全模型形式化验证领域也有着重要的应用.它具有有限数量的状态,用此来记忆过去输入的有关信息,并根据当前的输入确定下一步的状态和行为.一个有限自动机可等价地看作是一个系统行为(事件)驱动的状态转换图,系统的行为(事件)包括输入事件、输出事件和内部事件.因而,信息流安全策略可以由系统状态转移规则来进行形式化描述.基于自动机的信息流模型可简要描述如下.

系统 M 由系统状态集 S、动作集 A、输出动作集 O 和安全域集 D 这 4 个集合以及定义在这 4 个集合上的4 个函数组成,4 个函数分别是:

(1) 单步状态转换函数表示在动作集 A 下系统的状态转换,可表示为 $step:S\times A\to S$;

(2) 系统运行函数表示在动作集 A 下系统的状态转换,可表示为 $run:S\times A^*\to S$;

(3) 系统输出函数表示在动作集 A 下的系统输出,可表示为 $output:S\times A\to O$;

(4) 主域函数表示动作集 A 所属的安全域,可表示为 $dom:A\to D$.

在传递无干扰模型中,定义~为安全域 D 上的干扰关系.例如,对于 $u,v\in D$,$u\sim v$ 表示 u 域和 v 域之间相互干扰;从信息流角度看,则说明信息从 u 域流向 v 域.为定义状态间无干扰安全,还需要定义辅助函数 $purge:A^*\times D\to A^*$.

对于动作 $\alpha\in A^*$,$v\in D$,$purge(\alpha,D)$表示从动作序列α中删除所有从不干扰域 v 所发出的动作后的动作序列,可表示为

$$purge(a^\circ\alpha,v)=\begin{Bmatrix} purge(\wedge,v)=\wedge \\ a^\circ purge(\alpha,v), \quad \text{if } dom(a)\sim v \\ purge(\alpha,v), \quad \text{otherwise} \end{Bmatrix},$$

则系统安全可表达为 $output(run(s_0,\alpha),a)=output(run(s_0,purge(\alpha,dom(a))),a)$.当系统 M 执行了动作序列α到达状态 s 后,安全域 v 发起动作 a 对系统进行观察,以确定序列α与序列 $purge(\alpha,dom(a))$是否相同:如果不相同,则说明在序列α中必然含有 $purge(\alpha,dom(a))$中不存在的动作 b,并且 b 在某状态 s 下影响了动作 a 的观察结果,系统不安全.

最早由 Goguen 与 Meseguer 提出了基于自动机理论的确定性无干扰模型,Sutherland[22]研究了非确定系统的无干扰模型,Gargdey[23]基于时间自动机分析了时间系统中的各种无干扰安全属性以及它们之间的强弱关系, Philippou[24]则使用概率时间自动机建立了概率时间无干扰模型.

2 信息流控制机制

信息流控制最核心的部分是信息流跟踪与控制方式.鉴于已有信息流控制研究相关工作众多,本文仅选取相关工作中最具代表性的研究工作进行总结表述.以计算机层次结构自下而上为线索,分别从基于硬件的信息流控制、基于操作系统的信息流控制、基于虚拟机的信息流控制、基于语言的信息流控制、基于数据库的信息流控制和基于网络的信息流控制这 6 个方面出发,分析信息流控制机制的实现方法及其优缺点.

2.1 基于硬件的信息流控制

基于硬件的信息流控制的核心思想是:扩展 CPU 寄存器和内存(高级缓存),添加相应位数保存数据标签信息.Chen[25]基于 SimpleScalar 实现了信息流跟踪与检测,其核心包括内存与寄存器扩展、信息流跟踪和信息流检测这 3 个部分.

- 第一,扩展内存以实现污点的标记.对于内存中每 Byte 进行每 bit 的标记,同时也扩展 L2 和 L1 级缓存、CPU 和寄存器的标记.在访问或存储内存数据时,数据与污点通过扩展的数据总线一起传送;
- 第二,加入 4 种类型的 ALU 指令以传播寄存器污点.对于每条可能引起污点传播的指令,区分数值计数逻辑和污点传播逻辑,并由不同的硬件处理部分进行计算;
- 第三,在指针数据流和控制流处检测数据污染状态.对于不满足 SimpleScalar 指针处理逻辑的数据污染状态,CPU 将抛出安全异常.

Minos[26]在硬件层次上实现了一个 Biba 低水标数据完整性保护方法,以有效地防止控制数据的攻击.SecVerilog[27]在 Verilog 的基础上提出了一个具备了安全类型系统的硬件设计语言(hardware description language,简称 HDL),可以静态分析硬件层次信息流,关闭因内存(缓存)共享而导致的时间隐蔽通道,保护进程和虚拟机间通信的信息流安全,以有效防止缓存探测的攻击[28].Raksha[29]考虑了不同型号处理器上进行信息流跟踪改造的兼容性问题,设计了一个协处理器,专门处理信息流跟踪问题,而主(或者其他多核)处理器进行正常数据处理工作.Raksha 为信息流跟踪与控制设计了 9 种专用寄存器:标记状态寄存器、标记传播寄存器、标记检测寄存器、客户操作寄存器、引用监控寄存器和用于标记异常处理的 4 类寄存器.其中,标记状态寄存器用于维护数据标记,标记传播寄存器用于维护标记传播策略,标记检测寄存器用于策略实施与检查,异常处理寄存器主要用于记录各类标记传播异常所发生的程序计数器(PC).Raksha 通过硬件与软件层改造,能够有效地防护诸如 SQL 注入攻击、内存共谋攻击、缓冲区溢出攻击和跨站脚本攻击等.其他相关工作还包括 PIFT[30]、FlexiTaint[31]和文献[32,33]中涉及到的工作.

基于硬件的信息流控制机制系统效率高,跟踪准确且易处理控制流,无需修改二进制可执行文件代码.但其

只能看到硬件层的信息,对上层应用数据过于透明,使其只适应于信息流跟踪而不适应于信息流控制;而且,硬件的改造成本较大,推广较难.

2.2 基于操作系统的信息流控制

操作系统级别的信息控制核心思想是对系统所有资源进行安全标记,并依据安全标记关系约束和控制信息流.系统资源包括系统进程与线程、系统内存、文件、进程间通信数据、系统设备等.

最广泛应用的基于操作系统的信息流控制系统是 SELinux[34].SELinux 实质上是一种基于域-类型(domain-type)模型的强制访问控制(MAC)安全系统,并且支持多级安全.它由 NSA 设计成内核模块,并包含到主流 Linux 内核中.关于 SELinux,已有资料较为完备,这里不再赘述.Flume[35]实现了对操作系统关键资源信息流的控制,它使用 tags 标记系统污点数据,labels 标记主客体的机密性、完整性和能力,其系统规则包括标记管理规则和信息流动规则.其系统结构如图 1 所示,其中,阴影部分是 Flume 系统可信计算基.引用监视器负责跟踪和处理进程标记调整和系统调用监控,主要与专用进程孵化器、远程标记注册器和用户空间文件服务器相交互.但 Flume 系统的进程标记采用显式调整方法而未采用污点传播和跟踪的思想,使得可用性和精确性大为下降,而且对于每个应用程序的开发人员需要增加相应的标记调整功能以支持信息流控制,对应用软件与系统兼容性要求较高.

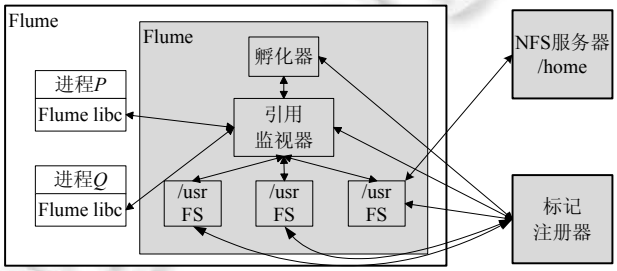

Fig.1 System structure and composition of Flume

图 1 Flume 系统结构与组成

Asbesto[36]系统是较早基于污点传播思想实现了基于操作系统的信息流控制.污点传播主要体现在系统进程通信方面,包括管道(pipe)、有名管道(named pipe)、信号量(semophore)、消息队列(message queue)、信号(signal)和套接字(socket)等方法.但未考虑对系统关键资源的控制,也未考虑系统资源在进程间的共享问题,而且进程标记的隐式调整存在隐蔽通道问题.HiStar[37]继承于 Asbestos 并进行了改进,可提供进程间内存共享机制,但不能实施最小特权原则.DStar[38]在网络层对数据报文进行加密处理,并在每个本地操作系统加入数据安全机制,可保证互不信任的网络主机系统之间的通信安全.HiStar 中的进程可以通过 DStar 把不受信任的代码安全地、分散式地运行在多台 HiStar 主机上.STBAC[39]提出了一种新的操作系统访问控制模型,可有效跟踪控制污点在网络主机间的传播与污染.GTPM[40]系统拓展了污点传播的语义至广义、增加特殊和例外情况下污点的传播、支持污点标记的隐式调整、关闭污点传播已知的隐蔽通道、赋予了主体可删除自身标记的权限以支持系统持续性和系统主体降密能力与发送能力.上述研究工作对比情况见表 1.

Table 1 Expression ability comparison of information flow model based on the operating system

表 1 基于操作系统信息流模型表达能力对比

	SELinux	Asbesto	HiStar	Flume	GTPM
机密性	支持	支持	支持	支持	支持
完整性	支持	支持	支持	支持	支持
安全属性证明	未证明	未证明	未证明	未证明	已证明
最小特权	支持	支持	支持	支持	支持
职责分离	支持	不支持	不支持	不支持	支持
标签动态调整	不支持	不支持	支持	支持	支持
标记传播方法	不传播	污点传播	污点传播	污点传播	污点传播

Table 1 Expression ability comparison of information flow model based on the operating system (Continued)

表 1 基于操作系统信息流模型表达能力对比(续)

	SELinux	Asbesto	HiStar	Flume	GTPM
系统控制对象	进程文件	进程	进程文件	进程套接字	系统未实现
扩展性	是	是	是	是	是
应用兼容性	是	是	是	否	否
策略易管理性	否	是	是	否	否
隐蔽通道问题	未处理	关闭存储隐蔽通道	存在时间隐蔽通道	关闭能力隐式调整隐蔽通道	关闭广义污点传播隐蔽通道
降密问题	不支持	支持	支持	支持	支持
支持 DIFC	不支持	支持	支持	支持	支持

基于操作系统的信息流控制可以有效地保护操作系统的关键资源,系统实现时较为简单,代码改动量小,加入信息流控制机制后系统效率影响较小.但其保护粒度较为粗糙,不能跟踪进程内部的信息流.

2.3 基于虚拟机的信息流控制

基于虚拟机的信息流控制的核心思想是:修改虚拟机具体的实现方式以加入污点跟踪机制,并在虚拟机实时运行层加入相应的控制机制.虚拟机实现方式包括虚拟机指令解释方式、虚拟机方法间调用方式、虚拟机解释栈格式和虚拟机内存堆格式等.

基于虚拟机的信息流控制的核心是虚拟机指令级别的信息流跟踪,依据虚拟机指令格式的不同,可分为基于栈架构和基于寄存器架构两种指令级信息流跟踪方法.第一,对于基于栈架构的虚拟机,为实现虚拟机指令级别的污点传播,需要扩充用于保存本地变量的变量数组和用于存储临时结果的操作栈以支持污点存取操作;第二,对于基于寄存器架构的虚拟机,为实现虚拟机指令级别的污点传播,需要扩充用于保存本地变量的方法帧和用于存储临时结果的寄存器组以支持污点存取操作.例如,基于寄存器架构的虚拟机指令 add-int/2addr v0,v1.其解释执行同样是取内存数值到寄存器中进行运算并保存数值.引入污点传播机制后,还需要额外完成以下 4 步:首先,通过 *SET_TAINT_FP*(*r*10)保存变量污点的内存地址存入寄存器;接着,通过 *GET_VREG_TAINT*(*r*3,*r*3,*r*10)与 *GET_VREG_TAINT*(*r*2,*r*2,*r*10)获取变量污点值;然后,通过 orr *r*2,*r*3,*r*2 计算更新后的污点值;最后,通过 *SET_VREG_TAINT*(*r*2,*r*9,*r*10)将该污点值存入内存.

Trishul[41]实现了基于虚拟机的信息流跟踪,通过修改 kaffe 虚拟机的堆栈和指令解释方式,实现对 Java 程序执行过程中信息流的跟踪.其另一贡献是实现了对控制流的追踪.Laminar[42]结合了基于虚拟机的信息流控制和基于操作系统的信息流控制方式,不但能够追踪程序变量级别的信息流,而且能够保护部分操作系统的关键资源.其结构如图 2 所示.

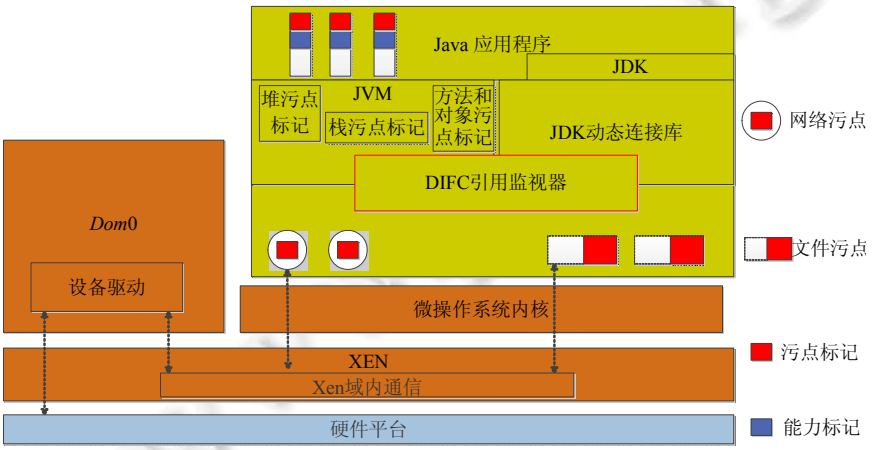

Fig.2 System structure and composition of Laminar

图 2 Laminar 系统结构与组成

用户可以定义相应的 Java 线程能力标签,在 VM 线程初始化时,这些能力标签将被标记在线程中.同时,Lminar 在操作系统关键调用点插入钩子函数,为程序变量和数据结构打上污点标记.这样,VM 中的 DIFC 监控器模块可依据线程能力标签和数据污点标记决定信息流是否合法.

相关研究还包括文献[43–45].基于虚拟机的信息流控制能够跟踪程序内部信息流,且其跟踪粒度可以细化到程序变量和对象,但并不能够保证操作系统资源的安全性,因而需要与基于操作系统的信息流控制结合使用.而且,基于虚拟机的信息流控制在虚拟机解释执行时进行信息流跟踪,对系统效率影响较大.

2.4 基于高级语言的信息流控制

基于高级语言的信息流控制系统的主要思想是编译时实现一个具备机密性和完整性约束的类型系统.编译器读取带有安全标签的源程序代码,检查是否存在类型错误和是否存在违反安全策略的信息流.

Perl 是第一个采用污点传播与检测思想的脚本语言,它实现了程序执行过程中信息流的跟踪与控制,但在其设计之初并非用于保护数据机密性与完整性,而是用于检测和发现系统和程序漏洞,已被广泛用于软件测试和漏洞分析.其主要思想是:将来自于可能恶意源(网络)的数据进行污点标记,跟踪这些数据在系统内的传播,并在其传播到系统关键函数(如系统调用)时显示安全报警.由于它仅使用单一的动态控制机制,其安全策略是隐式且不可配置的,因此其实现时也未能考虑跟踪隐式的信息流.同时,污点跟踪与检测只能发现潜在的漏洞,并不能加以修复或阻止.

最经典的语言级别的静态信息流控制是由 Myers[46]提出的,以 Java 语言为研究对象,在普通类型上增加安全标签,提出扩展的安全类型程序语言 Jif(Java information flow),并在编译时增加了对程序执行时信息流的分析与约束,以保证敏感数据的机密性与完整性.同时,为了克服传统信息流控制方法中标签集中管理所存在的问题并支持互不信任的用户间数据共享,Myers 提出了一个分散式标签模型 DLM(deeentralized label model).Fabric[47]扩展了 Jif 以支持分散式的编程与事务,并通过双重控制-权限控制和信息流控制机制来防止网络中不受信任节点对数据保密性和完整性的破坏,解决了不受信任节点带来的安全问题.Fabric 对应用程序中所有的基本数据类型和对象都采用了分散式标签标记,从而可细粒度地控制程序间的读写操作.Fabric 安全策略可以有效地支持计算的迁移与数据的迁移.Fabric 结合了语言层与系统层的多种安全机制,保证了系统的整体安全.

基于高级语言的信息流控制以安全性分析为中心,结合静态安全性保证和无干扰分析这两类方法,从丰富程序的语言表达能力、探索并发性对安全的影响、分析隐蔽通道问题和重定义安全策略这 4 个方面不断深入研究.

- 其一,丰富程序的语言表达能力.从程序级[11]、程序方法级[48]、程序异常[49]和程序对象级[50]这 4 个层面上建立相应的安全类型系统进行研究;
- 其二,探索并发性对安全的影响.从确定性程序[51]、线程级别并发[52]和分布式程序并发性[53]进行研究;
- 其三,分析隐蔽通道问题.针对已有的隐蔽通道类型,如程序控制流引起的隐式信息流通道、终止隐蔽信道[54]、时间隐蔽信道[55]、概率隐蔽信道[56]和资源耗尽型隐蔽信道等,分析其产生方式,并试图有针对性地将其关闭;
- 其四,重定义安全策略.从程序安全降密[57]、可受理性[58]、安全相对性[59]和定量安全[60]这 4 个方面探讨基于无干扰的新型安全策略.

基于高级语言的信息流控制通过扩展类型语言完成信息流的分析和控制,可追踪细粒度的程序内部数据结构之间的信息流,其优点在于,能在程序编译时就检测程序中的非法信息流,并能够快速定位其语句所在位置,而不必等到程序运行后才被发现,能够有效地减少系统运行时的开销,提高了系统运行的效率.但基于高级语言的信息流控制工作对操作系统中文件和套接字等系统资源的安全性没有保证,同时要求实现嵌入式的类型系统或者一种全新的兼容的高级语言,这对用户和程序开发者来说都存在一定的难度.

2.5 基于低级语言的信息流控制

基于低级语言的信息流控制的核心思想是:在汇编语言或机器语言级插入相关污点跟踪代码,以跟踪程序

执行时的信息流.最常采用的技术是动态二进制插装方法.有别于高级语言,基于低级语言的信息流控制存在两个难点:其一,低级语言一般表现为二进制代码,而污点传播分析必须对这些代码进行精确性建模,而在现代计算机体系中,复杂的指令类型和巨大的指令规模令精确建模非常困难;其二,低级语言缺乏高层语义(没有函数、类型),甚至还存在代码混淆的可能,因而,基于低级语言的信息流控制系统一般依赖于插装平台或者二进制反汇编工具以简化工作量.

二进制代码插装平台在程序验证、反汇编、渗透测试、逆向分析、软件漏洞挖掘[61,62]、数据生命周期分析、软件配置诊断和输入过滤等方面的应用很广,其核心思想是动态污点跟踪:将从特定输入接口传入进程的用户输入数据标记为原始污点数据,通过插装平台插入相应的污点传播代码,可实时、动态地监控污点数据在进程执行过程中的传播情况,并记录污点数据流路径.这些信息可用于进一步的程序行为分析(如语义分析或模糊测试),实现程序验证、反汇编、渗透测试和程序逆向分析.鉴于此,众多研究将基于二进制插装平台的动态污点分析工作引入信息流控制领域.将用户敏感信息标记为原始污点数据,并依据数据污染状态控制其在系统相应范围内的流动.

Libdft[63]基于 pin[64]插装平台实现了基于低级语言的信息流控制,插装工具 pin 由 VM,Code Cache, Instrumentation API 组成,其中,VM 是 pin 的核心组成部分,负责调度插装代码的执行.VM 包含 JIT Compiler, Emulation Unit 和 Dispatcher 等 4 个部分:JIT Compiler 主要是对源程序指令进行即时重编译,Emulation Unit 用来解释一些不能被直接执行的指令,Dispatcher 用来启动 JIT 编译及检测程序代码.Code Cache 是存放即时编译所生成的新二进制代码的缓存空间.Instrumentation API 是由 pin 提供给 pintool,方便 pin 与 pintool 通信的接口.

Libdft 系统框架如图 3 所示,Libdft 则利用 pintool 可透明地提供机器指令级别的信息流跟踪与检测.

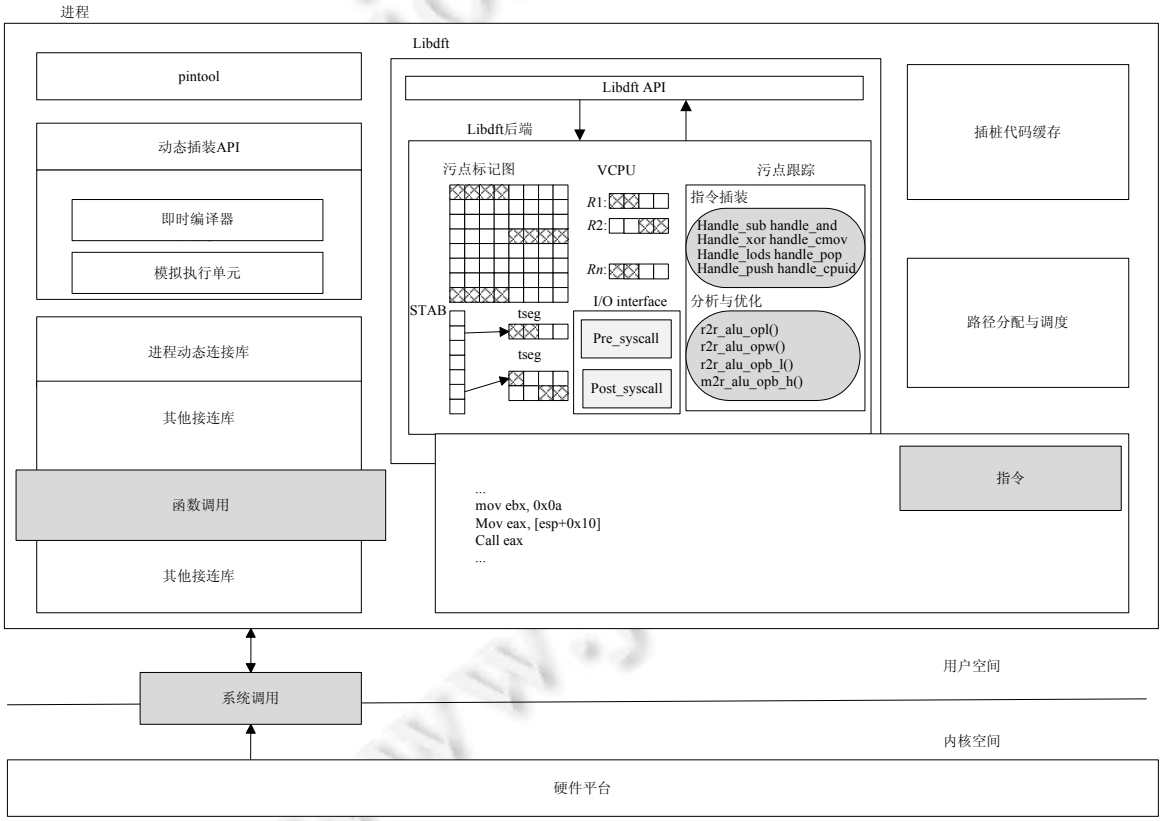

Fig.3 System structure and composition of Libdft

图 3 Libdft 系统结构与组成

Libdft 采用 taint map 存取污点,对 x86 体系中 8 个通用寄存器的污点保存在 vcpu 结构中,对内存污点采用两种标记方式:每位标记和每字节标记.图 3 中,方框内深色部分是 Libdft 提供给用户的 API 接口,通过这些接口,用户可以方便地定义污染源、污染传播方式和污点信息检测点.

类似工作还包括 BAP[65]基于 bitblaze[66]、DTA++[67]基于 bitblaze、Dytan[68]基于 pin、lift[69]基于 StarDBT[70]和 TaintCheck[71]基于 Valgrind[72]以实现污点跟踪与分析.基于低级语言的信息流控制能够实现机器指令级别的污点跟踪和检测,其精确性高.但基于低级语言的信息流控制一般采用动态插装,对系统影响较大.而且,该方法对操作系统中文件和套接字等系统资源的安全性没有保证,同时要求使用者对插装平台工具的使用和机器指令级别的污点跟踪方法有一定的了解,影响了普通用户使用上的方便性.

2.6 基于数据库的信息流控制

基于数据库的信息流控制的核心思想是扩展结构化查询语言(structured query language,简称 SQL),以支持按标签查询,并且扩展数据库中的表(per-table label)、元组(per-tuple)或字段(per-field label),以支持粗细粒度的数据标记.

IFDB[73]基于已有操作系统信息流控制系统 Aeolus[74]实现了细粒度数据库信息流控制,其整体结构如图 4 所示.系统所有通信与查询都需通过 Aeolus API 和数据库驱动发生,应用程序在查询数据时使用按标签查询的方法,在写数据库时,需通过存储程序调用 Aeolus DB API 进行存储.Aeolus DB API 依据信息流策略,决定是否允许该信息流通过.IFDB 提出了按标签查询这一新的数据库查询语言.首先,IFDB 扩展数据库元组加入机密性标签(_label)和完整性标签(_ilabel);其次,每句查询语言也标记机密性和完整性标签(继承查询进程的安全标签);最后,在查询时依据元组标签和查询语句标签返回查询结果.

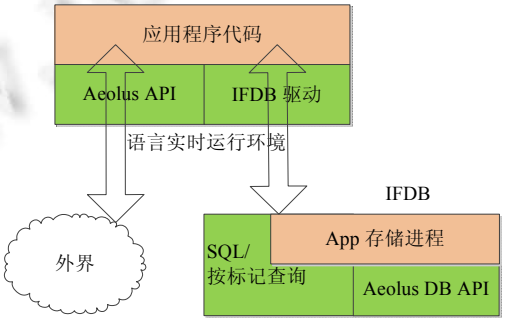

Fig.4 System structure and composition of IFDB

图 4 IFDB 系统结构与组成

DBtaint[75]重写了 SQL 查询语句,在 Perl 和 Java 语言上实现了字段级的细粒度的数据库信息流控制.例如:对于插入语句 INSERT INTO posts (msg) VALUES (STRING),DBtaint 在 JDBC 接口处获取该语句,并检查 STRING 是否含有污点.若含有污点,则将该语句转化为 INSERT INTO messages (msg) VALUES(ROW(STRING, 1)),其中,数值 1 表示字段 STRING 的污点信息.

其他基于数据库的信息流控制研究工作还包括:UrFlow[76]基于 Ur/Web 语言提供了一个静态信息流分析工具,通用安全策略采用 SQL 查询语言定义,用户的独立安全策略可依据实时运行信息来制定;SELinks[77]基于 Links[78]探讨了执行数据库查询和用户自定义函数的标记传播;DIFCA-J[79]基于 Java 提供了一个系统的动态信息流跟踪框架,用户可通过 JDBC APIs 查询和保存数据污点;SELinq[80]基于使用了 F#函数语言风格的嵌入式查询语言 linq,提出了一个包含数据代数类型和模式匹配的安全类型系统,在程序执行时实现数据库与应用程序的边界安全.

基于数据库的信息流控制的其他类似研究工作及对比情况见表 2.

Table 2 Comparison of database-based information flow control

表 2 基于数据库的信息流控制研究对比

系统名称	发表时间	系统主要修改部分	集中/分散	动态/静态	跟踪粒度
mlds[81]	1986	数据库	集中	动态	元组
asd_Views[82]	1988	数据库	集中	动态	关系
SeaView[83]	1988	数据库	集中	动态	字段
ldv[84]	1988	数据库	集中	动态	字段
sintra[85]	1991	数据库	集中	动态	字段
Trusted Oracle[86]	1992	数据库	集中	动态	元组
Trusted Rubix[87]	1992	数据库	集中	动态	元组
Informix Secure[88]	1992	数据库	集中	动态	元组
UrFlow	2010	数据库/语言	分散	静态	字段
DBtaint	2010	数据库	集中	动态	字段
IFDB	2012	数据库/操作系统	分散	动态	元组/进程

2.7 基于网络的信息流控制

基于网络的信息流控制的核心思想是在修改浏览器(例如,提供给脚本语言的 DOM(document object model) API)或脚本语言,以支持信息流控制.在过去的 20 年中,随着基于 Web 应用的爆发式增长,其覆盖领域包括商务、社交、生活、医疗、旅游和理财等生活的方方面面,但随之而来的安全问题也日益严重.因而,网络安全在众多安全研究工作中占据很大部分,而信息流控制的研究在网络安全研究工作中又占据绝大部分.在本节中,仅能选择最具影响和代表性的工作来加以介绍和分析.同时,JavaScript 几乎是所有此类研究工作的对象,因此本节也以 JavaScript 为讨论对象.

基于网络的信息流控制研究工作可分为静态分析、动态监控和安全多次执行(secure multi-execution,简称 SME)这 3 类.前两类与第 2.3 节和第 2.4 节类似,这里不再赘述.SME 的核心思想是多次执行程序,每次执行时使用不同的输入和输出规则,若系统安全,则低级输入只能影响低级输出.Vogt[89]首次将信息流分析引入浏览器安全,它采用动态污点分析跟踪显式信息流,静态信息流分析跟踪隐式信息流,但并未证明系统安全性.Chugh[90]提出了一个阶段式技术以实现信息流控制,阶段式技术的核心思想是:在服务器端完成重量级的静态信息流分析工作,随后,在浏览器客户端沿用静态分析结果并实施轻量级动态污点跟踪.其缺点在于要完全信任内联的脚本.Mash-IF[91]提出了基于 mashups 技术的信息流控制方式,其实现由标记工具、引用监视器和基于静态分析 mashup 组件的降密器组成,通过 Mozilla Firefox 得以实现并测试.FlowFox 基于 Mozilla Firefox[92]实现了 SME,在 alexa 上 500 强网站上的测试结果表明了其可行性和可用性,但对系统效率影响较大,达到 22%,且内存占用开销多达 88%,这也是 SME 方法的缺点.Austin[93]提出了一种新的方法,它结合 SME 方法的正确性和单一执行的高效性,通过引入方位值,可用单一执行模拟多次执行的多个过程将多次执行折叠成一个,有效地减轻了系统负荷.Hedin[94]引入了一个动态类型系统,证明了 JavaScript 信息流的安全性.其他类似工作还包括文献[95−97].其研究对比情况见表 3.

Table 3 Comparison of network-based information flow control

表 3 基于网络的信息流控制研究对比

	Vogt	chugh	Mash-IF	FlowFox	Austin	Hedin
显式信息流	动态	静态	静态	动态	动态	动态
隐式信息流	静态	静态	否	动态	动态	动态
终止不敏感无干扰	否	否	否	是	是	是
时间敏感无干扰	否	否	否	是	否	否
精确性与透明性	否	否	否	是	否	是
浏览器扩展	否	是	是	否	否	否
JavaScript 引擎扩展	是	否	否	是	是	是

综合上述 7 个小节,从信任基础、应用兼容性、系统效率、保护粒度、保护对象、分散或集中、隐蔽通道问题、支持标签数量、形式化验证、流敏感、静态或动态和降密这 12 个方面,分别对比了基于硬件操作系统、

虚拟机、高级语言、低级语言、数据库和网络各研究工作的实现方式与优缺点.其结果见表 4.

Table 4 Comparison of different information flow control mechanisms

表 4 信息流控制机制研究对比

	硬件	操作系统	虚拟机	高级语言	低级语言	数据库	网络
信任基础	系统硬件	操作系统内核及以下	虚拟机及以下	编译器及以下	插装工具及系统硬件	数据库和操作系统及以下	浏览器和操作系统及以下
应用兼容性	兼容	部分兼容	兼容	部分兼容	兼容	部分兼容	部分兼容
系统效率	好	中等	较差	中等	差	中等	差
保护粒度	内存单元、寄存器、CPU 运算单元	进程、文件、套接字	程序变量、对象、方法	程序变量(常量)、对象(数组,字符串)	程序变量、控制变量	视图、数据表、元组、列、单元格	格式流、网络协议
保护对象	内存、寄存器	进程、文件、套接字	解释器栈、堆	程序变量、对象	寄存器	数据库	网页、动作、脚本对象
分散/集中	集中	集中和分散	集中	集中	集中	集中分散	集中分散
隐蔽通道问题	未考虑	关闭资源耗尽型隐蔽通道	关闭控制流型隐蔽通道	关闭部分时间隐蔽通道控制流型隐蔽通道	未考虑	关闭部分资源互斥型隐蔽通道	关闭部分时间隐蔽通道控制流型隐蔽通道
支持标签数量	1bit	32/64bit	32/64bit	1bit	1bit	数据项	32/64bit
形式化验证	否	是	否	是	否	否	是
流敏感	否	否	否	是	否	否	是
静态/动态	动态	动态	动态	动态、静态	动态、静态	动态	动态
降密	否	支持	否	支持	否	支持	支持

3 信息流控制新应用

3.1 云计算环境下的信息流控制

云计算是继分布式计算、对等计算和网格计算之后的一种新型计算模式,并迅速发展成为计算机技术发展的新热点.云计算提供了一种动态可伸缩虚拟化的新型计算资源组织、分配和使用模式,使用户能够随时随地、便捷地、按需应变地访问共享资源池.云计算系统通过 Map-Reduce 编程模型、海量数据管理和虚拟化技术将计算资源以服务的形式提供给终端用户.但是,云计算在为用户提供高效、便捷的云服务的同时,因其自身的开放性、动态性、数据管理的不可控性、云中资源分布的不确定性等诸多特性,使得在云计算环境下数据存储与处理面临着新的安全风险.

根据云计算所提供服务类别的不同,云计算的服务模式可分为软件即服务(software as a service,简称 SaaS)、平台即服务(platform as a service,简称 PaaS)和基础设施即服务(infrastructure as a service,简称 IaaS).为确保云计算的安全性,随后又提出安全即服务(security as a service,简称 SaaS).CloudFence[98]实现了 SaaS 中子集信息流跟踪即服务(data flow tracking as a service,简称 DFTaaS).其系统结构如图 5 所示:① 用户通过用户空间 API 注册 DFTaaS,并获取用户身份统一标识证书(credentials);② 云服务器提供各种服务给用户;③ 在云基础设施框架内,用户隐私数据被标识和跟踪,并记录审计信息;④ 用户通过用户空间 API 接口获取自身数据审计记录.有效地保证了云中用户数据的隔离与安全共享.

云数据库中,数据流具有实时性、连续性、顺序性和数据量庞大等特点,为保证其安全,Xie[99]基于 DSMS (data stream management system)实现了云中数据流的信息流控制系统 CW-DSMS.该系统有如下特点:(1) 支持多用户服务器与用户认证;(2) 以不同安全级别重复执行查询;(3) 全局可信调度器协调所有查询任务;(4) 全局可信解释器支持所有请求集中式查询;(5) 可信流管理器依据不同信息流安全策略输出查询结果.CloudSafety Net[100]基于网络保证了云租户之间的数据与隐私安全.CloudSafetyNet 提供了一个轻量级数据监控框架以支持用户在云网络环境下追踪其隐私数据,其实现包括 3 个主要方法:透明地在 HTTP 请求域内加入安全标签、使用客户 JavaScript 库支持污点跟踪和套接字级别的信息流监控.其在 OpenShift 和 Appscale Paas 平台上实现了原型系统.

结果表明,CloudSafetyNet 能以可忽略的性能影响为代价实现云租户间的数据与隐私安全.类似工作还有SilverLine[101],FlowK[102],FlowR[103]和文献[104,105]中涉及到的工作等.

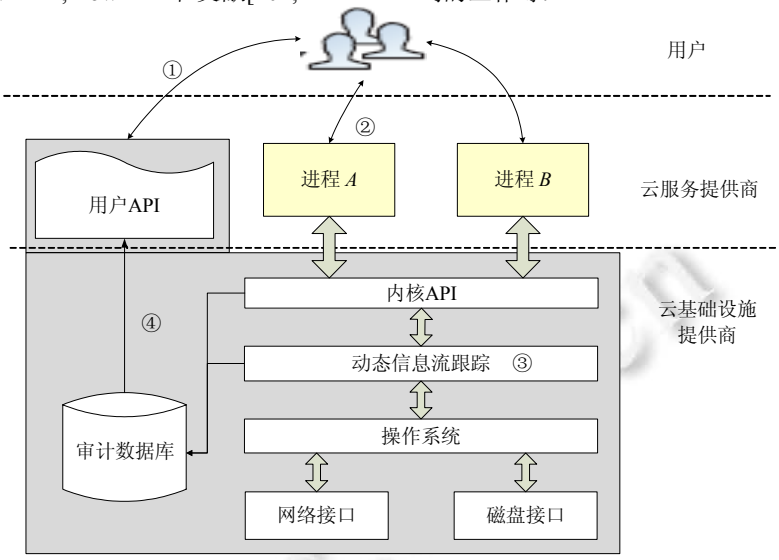

Fig.5 System structure and composition of CloudFence

图 5 CloudFence 系统结构与组成

3.2 移动操作系统的信息流控制

随着移动通信技术的高速发展和智能终端的快速普及,移动电子商务、即时通信、社交、支付等各种新型的业务开始渗入人们的日常生活,移动终端正在从简单的通话工具变为一个综合信息处理平台.但与此同时,随着移动终端越来越多地涉及商业秘密和个人隐私等敏感信息,移动终端承载的数据与隐私安全面临着严重的威胁.当今,移动操作系统主要分为 Android 和 iOS 两大阵营,鉴于 Android 开源且占据几乎 90%的市场,本文主要讨论 Android 操作系统平台下的信息流控制,其研究工作可主要划分为静态信息流分析与动态信息流控制.

3.2.1 静态信息流分析

LeakMiner[106]采用静态污点分析的方法来扫描市场上的应用程序,其结构如图 6 所示.首先,将 Android 应用 APK 文件转换为易于处理的 DEX 字节码,并从清单文件中提取和加载应用权限元数据,同时进行隐私信息污点源分析;然后,分析 DEX 代码并形成程序调用图,不同的调用图链接到相应的根功能节点;最后,在污点传播可能的传播路径敏感处检测数据污点并报告给用户.

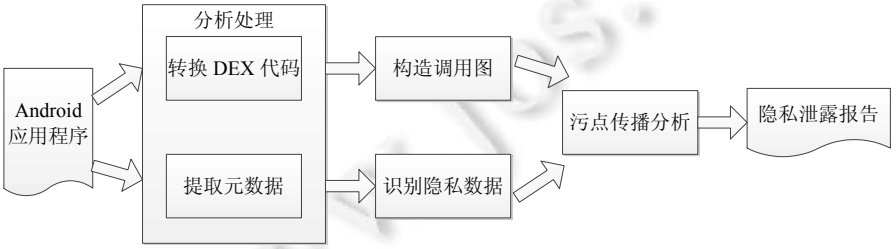

Fig.6 Static information flow analysis process in LeakMiner

图 6 LeakMiner 静态信息流分析流程

类似的相关工作包括:

- SDPL[107]将 218 种 Dalvik 虚拟机指令格式抽象为 61 种,并深入分析应用程序框架层提供的 API 以追

踪所有敏感信息来源.在此基础上设计静态信息流分析框架[108],它由定义方法签名和域签名策略产生器以及一个跟踪显示信息流的安全类型系统组成;

- UAPC[109]从应用程序.APK 文件中提取 Dalvik 字节码格式.DEX 文件;然后解析汇编代码,并在汇编代码关键指令处插入污点传播代码或调用污点传播库,污点传播库定义了敏感域,跟踪污点信息和发出信息泄露警告;最后,将新生成的文件使用相同的私钥重新签名打包;
- Aurasium[110]则采用应用程序自主式检测方法,无需对操作系统进行任何修改;
- FlowDroid[111]对整个应用程序生命期从创建、运行、暂停、杀死、唤醒到停止进行了详细分析,更加准确地提供了数据在应用程序上的整个运行变化过程;
- FlowDroid 针对 Android 系统设计了 DroidBench 实验床以测试污点分析工具的高效性和准确性,结果表明:FlowDroid 的正确性达到 86%,高于商业所使用的工具 AppScan Source 和 Fortify SCA.

上述静态信息流分析工作由于主要分析是在程序运行前完成的,对系统实时运行效率影响较小,且静态分析工作对程序代码覆盖面较大.但部分研究工作使用信息流非敏感算法,不能探测隐式的信息流泄露.同时,对相对错综复杂的系统调用,甚至通过 JNI 调用进入本地方法.以上静态分析工作不能完成信息流的跟踪与检测.

3.2.2 动态信息流控制

动态信息流控制最具代表性的工作是 TaintDroid[112],在源码关键处插入隐私数据污点标记与提取钩子,自动地对个人隐私数据打上污点标记和检测是否存在带有污点标记的数据通过网络等其他路径离开系统;然后,通过修改 Dalvik 虚拟机指令实时地执行逻辑和系统运行中间层等部分,定义了消息级别、程序变量级别、程序方法级别和文件级别的污点跟踪,实现了污点在程序运行过程中的准确传播和污点永久记忆.TaintDroid 提供日志记录数据系统行为,并使用 TaintDroidNotify 向通知栏发送消息,提醒用户存在隐私数据泄露.其系统结构如图 7 所示.

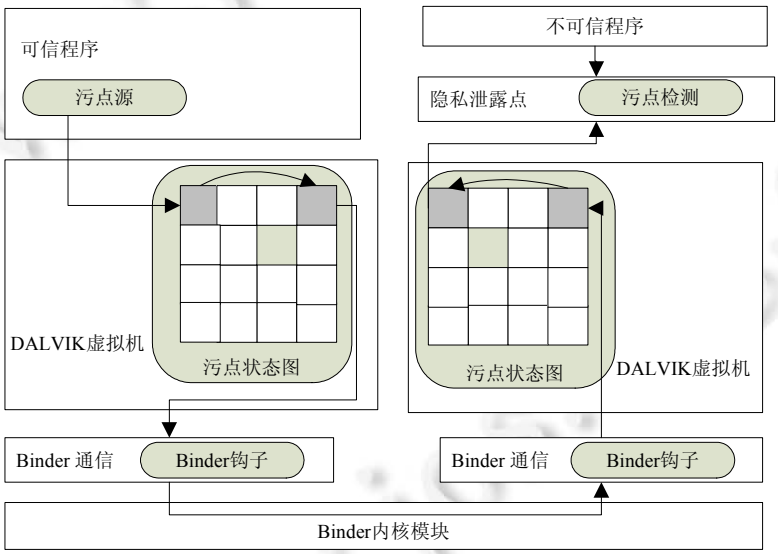

Fig.7 System structure and composition of TaintDroid

图 7 TaintDroid 系统结构与组成

Droidscope[113]是一个基于仿真模拟的 Android 系统恶意软件分析引擎,主要由本地指令跟踪器、Dalvik 指令跟踪器、API 跟踪器和污点跟踪器组成.通过分析 DroidKongFu 和 DroidDream 两个恶意软件,证实该工具能够有效防止恶意软件带来的隐私数据泄露.AppInspector[114]由输入产生器、执行探测器、信息流跟踪器和隐私分析工具组成,可对通过控制流产生隐蔽通道进行跟踪.其他后续研究扩展了隐私监测范围,包括 IMEI、电话号

码、地理位置信息、相册、录音、通信录、短信、通话记录、电子邮件等等,改进了字符串跟踪粒度,防止污点跟踪过程中可能产生的污点爆炸问题.

上述动态信息流控制的相关工作能够检测显式与隐式信息流,也能够跟踪本地指令可能引发的隐私泄露.但由于在系统运行时进行信息流的跟踪和控制,导致系统效率非常差.同时,动态信息流跟踪代码覆盖面比较小,只能检测单一执行路径.

3.3 大数据环境下的信息流控制

随着信息技术的发展,各种新兴网络服务和应用以飞快的速度产生类型各异、数量巨大的数据.同时,在云计算技术的支撑下,大数据已经成为新时代重要的战略资源.目前,大数据已经深入教育、电力、石油、天然气、交通、医疗、金融、IT、商业和政务等各行业,但它在提高经济和社会效益的同时,也为个人与团体的数据与隐私安全带来巨大的安全风险.美国白宫发布了《大数据与隐私保护:一种技术视角》白皮书,表明大数据下个人隐私安全保护已经上升至国家战略高度.

MrLazy[115]基于 MapReduce 提出了一种懒惰的信息流控制方法,它由 4 个子系统组成:亲缘关系跟踪系统、亲缘关系重建系统、静态信息流分析系统和数据标签产生与策略实施系统.其核心思想是:在 Hadoop MapReduce 任务执行过程中,通过捕获亲缘关系实现懒惰的标记传播,如图 8 所示.MrLazy 的输出值通过亲缘关系(图 8 中蓝色虚线表示)与相应的输入记录和输入标签相连接.输出标签定义为相应输入标签的函数.因而在 MapReduce 执行任务中,输入标签不必实时传播.

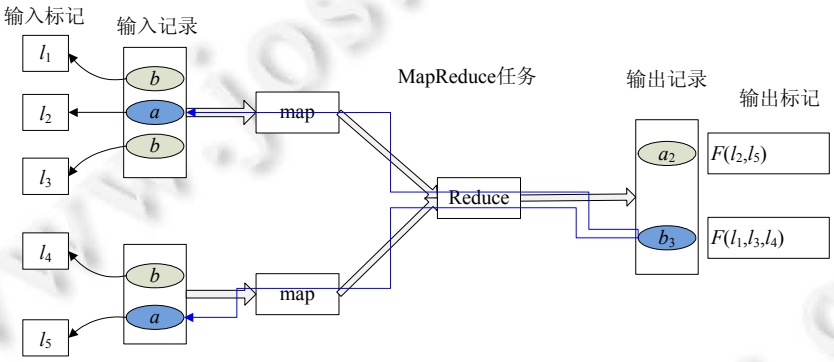

Fig.8 System structure and composition of MrLazy

图 8 MrLazy 系统结构与组成

Thomas[116]为处理大数据环境下的标签管理问题,提出了一个双组成(two-component)标签⟨concern;specier⟩模型.例如,有如下标签⟨medical;bob⟩,表示 bob(标签区分者)的 medical(数据类型)信息.标签产生主要分为 4 个阶段,每个阶段产生的标签用于不同规则下的信息流控制.例如,第 1 阶段将同时 patient_id 所含有的隐私信息归类并标记为⟨*,patient_id⟩.直至最后一个阶段,直观上可理解为大量隐私数据归为同一种类,并采用一种标签标记,从而可以节省大量标签数量与种类,有效地解决了大数据环境下标签管理的复杂问题.

3.4 物联网环境下的信息流控制

物联网是继计算机、互联网之后世界信息产业发展的第三次浪潮.然而,物联网的兴起也让恶意分子将攻击目标转移到可穿戴式设备、智能家电、车载系统与医疗设备仪器等智能设备上,由此窃取机密资料,破坏系统的完整性.

物联网是完全分散式的网络结构,因而数据标签面临被篡改的可能,标签的可信性成为重要的研究问题.Singh[117]针对物联网系统动态性、大规模性和端-端差异性等新特点,提出了一个基于证书的安全信息流模型,其核心在于安全标签如何正确地被产生、使用、传播和管理.其证书结构如图 9 所示:身份证书(identity certificates,简称 IDCs)用于标识设备所属用户,标签证书(tag certificates,简称 TCs)用于信息流控制时机密性保

护、完整性保护、数据共享和安全审计等.身份证书与标签证书可采用单片签名、自主签名和链签名这 3 种方法绑定.

Fig.9 Format of identity certificate and tag certificate

图 9 身份证书与标签证书格式

4 问题与展望

自信息流控制提出以来,虽然在各个应用领域取得了相当丰富的研究成果,但仍然存在许多困难问题并未完全得到解决.同时,随着信息技术的不断创新与发展,信息流控制研究也有了新的发展方向与动力,但也面临着更加艰难的挑战.

1. 全系统安全问题

计算机系统往往只在其安全最薄弱处实施安全保护.已有的信息流研究工作往往针对计算机系统层次结构中某一薄弱部分实施安全保护,并没有提供全系统的信息流安全保护.全系统的安全模型要求不仅能够保护计算机系统层次结构中各组件中的薄弱环节,还能够保护计算机系统层次结构中各组件之间的信息流安全.例如,基于数据库的信息流控制工作能够保证数据库中数据的信息流安全,但数据一旦离开数据库系统,流入操作系统或者系统虚拟机,由于操作系统或系统虚拟机缺乏相应的信息流跟踪和控制机制,这些数据仍然面临泄露或被破坏的安全问题.因而,需要构建一个全系统、多层次的信息流跟踪和控制系统,以保证数据在系统各个层次、各个组件内部和组件之间的信息流安全.实现全系统信息流控制的核心是标签的统一描述及互相转换问题.信息流控制的最关键部分在于安全标签,而已有研究工作的安全标签缺乏相互间的兼容性.基于硬件的安全标签实现有多种方式,如每字节分配 1 位安全标签、每 4 字节(32 位机器)分配 1 位安全标签、每 8 字节(64 位机器)分配一位安全标签.基于操作系统的标签实现则更加灵活,可以是无符号 32 位整数、无符号 64 位整数、长度一定的字符串或者由以上内容构成的结构体.基于语言、数据库和网络的安全标签也有多种不同形式.各类研究都针对自身研究对象的特点设计出各类各异的安全标签,这些安全标签相互间缺乏兼容性,也没有考虑如何相互转换的问题.例如,操作系统的数据安全标签难以转化为硬件数据安全标签.因而,后续研究可以尝试解决标签统一描述和相互转换问题,形成统一的信息流跟踪与控制框架,以实现硬件、操作系统、虚拟机、语言、数据库和网络的全方位、各层次的数据与隐私安全保护.

2. 信息流跟踪系统效率问题

在已有系统上引入动态(由于静态的信息流分析工作一般在编译或者利用相关分析工具完成,不存在系统效率过低问题)信息流跟踪机制后,系统效率将严重下降,这是制约信息流控制广泛应用的最大缺点.例如,在虚拟机实时运行层引入污点跟踪技术后,对于每一条可能传播污点的虚拟机指令,都将插入多条本地指令以完成污点跟踪.这些指令不但降低了系统的实时执行效率,而且通过即时编译后产生的冗余代码也占用了更多的系统内存.

污点跟踪优化的相关研究工作从如下 5 个方面尝试解决以上问题.

- 第一,增加额外的硬件资源,例如增加处理器、寄存器和内存或者将工作传输给远程主机[118,119].这种优化又有两个研究方向:其一,污点跟踪代码仍然嵌入到程序代码中,采用并行化[120]思想将混杂的代码分配到不同的 CPU 进行处理,从而提高系统运行效率;其二,将污点跟踪代码从程序中解耦合[121],ShadowReplica[122]将污点跟踪代码分配到不同的 CPU 处理,不同 CPU 间采用相应的通信机制传递污点警报;
- 第二,将部分动态污点跟踪工作转化到静态分析时完成,主要思想是:通过静态组件将程序源码或二进制文件统一转化为中间代码,并插入相应的污点跟踪指令,最终形成可直接执行并包含污点跟踪的程序代码.文献[123]将污点跟踪过程抽象成污点传播代数,并使用 DAG(direct acyclic graph)进行复写传播和公共子表达式删除等优化.文献[124]结合静态与动态分析,在静态分析时通过编译器保守地完成信息流跟踪代码的插入,有效地减少了动态运行时实施信息流跟踪的负荷.但静态分析缺乏运行时 profile 信息,从而导致优化效果较差,而且缺乏对操作系统和高级语言的支持;
- 第三,依据需求动态关闭与开启污点跟踪功能[125],但动态开启和关闭时机难以确定,且存在开启与关闭时的性能损耗.非专业用户在关闭污点跟踪时可能存在隐私泄漏情况;
- 第四,依据用户或者系统需求,仅对程序部分代码进行污点跟踪[126,127],而对一些非重要代码不进行污点跟踪.其缺点在于无法严格确定哪些程序代码需要进行跟踪,而且容易产生漏报,导致跟踪的精确性较差;
- 第五,依赖于插装平台,在即时编译时进行代码优化.以 valgrind 为例,其即时编译过程如下:首先,将机器码转化为 tree IR;之后,通过复写和常量传播、常量折叠、死代码删除、公共子表达式删除和循环展开等优化方法将 tree IR 转化为 flat IR;然后向 flat IR 中插装,并再次进行常量折叠和死代码消除优化;最后,将 flat IR 转化为本地代码.因而在使用 valgrind 进行指令级别代码插装时,可对形成的 flat IR 再次进行针对性优化.Lift 实现了 Fast-Path,Merged Check 和 Fast Switch 优化.

当前,虚拟化是研究的热门方向.提高基于虚拟机污点跟踪效率的方法有两种:其一,使用即时编译器将热点污点跟踪代码编译并优化形成本地代码;其二,使用提前编译技术完成整个程序代码静态编译、污点代码插装和优化,并在虚拟机实时运行层监控信息流向.基于虚拟机动态污点跟踪的优点有如下 3 个方面.

- 其一,高级语言虚拟机指令含有丰富的高层语义,更利于进行指令级别的污点传播分析;
- 其二,虚拟机具备解释器和即时编译器,解释器可以为即时编译器提供足够的运行时 profile 信息,可自适应地对热点路径进行污点跟踪优化.这样做的优点在于:第一,若冷路径污点跟踪优化算法所占用的时间长于运行代码的执行时间,则显然是入不敷出的,因而不宜进行优化;第二,即使污点跟踪优化算法所占有的时间长于运行代码的执行时间,但由于该热点路径反复执行,即,以一次优化的代价换取多次执行的开销,是值得的;
- 其三,基于虚拟机污点跟踪和底层操作系统污点跟踪是可完美结合的,不需要插装平台的支持和较高专业知识水平用户,具有良好的易用性和用户透明性.

3. 程序执行控制流跟踪问题

信息流跟踪中,控制流跟踪是最难处理的问题,也可以理解为信息流跟踪正确性和精确性问题.正确性是指污点传播代码与程序代码语义的一致性,正确性要求程序代码执行时发生污点传播时要准确跟踪,未发生污点

传播时不能错误跟踪.精确性是指在正确性基础上确保污点跟踪代码是精确的,精确性要求污点跟踪代码尽量简洁、高效.处理好信息流跟踪中的控制流问题,可有效防止污点传播假阳性和假阴性;否则,极容易导致污点爆炸,影响系统可用性和持续性.信息流跟踪工作大多集中于数据流的研究,已有的控制流研究工作存在假阳性问题,即,引入控制流跟踪后误报概率较大.这是由于大部分控制流虽然会产生信息的隐蔽通道,但此类通道极少泄露隐私.例如,程序语句 if (*x*) *y*=0;在控制流跟踪中,将 *x* 污点传播给 *y*,但实际上,若其 *x* 在 0~2^{32} 之间等概率取值,从基于熵的信息流定义来看,其信息流熵可以忽略不计.甚至有些因控制流产生的信息流在系统输出中是不可见的,对于这类控制流,大可不必进行跟踪.后续研究可以探讨跟踪控制流的精确性,通过全局控制流分析,仅跟踪可能产生隐私泄露的控制流.

4. 系统应用兼容性与策略管理问题

信息流控制的另一个制约其广泛应用的问题就是系统应用兼容性和策略管理问题.部分信息流控制系统实现后,需要应用程序开发者重新开发相应的应用程序以支持信息流控制.同时,在策略管理问题上,系统采用简单的安全模型会使得模型表达能力不够,导致无法表达多样的应用安全策略.而采用灵活的信息流控制模型存在表达方式多样的问题,导致要求系统使用者对安全策略的理解和运用能力较高.同时,模型中分布式降密、权限传递、标记动态调整等为安全策略管理和验证带来了难度.因此,下一步需要研究对用户和程序开发者使用透明的信息流控制系统,用户和开发者仅需要制定符合自身系统的安全策略.同时,研究既具备有效表达安全需求又最大限度地方便策略管理的信息流模型,并提供可安全保证信息流的策略验证方法和策略管理方法.

5. 大数据、云计算、物联网、移动互联网和 Web 3.0 环境下新的挑战

新信息技术环境下,采用信息流控制技术实现数据与隐私安全也面临新的挑战.

- 第一,上述传统信息流控制的难点在新环境下仍然是难点.如大数据、物联网环境下,隐私信息种类更加多样,对于信息流标签支持的种类进一步增大.个人隐私包括信息类隐私、通信类隐私、空间类隐私和身体隐私等等,各类隐私数量繁多,表现形式各式各样,而且还需要考虑不同机构人的特性和动态环境,隐私保护的范围难以确定,安全标签的设计也更加复杂.同时,云计算环境信息流跟踪系统效率、物联网下完全分散标签管理和策略管理也是研究的重要方向;
- 第二,在大数据环境下,应用信息流控制技术保护隐私问题需要进行重新思考:其一,顾名思义,大数据就是指海量、庞大的数据,传统的信息流控制技术要求对所有数据都进行标记,而在大数据情况下,又需要增加海量的数据标签,这份系统开销是巨大的;其二,信息流控制技术最致命的弱点是系统效率低下,而大数据环境下更是要求并发地、实时地传播和跟踪海量数据标签,对系统效率影响也是巨大的;其三,大数据环境下,隐私泄露具有聚合性,即,大量非敏感信息可以聚合推导出敏感信息,这要求信息流控制能够发现并阻止此类隐私泄露问题;
- 第三,在面向大数据、云计算、移动互联、软件定义网络、机器学习和 Web 3.0 等新领域下,程序设计语言的安全理论和技术得到了新的发展,要求基于语言的信息流控制能够适合众核多核并行程序设计语言等新型程序设计语言,并且给出信息流安全性无干扰验证与分析.在 Web 3.0 环境下,可继续研究面向 Web service 的信息流跟踪技术实现方法.

6. 其他问题

信息流研究已有 40 多年的历史,在各个研究方向都存在许多研究难题和挑战,值得进一步深入地研究.基于语言的安全信息流挑战包括支持动态的安全策略、支持多线程和分布式系统下并发、支持动态移动寻址安全、支持表达能力更为丰富的计算机编程语言的安全类型系统、支持可信编译和局部可信等.基于数据库的安全信息流挑战包括支持更多的查询语义表达常见标签约束,同时,设计特殊的迭代器,使得每个元组查询处理自身的上下文元组标签.

5 结束语

当今是一个信息化飞速发展的时代,技术发展在方便人们生活的同时也带来了巨大的数据与隐私安全问

题.传统的安全保护机制,如访问控制、加密、防火墙、数字签名、病毒扫描等方法已经不能够满足数据端到端的安全需求.信息流控制能够保证数据与隐私端到端的安全性,与传统的保护机制一起构建数据与隐私安全的保护墙.本文综述了近 40 年来信息流控制领域内的相关理论与系统设计思想,并讨论了未来信息流控制研究的可能方向.希望能够有助于促进该领域的发展和进步.

References:

[1] Shannon CE. A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review, 2001, 5(1):3−55. [doi: 10.1145/584091.584093]

[2] Goguen JA, Meseguer J. Security policies and security models. In: Proc. of the IEEE Symp. on Security and Privacy. IEEE. 1982. 17−24. [doi: 10.1109/SP.1982.10014]

[3] McCullough D. Noninterference and the composability of security properties. In: Proc. of the IEEE Symp. on Security and Privacy. IEEE, 1988. 177−186. [doi: 10.1109/SECPRI.1988.8110]

[4] Wittbold JT, Johnson DM. Information flow in nondeterministic systems. In: Proc. of the IEEE Symp. on Security and Privacy. IEEE, 1990. 144−161. [doi: 10.1109/RISP.1990.63846]

[5] O'Halloran C. A calculus of information flow. In: Proc. of the ESORICS. Toulouse, 1990. 147−159.

[6] McLean J. Security models and information flow. In: Proc. of the IEEE Computer Society Symp. on Research in Security and Privacy. IEEE, 1990. 180−187. [doi: 10.1109/RISP.1990.63849]

[7] McCullough D. Specifications for multi-level security and a hook-up. In: Proc. of the '87 IEEE Symp. on Security and Privacy. IEEE, 1987. 161−161. [doi: 10.1109/SP.1987.10009]

[8] Denning DE. A lattice model of secure information flow. Communications of the ACM, 1976,19(5):236−243. [doi: 10.1145/360051.360056]

[9] Bell DE, La Padula LJ. Secure computer system: Unified exposition and multics interpretation. 1976. http://www.researchgate.net/publication/235107472_Secure_Computer_System_Unified_Exposition_and_Multics_Interpretation

[10] Abadi M, Cardelli L. A theory of primitive objects. In: Proc. of the Programming Languages and Systems (ESOP'94). Berlin, Heidelberg: Springer-Verlag, 1994. 1−25. [doi: 10.1007/3-540-57880-3_1]

[11] Volpano D, Smith G. A Type-Based Approach to Program Security. Berlin, Heidelberg: Springer-Verlag, 1997.

[12] Yao JB. Syntactic based safety information flow analysis [Ph.D. Thesis]. Guizhou: Guizhou University, 2006 (in Chinese).

[13] Foley SN. A universal theory of information flow. In: Proc. of the '87 IEEE Symp. on Security and Privacy. IEEE, 1987. 116−116. [doi: 10.1109/SP.1987.10012]

[14] Focardi R, Gorrieri R. A classification of security properties for process algebras. Journal of Computer Security, 1995,3(1):5−33. [doi: 10.3233/JCS-1994/1995-3103]

[15] Focardi R, Gorrieri R. The compositional security checker: A tool for the verification of information flow security properties. IEEE Trans. on Software Engineering, 1997,23(9):550−571. [doi: 10.1109/32.629493]

[16] Piazza C, Pivato E, Rossi S. CoPS—Checker of persistent security. In: Proc. of the Tools and Algorithms for the Construction and Analysis of Systems. Berlin, Heidelberg: Springer-Verlag, 2004. 144−152. [doi: 10.1007/978-3-540-24730-2_11]

[17] Victor B, Moller F. The mobility workbench—A tool for the π-calculus. In: Proc. of the Computer Aided Verification. Berlin, Heidelberg: Springer-Verlag, 1994. 428−440. [doi: 10.1007/3-540-58179-0_73]

[18] Gardiner P, Goldsmith M, Hulance J, Jackson D, Roscoe AW, Scattergood B, Armstrong P. FDR2 user manual. 2000. https://ora.ox.ac.uk/objects/uuid:e519e725-d4f5-41d7-b061-cacc0647d207

[19] Aldini A, Bravetti M, Gorrieri R. A process algebraic approach for the analysis of probabilistic non interferenee. Journal of Computer Security, 2004,12(2):191−204. [doi: 10.3233/JCS-2004-12202]

[20] Zhao BH, Chen B, Lu C. Analysis for probabilistic information flow security properties. Chinese Journal of Computers, 2006,29(8): 1447−1452 (in Chinese with English abstract).

[21] Li C, Yin LH, Guo YC. Analysis for probabilistic and timed information flow security properties via ptSPA. Journal of Computer Research and Development, 2011,48(8):1370−1380 (in Chinese with English abstract).

[22] Sutherland D. A model of information. In: Proc. of the 9th National Computer Security Conf. Gaithersburg: MD, 1986. 175−183.

[23] Gardey G, Mullins J, Roux OH. Non-Interference control synthesis for security timed automata. Electronic Notes in Theoretical Computer Science, 2007,180(1):35−53. [doi: 10.1016/j.entcs.2005.05.046]

[24] Philippou A, Lee I, Sokolsky O. Weak bisimulation for probabilistic systems. In: Proc. of the CONCUR 2000—Concurrency Theory. Berlin, Heidelberg: Springer-Verlag, 2000. 334−349. [doi: 10.1007/3-540-44618-4_25]

[25] Chen S, Xu J, Nakka N, Iyer RK. Defeating memory corruption attacks via pointer taintedness detection. In: Proc. of the Int'l Conf. on Dependable Systems and Networks (DSN 2005). IEEE, 2005. 378−387. [doi: 10.1109/DSN.2005.36]

[26] Crandall JR, Chong FT. Minos: Control data attack prevention orthogonal to memory model. In: Proc. of the 37th Int'l Symp. on Microarchitecture (MICRO-37 2004). IEEE, 2004. 221−232. [doi: 10.1109/MICRO.2004.26]

[27] Zhang D, Wang Y, Suh GE, Myers AC. A hardware design language for timing-sensitive information-flow security. In: Proc. of the ASPLOS. New York: ACM Press, 2015. 503−516. [doi: 10.1145/2694344.2694372]

[28] Osvik DA, Shamir A, Tromer E. Cache attacks and countermeasures: The case of AES. In: Proc. of the Topics in Cryptology. Berlin, Heidelberg: Springer-Verlag, 2006. 1−20. [doi: 10.1007/11605805_1]

[29] Kannan H. The design and implementation of hardware systems for information flow tracking [Ph.D. Thesis]. California: Stanford University, 2010.

[30] Yoon MK, Chen NSY, Christodorescu M. PIFT: Predictive information-flow tracking. In: Proc. of the ASPLOS. Atlanta: ACM Press, 2016. 246−253. [doi: 10.1145/2872362.2872403]

[31] Venkataramani G, Doudalis I, Solihin Y, Prvulovic M. Flexitaint: A programmable accelerator for dynamic taint propagation. In: Proc. of the IEEE 14th Int'l Symp. on High Performance Computer Architecture (HPCA 2008). IEEE, 2008. 173−184. [doi: 10.1109/HPCA.2008.4658637]

[32] Saal HJ, Gat I. A hardware architecture for controlling information flow. In: Proc. of the 5th Annual Symp. on Computer Architecture. ACM Press, 1978. 73−77. [doi: 10.1145/800094.803030]

[33] Ho A, Fetterman M, Clark C, Hand S. Practical taint-based protection using demand emulation. ACM SIGOPS Operating Systems Review, 2006,40(4):29−41. [doi: 10.1145/1218063.1217939]

[34] Smalley S, Vance C, Salamon W. Implementing SELinux as a Linux security module. NAI Labs Report, 2001,1(43):139−146.

[35] Krohn M, Yip A, Brodsky M, Kaashoek MF, Kohler E, Morris R. Information flow control for standard OS abstractions. In: Proc. of the ACM SIGOPS Operating Systems Review. New York: ACM Press, 2007. 321−334. [doi: 10.1145/1294261.1294293]

[36] Efstathopoulos P, Krohn M, VanDeBogart S, Frey C, Ziegler D, Kohler E, Morris R. Labels and event processes in the Asbestos operating system. In: Proc. of the SOSP. Brighton: ACM Press, 2005. 17−30. [doi: 10.1145/1095810.1095813]

[37] Zeldovich N, Boyd-Wickizer S, Kohler E. Making information flow explicit in HiStar. In: Proc. of the 7th Symp. on Operating Systems Design and Implementation. Berkeley: USENIX, 2006. 263−278.

[38] Zeldovich N, Boyd-Wickizer S, Mazieres D. Securing distributed systems with information flow control. In: Proc. of the NSDI. San Francisco: USENIX, 2008. 293−308.

[39] Shan ZY, Shi WC. STBAC: A new access control model for operating system. Journal of Computer Research and Development, 2008,45(5):758−764 (in Chinese with English abstract).

[40] Yang Z, Yin LH, Duan MY, Wu JY, Jin SY, Guo L. Generalized taint propagation model for access control in operation systems. Ruan Jian Xue Bao/Journal of Software, 2012,23(6):1602−1619 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4083.htm [doi: 10.3724/SP.J.1001.2012.04083]

[41] Nair SK, Simpson PND, Crispo B, Tanenbaum AS. A virtual machine based information flow control system for policy enforcement. Electronic Notes in Theoretical Computer Science, 2008,197(1):3−16. [doi: 10.1016/j.entcs.2007.10.010]

[42] Roy I, Porter DE, Bond MD, McKinley KS, Witchel E. Laminar: Practical fine-grained decentralized information flow control. ACM SIGPLAN Notices—PLDI, 2009,44(6):63−74. [doi: 10.1145/1543135.1542484]

[43] Manivannan K, Wimmer C, Franz M. Decentralized information flow control on a bare-metal JVM. In: Proc. of the 6th Annual Workshop on Cyber Security and Information Intelligence Research. ACM Press, 2010. 64−74. [doi: 10.1145/1852666.1852738]

[44] Chandra D, Franz M. Fine-Grained information flow analysis and enforcement in a Java virtual machine. In: Proc. of the 23rd Annual Computer Security Applications Conf. (ACSAC 2007). IEEE, 2007. 463−475. [doi: 10.1109/ACSAC.2007.37]

[45] Nair SK, Simpson PND, Crispo B, Tanenbaum AS. Design and implementation of a virtual machine based information flow control system. Technical Report, IR-CS-024, Vrije Universiteit, 2007.

[46] Myers AC, Liskov B. Protecting privacy using the decentralized label model. ACM Trans. on Software Engineering and Methodology (TOSEM), 2000,9(4):410−442. [doi: 10.1145/363516.363526]

[47] Liu J, George MD, Vikram K, Waye L, Myers AC. Fabric: A platform for secure distributed computation and storage. In: Proc. of the ACM SIGOPS 22nd Symp. on Operating Systems Principles. ACM Press, 2009. 321−334. [doi: 10.1145/1629575.1629606]

[48] Heintze N, Riecke JG. The SLam calculus: Programming with secrecy and integrity. In: Proc. of the 25th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. ACM Press, 1998. 365−377. [doi: 10.1145/268946.268976]

[49] Myers AC. JFlow: Practical mostly-static information flow control. In: Proc. of the 26th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. ACM Press, 1999. 228−241. [doi: 10.1145/292540.292561]

[50] Banerjee A, Naumann DA. Secure information flow and pointer confinement in a Java-like language. In: Proc. of the IEEE Computer Security Foundations Workshop. New York: IEEE, 2002. 253−267.

[51] Sabelfeld A, Sands D. A per model of secure information flow in sequential programs. Higher-Order and Symbolic Computation, 2001,14(1):59−91. [doi: 10.1023/A:1011553200337]

[52] Smith G, Volpano D. Secure information flow in a multi-threaded imperative language. In: Proc. of the 25th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. San Francisco: ACM Press, 1998. 355−364.

[53] Zdancewic S, Zheng L, Nystrom N, Myers AC. Untrusted hosts and confidentiality: Secure program partitioning. ACM SIGOPS Operating Systems Review, 2001,35(5):1−14. [doi: 10.1145/502059.502036]

[54] Volpano D, Smith G. Eliminating covert flows with minimum typings. In: Proc. of the 10th IEEE Computer Security Foundations Workshop. IEEE, 1997. 156−168. [doi: 10.1109/CSFW.1997.596807]

[55] Agat J. Transforming out timing leaks. In: Proc. of the 27th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. ACM Press, 2000. 40−53. [doi: 10.1145/325694.325702]

[56] Volpano D, Smith G. Probabilistic noninterference in a concurrent language. In: Proc. of the11th IEEE Computer Security Foundations Workshop. New York: IEEE, 1998. 34−43.

[57] Ørbæk P, Palsberg J. Trust in the λ-calculus. Journal of Functional Programming, 1997,7(06):557−591. [doi: 10.1017/S0956796897002906]

[58] Dam M, Giambiagi P. Confidentiality for mobile code: The case of a simple payment protocol. In: Proc. of the 13th IEEE Computer Security Foundations Workshop (CSFW-13). IEEE, 2000. 233−244. [doi: 10.1109/CSFW.2000.856940]

[59] Volpano D, Smith G. Verifying secrets and relative secrecy. In: Proc. of the 27th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. ACM Press, 2000. 268−276. [doi: 10.1145/325694.325729]

[60] Clark D, Hunt S, Malacaria P. Quantitative analysis of the leakage of confidential data. Electronic Notes in Theoretical Computer Science, 2002,59(3):238−251. [doi: 10.1016/S1571-0661(04)00290-7]

[61] Wang TL. Research on binary-executable-oriented software vulnerability detection [Ph.D. Thesis]. Beijing: Peking University, 2011 (in Chinese).

[62] Sun H, Li HP, Zeng QK. Statically detect and run-time check integer-based vulnerabilities with information flow. Ruan Jian Xue Bao/Journal of Software, 2013,24(12):2767−2781 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4385.htm [doi: 10.3724/SP.J.1001.2013.04385]

[63] Kemerlis VP, Portokalidis G, Jee K, Keromytis AD. Libdft: Practical dynamic data flow tracking for commodity systems. ACM SIGPLAN Notices, 2012,47(7):121−132. [doi: 10.1145/2365864.2151042]

[64] Luk CK, Cohn R, Muth R, Klauser A, Lowney G, Hazelwood K. Pin: Building customized program analysis tools with dynamic instrumentation. ACM Sigplan Notices, 2005,40(6):190−200. [doi: 10.1145/1064978.1065034]

[65] Brumley D, Jager I, Avgerinos T, Schwartz EJ. BAP: A binary analysis platform. In: Proc. of the Computer Aided Verification. Berlin, Heidelberg: Springer-Verlag, 2011. 463−469. [doi: 10.1007/978-3-642-22110-1_37]

[66] Song D, Brumley D, Yin H, Caballero J, Jager I, Kang MG, Saxena P. BitBlaze: A new approach to computer security via binary analysis. In: Proc. of the Information Systems Security. Berlin, Heidelberg: Springer-Verlag, 2008. 1−25. [doi: 10.1007/978-3-540-89862-7_1]

[67] Kang MG, McCamant S, Poosankam P, Jager I, Kang MG, Saxena P. DTA++: Dynamic taint analysis with targeted control-flow propagation. In: Proc. of the NDSS. San Diego: Internet Society, 2011. 223−231.

[68] Clause J, Li W, Orso A. Dytan: A generic dynamic taint analysis framework. In: Proc. of the 2007 Int'l Symp. on Software Testing and Analysis. Portland: ACM, Press, 2007. 196−206. [doi: 10.1145/1273463.1273490]

[69] Qin F, Wang C, Li Z, Zhou Y, Wu Y. Lift: A low-overhead practical information flow tracking system for detecting security attacks. In: Proc. of the 39th Annual IEEE/ACM Int'l Symp. on Microarchitecture. Orlando: IEEE, 2006. 135−148. [doi: 10.1109/MICRO.2006.29]

[70] Wang C, Hu S, Kim HS. Stardbt: An efficient multi-platform dynamic binary translation system. In: Proc. of the Advances in Computer Systems Architecture. Berlin, Heidelberg: Springer-Verlag, 2007. 4−15. [doi: 10.1007/978-3-540-74309-5_3]

[71] Newsome J, Song D. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. Network & Distributed System Security Symp., 2005,29(5):720−724.

[72] Nethercote N, Seward J. Valgrind: A framework for heavyweight dynamic binary instrumentation. ACM Sigplan Notices, 2007, 42(6):89−100. [doi: 10.1145/1273442.1250746]

[73] Schultz D, Liskov B. IFDB: Decentralized information flow control for databases. In: Proc. of the 8th ACM European Conf. on Computer Systems. ACM Press, 2013. 43−56. [doi: 10.1145/2465351.2465357]

[74] Cheng W, Ports DRK, Schultz DA. Abstractions for usable information flow control in Aeolus. In: Proc. of the USENIX Annual Technical Conf. Berkeley: USENIX, 2012. 139−151.

[75] Davis B, Chen H. DBTaint: Cross-Application information flow tracking via databases. Proc. of the WebApps, 2010,22(8):2−10.

[76] Huang YW, Yu F, Hang C. Securing Web application code by static analysis and runtime protection. In: Proc. of the 13th Int'l Conf. on World Wide Web. ACM Press, 2004. 40−52. [doi: 10.1145/988672.988679]

[77] Chlipala A, Impredicative LLC. Static checking of dynamically-varying security policies in database-backed applications. In: Proc. of the OSDI. Berkeley: USENIX, 2010. 105−118.

[78] Corcoran BJ, Swamy N, Hicks M. Cross-Tier, label-based security enforcement for Web applications. In: Proc. of the 2009 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2009. 269−282. [doi: 10.1145/1559845.1559875]

[79] Yoshihama S, Yoshizawa T, Watanabe Y. Dynamic information flow control architecture for Web applications. In: Proc. of the Computer Security (ESORICS 2007). Berlin, Heidelberg: Springer-Verlag, 2007. 267−282. [doi: 10.1007/978-3-540-74835-9_18]

[80] Schoepe D, Hedin D, Sabelfeld A. SeLINQ: Tracking information across application-database boundaries. ACM SIGPLAN Notices, 2014,49(9):25−38. [doi: 10.1145/2692915.2628151]

[81] Garvey CE, Papaccio PN. Multilevel data store design. In: Proc. of the AIAA/ASIS/DODCI 2nd Aerospace Computer Security Conf. San Diego: Internet Society, 1986. 58−64.

[82] Garvey C, Wu A. ASDViews [relational databases]. In: Proc. of the '88 IEEE Symp. on Security and Privacy. IEEE, 1988. 85−95. [doi: 10.1109/SECPRI.1988.8100]

[83] Denning DE, Lunt TF, Schell RR. The SeaView security model. In: Proc. of the '88 IEEE Symp. on Security and Privacy. IEEE, 1988. 218−233. [doi: 10.1109/SECPRI.1988.8114]

[84] Dwyer P, Onuegbe E, Stachour P, Thuraisingham B. Query processing in LDV: A secure database system. In: Proc. of the 4th IEEE Aerospace Computer Security Applications Conf. IEEE, 1988. 118−124. [doi: 10.1109/ACSAC.1988.113426]

[85] McDermott JP, Jajodia S, Sandhu RS. A single-level scheduler for the replicated architecture for multilevel-secure databases. In: Proc. of the 7th Annual Computer Security Applications Conf. IEEE, 1991. 2−11. [doi: 10.1109/CSAC.1991.213023]

[86] ORACLE7 Server Application Developer's Guide. Oracle Corporation, 1992.

[87] Hasan M, O'Connor JP, Pryzby G. Trusted distributed rubix. Trusted Distributed Rubix, 1996,23(1):172−191.

[88] Hunt GC, Larus JR. Singularity: Rethinking the software stack. ACM SIGOPS Operating Systems Review, 2007,41(2):37−49. [doi: 10.1145/1243418.1243424]

[89] Vogt P, Nentwich F, Jovanovic N. Cross site scripting prevention with dynamic data tainting and static analysis. In: Proc. of the NDSS. San Diego: Internet Society, 2007. 76−82.

[90] Chugh R, Meister JA, Jhala R, Lerner S. Staged information flow for JavaScript. ACM SIGPLAN Notices, 2009,44(6):50−62. [doi: 10.1145/1543135.1542483]

[91] Li Z, Zhang K, Wang XF. Mash-If: Practical information-flow control within client-side mashups. In: Proc. of the 2010 IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN). IEEE, 2010. 251−260. [doi: 10.1109/DSN.2010.5544312]

[92] De Groef W, Devriese D, Nikiforakis N. FlowFox: A Web browser with flexible and precise information flow control. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. ACM Press, 2012. 748−759. [doi: 10.1145/2382196.2382275]

[93] Austin TH, Flanagan C. Multiple facets for dynamic information flow. ACM SIGPLAN Notices, 2012,47(1):165−178. [doi: 10.1145/2103621.2103677]

[94] Hedin D, Sabelfeld A. Information-Flow security for a core of JavaScript. In: Proc. of the 25th IEEE Computer Security Foundations Symp. (CSF). IEEE, 2012. 3−18. [doi: 10.1109/CSF.2012.19]

[95] Just S, Cleary A, Shirley B. Information flow analysis for javascript. In: Proc. of the 1st ACM SIGPLAN Int'l Workshop on Programming Language and Systems Technologies for Internet Clients. ACM Press, 2011. 9−18. [doi: 10.1145/2093328.2093331]

[96] Bichhawat A, Rajani V, Garg D. Information flow control in WebKit's JavaScript bytecode. In: Proc. of the Principles of Security and Trust. Berlin, Heidelberg: Springer-Verlag, 2014. 159−178. [doi: 10.1007/978-3-642-54792-8_9]

[97] Chinis G, Pratikakis P, Ioannidis S, Athanasopoulos E. Practical information flow for legacy Web applications. In: Proc. of the 8th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems. ACM Press, 2013. 17−28. [doi: 10.1145/2491404.2491410]

[98] Pappas V, Kemerlis VP, Zavou A, Polychronakis M, Keromytis AD. CloudFence: Data flow tracking as a cloud service. In: Proc. of the Research in Attacks, Intrusions, and Defenses. Berlin, Heidelberg: Springer-Verlag, 2013. 411−431. [doi: 10.1007/978-3-642-41284-4_21]

[99] Xie X, Ray I, Adaikkalavan R, Gamble R. Information flow control for stream processing in clouds. In: Proc. of the 18th ACM Symp. on Access Control Models and Technologies. ACM Press, 2013. 89−100. [doi: 10.1145/2462410.2463205]

[100] Priebe C, Muthukumaran D, O'Keeffe D, Eyers D, Shand B, Kapitza R, Pietzuch P. CloudSafetyNet: Detecting data leakage between cloud tenants. In: Proc. of the 6th Edition of the ACM Workshop on Cloud Computing Security. ACM Press, 2014. 117−128. [doi: 10.1145/2664168.2664174]

[101] Mundada Y, Ramachandran A, Feamster N. Silverline: Data and network isolation for cloud services. Proc. of HotCloud, 2011, 23(3):342−356.

[102] Pasquier TFJM, Bacon J, Eyers D. FlowK: Information flow control for the cloud. In: Proc. of the 6th IEEE Int'l Conf. on Cloud Computing Technology and Science (CloudCom). IEEE, 2014. 70−77. [doi: 10.1109/CloudCom.2014.11]

[103] Pasquier TFJM, Bacon J, Shand B. FlowR: Aspect oriented programming for information flow control in ruby. In: Proc. of the 13th Int'l Conf. on Modularity. ACM Press, 2014. 37−48. [doi: 10.1145/2577080.2577090]

[104] Ganjali A, Lie D. Auditing cloud management using information flow tracking. In: Proc. of the 7th ACM Workshop on Scalable Trusted Computing. ACM Press, 2012. 79−84. [doi: 10.1145/2382536.2382549]

[105] Bacon J, Eyers D, Pasquier TF, Singh J, Papagiannis I, Pietzuch P. Information flow control for secure cloud computing. IEEE Trans. on Network and Service Management, 2014,11(1):76−89. [doi: 10.1109/TNSM.2013.122313.130423]

[106] Yang Z, Yang M. Leakminer: Detect information leakage on android with static taint analysis. In: Proc. of the 3rd World Congress on Software Engineering (WCSE). IEEE, 2012. 101−104. [doi: 10.1109/WCSE.2012.26]

[107] Mann C, Starostin A. A framework for static detection of privacy leaks in android applications. In: Proc. of the 27th Annual ACM Symp. on Applied Computing. Sierre: ACM Press, 2012. 1457−1462. [doi: 10.1145/2245276.2232009]

[108] Gibler C, Crussell J, Erickson J, Chen H. Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale. In: Proc. of the Trust and Trustworthy Computing. Berlin, Heidelberg: Springer-Verlag, 2012. 291−307. [doi: 10.1007/978-3-642-30921-2_17]

[109] Xiao X, Tillmann N, Fahndrich M. User-Aware privacy control via extended static-information-flow analysis. In: Proc. of the 27th IEEE/ACM Int'l Conf. on Automated Software Engineering. Palo Alto: ACM Press, 2012. 80−89. [doi: 10.1145/2351676.2351689]

[110] Xu R, Saïdi H, Anderson R. Aurasium: Practical policy enforcement for android applications. In: Proc. of the 21st USENIX Security Symp. Bellevue: USENIX, 2012. 32−44.

[111] Fritz C, Arzt S, Rasthofer S, Bartel A, Klein J, McDaniel P. Highly precise taint analysis for android application. EC SPRIDE Technical Report, TUD-CS-2013-0113, 2013. http://www.bodden.de/pubs/TUD-CS-2013-0113.pdf

[112] Enck W, Gilbert P, Chun BG, Chun BG, Cox LP, Sheth AN. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In: Proc. of the OSDI. Berkeley: USENIX Association, 2010. 255−270.

[113] Yan LK, Yin H. Droidscope: Seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis. In: Proc. of the 21st USENIX Security Symp. Bellevue: USENIX, 2012. 135−146.

[114] Gilbert P, Chun BG, Cox L. Automating privacy testing of smartphone applications. Technical Report, CS-2011-02, Duke University, 2011.

[115] Akoush S, Carata L, Sohan R. Mrlazy: Lazy runtime label propagation for mapreduce. In: Proc. of the 6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 2014). USENIX Association, 2014. 332−345.

[116] Pasquier TFJM, Singh J, Bacon J. Managing big data with information flow control. SIGARCH Computer Architecture News, 2014, 14(9):721−731.

[117] Singh J, Pasquier TFJM, Bacon J. Securing tags to control information flows within the Internet of Things. In: Proc. of the 2015 Int'l Conf. on Recent Advances in Internet of Things (RIoT). IEEE, 2015. 1−6. [doi: 10.1109/RIOT.2015.7104903]

[118] Portokalidis G, Homburg P, Anagnostakis K. Paranoid android: Versatile protection for smartphones. In: Proc. of the 26th Annual Computer Security Applications Conf. ACM Press, 2010. 347−356. [doi: 10.1145/1920261.1920313]

[119] Chen S, Kozuch M, Strigkos T, Falsafi B, Gibbons PB, Mowry TC, Vlachos E. Flexible hardware acceleration for instruction-grain program monitoring. ACM SIGARCH Computer Architecture News, 2008,36(3):377−388. [doi: 10.1145/1394608.1382153]

[120] Ruwase O, Gibbons PB, Mowry TC, Ramachandran V, Chen S, Kozuch M, Ryan M. Parallelizing dynamic information flow tracking. In: Proc. of the 20th Annual Symp. on Parallelism in Algorithms and Architectures. ACM Press, 2008. 35−45. [doi: 10.1145/1378533.1378538]

[121] Chow J, Garfinkel T, Chen PM. Decoupling dynamic program analysis from execution in virtual environments. In: Proc. of the USENIX 2008 Annual Technical Conf. on Annual Technical Conf. Berkeley: USENIX Association, 2008. 1−14.

[122] Jee K, Kemerlis VP, Keromytis AD. ShadowReplica: Efficient parallelization of dynamic data flow tracking. In: Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security. ACM Press, 2013. 235−246. [doi: 10.1145/2508859.2516704]

[123] Jee K, Portokalidis G, Kemerlis VP, Ghosh S, August DI, Keromytis AD. A general approach for efficiently accelerating software-based dynamic data flow tracking on commodity hardware. In: Proc. of the 19th NDSS. San Diego: Internet Society, 2012. 324−335.

[124] Chang W, Streiff B, Lin C. Efficient and extensible security enforcement using dynamic data flow analysis. In: Proc. of the 15th ACM Conf. on Computer and Communications Security. Alexandria: ACM Press, 2008. 39−50. [doi: 10.1145/1455770.1455778]

[125] Portokalidis G, Bos H. Eudaemon: Involuntary and on-demand emulation against zero-day exploits. In: Proc. of the 2008 EuroSys. 2008. 287−299. [doi: 10.1145/1352592.1352622]

[126] Saxena P, Sekar R, Puranik V. Efficient fine-grained binary instrumentation with applications to taint-tracking. In: Proc. of the 6th CGO. ACM Press, 2008. 74−83. [doi: 10.1145/1356058.1356069]

[127] Kim HC, Keromytis AD. On the deployment of dynamic taint analysis for application communities. IEICE Trans. on Information & Systems, 2009,92(3):548−551.

附中文参考文献:

[12] 姚剑波.基于句法分析的安全信息流[博士学位论文].贵州:贵州大学,2006.

[20] 赵保华,陈波,陆超.概率信息流安全属性分析.计算机学报,2006,29(8):1447−145.

[21] 李超,殷丽华,郭云川.基于 ptSPA 的概率时间信息流安全属性分析.计算机研究与发展,2011,48(8):1370−1380.

[39] 单智勇,石文昌.STBAC:一种新的操作系统访问控制模型.计算机研究与发展,2008,45(5):758−764.

[40] 杨智,殷丽华,段淼毅,吴金宇,金舒原,郭莉.基于广义污点传播模型的操作系统访问控制.软件学报,2012,23(6):1602−1619. http://www.jos.org.cn/1000-9825/4083.htm [doi: 10.3724/SP.J.1001.2012.04083]

[61] 王铁磊.面向二进制程序的漏洞挖掘关键技术研究[博士学位论文].北京:北京大学,2011.

[62] 孙浩,李会朋,曾庆凯.基于信息流的整数漏洞插装和验证.软件学报,2013,24(12):2767−2781. http://www.jos.org.cn/1000-9825/4385.htm [doi: 10.3724/SP.J.1001.2013.04385]

吴泽智(1990－),男,湖南长沙人,博士生,主要研究领域为网络与信息安全,信息流控制.

杨智(1975－),男,博士,副教授,主要研究领域为操作系统安全,信息流控制,云计算安全.

陈性元(1963－),男,博士,教授,博士生导师,主要研究领域为网络与信息安全.

杜学绘(1968－),女,博士,教授,博士生导师,主要研究领域为信息系统多级安全,云计算安全.