一种基于信息流分析的文档动态访问控制方法

陆彬彬,王隽竹,唐发根
LU Bin-bin,WANG Jun-zhu,TANG Fa-gen

北京航空航天大学 计算机学院,北京 100083
School of Computer Science and Engineering,Beihang University,Beijing 100083,China
E-mail:lubinbin_fr_79@hotmail.com

LU Bin-bin,WANG Jun-zhu,TANG Fa-gen.Document dynamically access control method base on analysis of information flow.Computer Engineering and Applications,2008,44(3):139-142.

Abstract: In this paper,a method of dynamically document access control based on analysis of the information flow is introduced,which builds on the traditional document access control methods and combines the methodology of graph theory.The algorithms of this method first analyze document information flow dynamically,and then find out the potential approach that leads the leak of information.Consequently,some measures can be taken to prevent it from happening,so that the document content can be protected.Also,there are particular descriptions,analysis and validation for the algorithms via simulating a classic scene.

Key words: electronic documents;access control;information flow

摘 要:提出一种基于信息流分析的文档动态访问控制方法,其核心是防止隐性信息泄露的非法信息流检测算法。算法在传统文档访问控制方法的基础上,结合图论的理论方法,通过对文档信息流的动态分析,找出潜在的信息泄漏的可能途径,进而采取相应的措施防止信息泄漏的发生,从而达到对文档内容保护的目的。以一个典型的场景为例进行模拟,对算法进行了详细的描述,分析和验证。

关键词:电子文档;访问控制;信息流
文章编号:1002-8331(2008)03-0139-04 **文献标识码**:A **中图分类号**:TP393.08

当前,随着计算机和互联网技术的普及,电子文档已成为许多企业、组织最为重要的内部资源。文档内容的安全直接关系到组织的利益、发展甚至生存。电子文档也成为许多内部威胁人员滥用和攻击的目标。对相关涉密文档采取合理、有效的访问控制机制,对文档内容加以保护,杜绝机密信息非法泄漏是保证机构信息安全至关重要的环节。

传统的文档访问控制方法主要是对用户静态地设置访问控制权限,控制粒度较粗,不能根据操作环境的变化做出动态调整,对于已授权用户在其权限范围之内对文档的访问操作无法进行监控和分析,不能阻止由已授权用户单独或合作导致的隐性的信息窃取。目前,国内外对文档访问控制的研究尚没有统一完善的标准和实施方案。已经提出的一些方法,比如:将文档按访问频率动态分类[1],实施监控机制 EM[2],按执行时间段详细划分安全规则[3],扩展基于角色的访问控制[4]等也各有利弊。

本文提出一种方法,根据当前操作环境,动态分析文档间信息流向,找出潜在可能泄露文档内容的信息流途径,进而采取相应措施去防止信息泄漏的发生,可作为对传统文档访问控制方法的补充。

1 典型场景描述

假设某系统环境下有两个用户、三个受控文档,用户对文档的访问权限定义集合如表 1 所示。在本例中,权限定义集合具体实现形式为访问控制矩阵,如表 2 所示。

表 1 用户访问权限定义集合

用户	受控文档	操作权限
s_1	d_1	读写[R&W]
s_1	d_2	无权限[NONE]
s_1	d_3	读写[R&W]
s_2	d_1	读[R]
s_2	d_2	读写[R&W]
s_2	d_3	无权限[NONE]

表 2 访问控制矩阵

	用户 1[s_1]	用户 2[s_2]
文档 1[d_1]	读写[R&W]	读[R]
文档 2[d_2]	无权限[NONE]	读写[R&W]
文档 3[d_3]	读写[R&W]	无权限[NONE]

用户 1 和用户 2 依次按照以下顺序对文档集合进行操作将会造成文档信息的泄漏。具体操作过程如下:

(1)s_1 打开 d_1;

(2)s_1 对 d_2 没有任何访问权限;

(3)s_1 打开 d_3 读取其内容;

基金项目:国家部委科学技术工业委员会“十一五”网规划课题(No.A2120061061)。

作者简介:陆彬彬(1983-),男,硕士生,主要研究方向:网络安全及威胁评估、软件工程;王隽竹(1982-),女,硕士生,主要研究方向:软件工程;唐发根(1953-),男,教授,主要研究方向:软件工程与环境、系统集成。

(4)s_1将从 d_3 中读取出的内容写入 $d1$;

(5)s_2打开 d_1;

(6)s_2打开 d_2;

(7)s_2将 d_1 中的内容复制到 d_2。

图 1 中,矩形框代表用户,圆形框代表文档,标有“OP”的箭头操作表示用户对文档的操作,标有“IF”的箭头表示由于用户操作而引起的文档之间的信息流向,箭头上的数字代表操作的顺序。

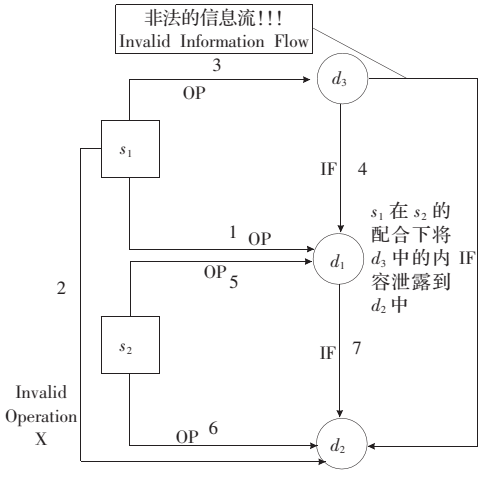

图 1 用户非法操作过程示意

从上述分析和图示可以看到,用户 1 原本对文档 2 没有任何操作权限,然而用户 1 在用户 2 的配合下却可以间接地将文档 3 中的内容泄漏到文档 2 中,传统的设置存取权限列表或者存取控制矩阵的方法无法阻止此类信息泄漏现象的发生。这些静态设定访问控制权限的方法没有考虑不同的操作上下文环境,忽略了不同用户之间协同配合对文档进行非法操作的可能,只能防止最基本的非法操作,稍有经验或者是对系统内部资源比较熟悉的用户可以绕开这些基本权限有目的地进行非法操作。利用本文后续介绍的分析文档信息流的方法可以在一定程度上解决上述问题。

2 算法描述与分析

用户对文档的访问及操作会产生信息的流向,可以根据不同的操作环境(上下文),动态监控和分析不同文档之间的信息流向,找出潜在的可能泄露文档内容的信息流,发现隐性的信息泄漏。具体思想是:根据系统内部的访问权限定义集合为用户抽象出各自的图模型(权限图和信息流图),并对其进行各种操作(如差、交、并、补等集合运算),最终得到可能存在的泄露信息流的集合。这样便可采取相应措施(如对用户的操作附加约束条件)以防止信息泄露的产生,达到保护文档内容安全的目的。这种对文档内容的保护是动态进行的,能够适应用户操作环境的改变。

采用基于信息流分析的方法对文档访问进行控制主要是基于以下考虑:一系列经过授权的、原则上合法的行为仍然有可能导致信息的泄漏,例如前面的例子。文档之间的信息流可以真实地反映用户操作所带来的一系列效果,因为只有用户对文档进行操作才会引起不同文档之间信息内容的流动,即产生信息流。更重要的是,信息流是动态产生的,因而间接地它也可以动态地反映用户对文档的操作,这也为对文档实施动态访问控制提供了可能。

如果一个用户对一个文档有写权限,那么该用户所有其它有权限(无论是读权限还是写权限)访问的文档的内容都有流入到这个文档的可能,换言之,用户对一个文档具有写权限,才有可能产生文档之间的信息流。因此,对于用户具有写权限的文档要进行特别的分析和对待。同时,考虑到文档和信息流这样两个客体的具体特点,用有向图作为算法具体的数据结构非常合适,图的节点对应文档,有向边对应信息流(有向边的方向代表文档内容有流入可能的方向)。

定义 1 信息流 IF(Information Flow)

如果一个用户同时打开文档 d_i 和 d_j,且该用户对文档 d_j 有写权限,则文档 d_i 和 d_j 之间存在一条信息流,记作:$d_i \rightarrow d_j$。

定义 2 信息流图 IFG(Information Flow Graph)

一个信息流图记作:$IFG=(V,E)$,其中 V 是用户当前打开的所有文档的集合,E 是边的集合:$(d_i,d_j)\in E$,当且仅当,$d_i,d_j\in V$ 且 $d_i \rightarrow d_j$。

定义 3 权限图 PG(Privilege Graph)

一个权限图是一个有向图,记作:$PG=(V,E)$,其中 V 是用户具有存取权限的全部文档的集合,E 是边的集合:$(d_i,d_j)\in E$,当且仅当,$d_i,d_j\in V$ 且该用户对 d_j 有写权限。

定义 4 全部文档集合(D)

一个组织或机构内部需要进行权限控制的全部电子文档的集合。

定义 5 当前打开文档集合(OD)

用户当前会话环境下处于打开状态的所有文档的集合,是全部文档集合 D 的子集。

定义 6 全部用户集合

组织或机构内部所有对电子文档有操作权限的注册用户的集合。

定义 7 活动用户

处于会话状态的,正在操作文档集合中文档的用户。

定义 8 活动用户集合

所有活动用户的集合,是全部用户集合的子集。

定义 9 访问权限定义集合(P)

所有访问权限定义内容的集合,可以是访问策略集合、访问控制列表或者访问控制矩阵。这个集合包括了对全部用户的权限定义。

定义 10 单个用户访问权限定义集合

针对单个用户的访问权限定义的集合,是访问权限定义集合 P 的子集。

算法 1 权限图生成算法

输入:

(1)组织内部的全部文档集合 D;

(2)访问权限定义集合 P。

输出:

一个用户的权限图(PG)。

算法描述:

(1)初始化:$PG=(V,E)$,$V=\phi$ 且 $E=\phi$

(2)for all $d\in D$

(3) for all $p\in P$

(4) if p 规定该用户对文档 d 有读权限

(5) 将 v_d 加入顶点集合 V 中;

(6)//v_d 在权限图中代表文档d 的顶点

(7) if p 规定该用户对文档d有写权限

(8) 向v_d 添加一个标记 tag;

(9) for all $v \in V$

(10) If v_d 带有一个标记 tag

(11) 将边 $e=(v,v_d)$ 加入边集合 E;

(12) If v 带有一个标记 tag 且 $v \neq v_d$

(13) 将边 $e=(v_d,v)$ 加入边集合 E;

运行一次此算法,生成一个用户的权限图。若为全部用户(假设数目为 N)生成权限图,则运行 N 次此算法。

算法 2 信息流图生成算法

输入:

(1)用户当前打开的文档集合 OD;

(2)访问权限定义集合 P。

输出:

一个用户的信息流图(IFG)。

算法描述:

生成 IFG 图的算法与生成 PG 图的算法大致相同,仅仅是将全部文档的集合 D 换成用户当前打开的文档集合 OD,此处不再重复。

类似地,运行一次信息流图生成算法,生成一个用户的信息流图。若为当前全部活动用户(假设数目为 M)生成信息流图,则运行 M 次该算法。

通常情况下,权限图只在系统部署时计算一次,但有两个例外:(1)当组织内部的文档集合发生变化(添加或者删除文档)时,要为每个用户重新计算权限图;(2)系统管理员调整了某个用户的某些权限时,要为该用户重新计算权限图。信息流图的情况则有所不同, 其计算依赖于当前打开文档的集合,当用户打开或关闭一个文档的时候, 这个集合就会发生变化,所以只要用户打开或者关闭了文档,都要重新计算信息流图。相对而言,权限图是静态的,信息流图是动态的。

算法 3 非法信息流分析算法

输入:

(1)组织内部的全部文档集合 D;

(2)访问权限定义集合 P;

(3)用户的操作请求。

输出:

非法的文档信息流的集合 I。

算法描述:

假设请求操作的用户为 s_i,系统中的另一个用户为 s_j,$i \neq j$:

(1)生成全部文档的完全图 CG(即假设组织内部的所有文档的读写权限对某个用户都被允许的情况下得到的该用户的权限图);

(2)调用算法 1,计算当前请求操作的用户 s_i 的权限图(PG_{s_i});

(3)对步骤(1)计算得到全部文档的完全图和步骤(2)计算得到的用户的权限图 PG_{s_i} 进行求差(difference)运算,得到当前请求操作的用户 s_i 的权限图的补集($\neg PG_{s_i}$);

(4)调用算法 2,计算当前请求操作的用户的信息流图(IFG_{s_i});

(5)调用算法 1,计算其他某个用户 s_j 的权限图(PG_{s_j});

(6)对通过步骤(4)和步骤(5)得到的结果进行求并(union)运算,得到集合 U,$U=IFG_{s_i} \cup PG_{s_j}$,$i \neq j$;

(7)对 U 和步骤(3)得到的$\neg PG_{s_i}$进行求交(intersection)运算,得到集合 I,$I=U \cap \neg PG_{s_i}$。

算法 4 决策算法

算法描述:

(1)调用算法 3,分析当前请求操作的用户信息流图(IFG)和另一用户的权限图(PG),计算非法信息流集合 I;

(2)若允许此用户请求会导致文档信息的非法泄漏,即集合 I 非空,并且 I 中的边不全是环(指向自身结点的边),则执行以下两种操作之一:

①直接拒绝(deny)该请求;

②在允许该请求的情况下附加一些约束条件(obligations);

(3)对其他所有用户重复步骤(1)和(2)。

每当有用户向服务器端发送一个操作请求,服务器端都应该运行此算法进行检测。

从信息流图和权限图的定义和生成算法可以看出,这两种图模型都是对用户操作权限的抽象,从本质上看,都是由于用户操作文档所引起的文档之间信息流向的一种图形化的表示。它们的区别在于,信息流图是对用户当前操作文档集合的信息流向的描述,而权限图是对用户权限范围内所有文档集合的信息流向的描述。一般情况下,信息流图是权限图的一个子集。特别地,当用户打开了其权限范围内的所有文档时,其信息流图等同于权限图。

3 算法验证与场景模拟

本章利用上述算法对本文前面提到的典型场景进行模拟,演示算法的执行流程,验证算法有效性和正确性。具体检测过程分析如下:

(1)计算全部文档的完全图 CG(算法 3 步骤(1)),如图 2;

(2)计算 s_1 的权限图(PG_{s_1})及其补集($\neg PG_{s_1}$)(算法 3 步骤(2)、(3)),如图 3;

(3)计算 s_2 的权限图(PG_{s_2})及其补集($\neg PG_{s_2}$)(算法 3 步骤(5)),如图 4;

(4)计算 s_1 的信息流图(IFG_{s_1})(算法 3 步骤(4),当前打开文档为 d_1,d_3),如图 5;

(5)计算集合 U 和集合 I 检测非法信息流 $U=IFG_{s_1} \cup PG_{s_2}$(算法 3 步骤(6))$I=U \cap \neg PG_{s_1}$(算法 3 步骤(7))如图 6;

I 中所有的边代表了可能存在的信息泄露途径。在本例中,计算得到的集合 I 非空,它包含一条边(d_1,d_2),所以任何写入 d_1 的内容都有可能被流入到 d_2 中。对 U 分析可知,d_3 和 d_2 之间通过 d_1 存在着一条权限中禁止的信息流途径 $d_3 \rightarrow d_1 \rightarrow d_2$,即可通过此途径造成信息泄漏。为了防止信息泄露的发生,可以直接拒绝用户 s_1 对文档 d_1 的任何写请求(永久撤销写权限)(算法 4 步骤①),以达到保护文档内容的目的。但某些情况下,这样做未免过于严格,也不够灵活,可以通过添加一个约束条件(obligation)来控制用户对文档的访问以防止这种信息泄露(算法 4 步骤②), 即当有其它的文档处于打开状态的情况下,禁止用户 s_1 对文档 d_1 进行写操作(临时撤销写权限),但对文档 d_1 的读权限和对文档 d_3 的读写权限保持不变。这可以简单地通过临时禁用客户端文档编辑器的编辑选项实现。这样,由于用户 s_1 对文档 d_1 没有了写权限,他就不能将文档 d_3 的内容写入文档 d_1 中,进而也就无法在 s_2 的协助下将文档 d_3 中的内容窃取到文档 d_2 中。可以看出,解决问题的关键是切断了文档 d_3 到文档 d_1 之间的信息流($d_3 \rightarrow d_1$),在图 7 中表示为虚线箭头。当文档 d_3 关闭后,用户 s_1 对文档 d_1 的写权限可以恢复。相

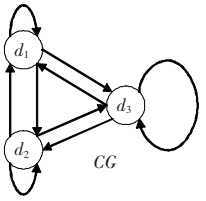

图2 全部文档的完全图

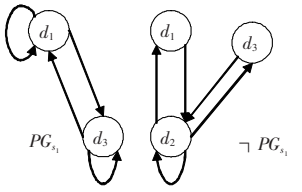

图3 用户1的权限图及其补集

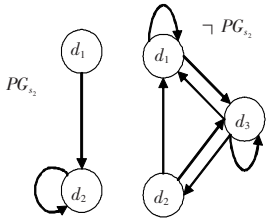

图4 用户2的权限图及其补集

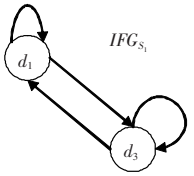

图5 用户1的信息流图

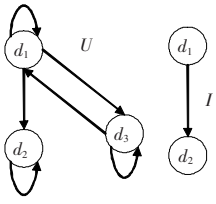

图6 集合 U 和非法信息流集合 I

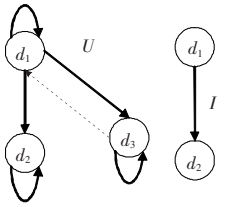

图7 添加附加条件后的集合 U 和集合 I

对于完全撤销用户的某些权限,这种在保持原有权限不变的情况下动态地添加附加规则的方法可更灵活地实现安全的访问控制,并将对用户合法操作带来的不便降到最低。通过分析PG图和IFG图动态生成的附加规则(Obligation)与通过静态策略得到的授权结果(Decision)一起返回给用户,共同控制用户对文档的访问。需要说明的是,上述方法的目的是阻止已有的访问权限定义下可能存在的隐性的信息泄露(这些泄露信息流的产生是静态的访问权限定义的固有缺陷所致,如本文引言所述),使对传统的访问控制方法的补充和完善,并不能阻止通过人为方式故意造成的信息窃取行为。

算法在具体实现时有很大的灵活性,如生成用户权限图所采用的用户访问权限定义集合 P,它可以是访问策略集合、访问控制列表或访问控制矩阵,由系统本身安全策略定义机制决定;决策算法中生成附加规则可通过XACML(eXtensible Access Control Markup Language)中的职责(Obligation)实现,也可通过其它方式去约束用户操作。

4 结束语

本文比较详细地阐述了基于文档信息流分析实现灵活的访问控制机制的方法,主要介绍和分析了相关算法的思想和流程,并通过一个实际的例子演示了算法的执行过程和效果。通过与传统的文档访问控制方法有机结合,可有效地实现对文档访问的动态控制,提高文档访问控制机制的安全性与灵活性。本文的核心是一种思想,在具体实现算法时,可结合不同实际情况加以改进,同时还应权衡考虑服务器端的负载能力、客户端与服务器端的通信开销(包括通信格式、通信频率等)、客户端本身对用户操作文档的监测(系统环境监测、用户行为监测、网络状况监测)等多方面的问题。(收稿日期:2007年7月)

参考文献:

[1] Garg A,Pramanik S,Sankaranarayanan V,et al.Dynamic document reclassification for preventing insider abuse[C]//Proceedings of the 2004 IEEE Workshop on Information Assurance United States Military Academy,West Point,NY,2004.

[2] Schneider F.Enforceable security policies[J].ACM Transactions on Information and System Security,2000,3(1).

[3] Ryutov T,Neuman C.The specification and enforcement of advanced security policies[C]//Proceedings Third Intworkshop,2002.

[4] Bertino E,Bonatti P A,Ferrari E.TRBAC:a temporal role-based access control[J].ACM Transactions on Information and System Security,2001,4(3):191-223.

[5] Clifford A S.A practical introduction to data structures and algorithm analysis[M].张铭,刘晓丹,译.北京:电子工业出版社,2002.

[6] Douglas B W.Introduction to graph theory[M].2nd ed.北京:机械工业出版社,2004.

(上接109页)

表1 不同初始容量不同 qx 序列条件下优化方法哈希性能优化比

q	3	5	6	7	8	9	10	11	12	13	15
$R(C=11)$	0.067	0.999	0.067	0.067	0.067	0.067	0.999	0.987	0.067	0.965	0.999
$R(C=10)$	0.999	0.998	0.999	0.999	0.000	0.999	0.997	0.067	0.999	0.926	0.999

6 结束语

本文根据实际问题要求,通过分析Java哈希表性能,发现了一种导致其性能显著下降 qx 的攻击序列,并设计了一种优化Java哈希表的基于素数序列的哈希表扩张方法(所有测试源码和支持源码可以通过作者E-mail索取,代码在JDK1.4.2上调试通过。方法可用于JDK5.0和6.0,因为这两个版本的Hashtable的实现算法未变)。理论和实践都表明优化方法能有效避免 qx 序列攻击。对随机数据序列,优化方法取得略好性能(当用户指定不当初始哈希表容量时,优化方法性能提高得更为明显)。基于此,提出两个结论:一是用户应该避免指定初始 C,哈希表容量应该交给JDK管理,因为JDK缺省的哈希表扩张序列还是比较好的;二是由于用户程序和库的公开接口都是不便改动的,因此建议采用本文提出的基于素数序列的方法改进Java哈希表性能,使其性能在:改变公开接口的情况下避免受到 qx 序列的影响。(收稿日期:2007年8月)

参考文献:

[1] Sun MicroSystems.Java SE Downloads [EB/OL].[2007-06-15].http://java.sun.com/javase/downloads/index.jsp.

[2] 徐凤刚,许俊奎,潘清.可扩展Hash方法的一种改进算法[J].计算机工程与应用,2006,42(4):95-97.

[3] 陈军,韦鹏程,张伟,杨华千.基于RBF神经网络和混沌映射的Hash函数构造[J].计算机科学,2006,33(8):198-201.

[4] 王张宜,李波,张焕国.Hash函数的安全性研究[J].计算机工程与应用,2005,41(12):18-19.

[5] 毕秀丽,王晅,肖斌,等.一种基于HASH变换的循环散列分档排序算法[J].计算机工程与应用:2006,42(14):50-51.

[6] 唐红,吴勇军,赵国锋.用于特定流匹配的随机矩阵映射Hash算法研究[J].通信学报,2007,28(2):17-22.

[7] Message Passing Interface Forum.The Message Passing Interface Standard[EB/OL].[2003-11-15].http://www-unix.mcs.anl.gov/mpi/.

[8] Sun MicroSystems.Hashtable [EB/OL].[2003].http://java.sun.com/j2se/1.4.2/docs/api/java/util/Hashtable.html.

[9] Thomas H C,Charles E L,Ronald L R,et al.Introduction to algorithms[M].潘金贵,译.北京:机械工业出版社,2007.

[10] Sun MicroSystems.Hashtable[EB/OL].[2003].http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Object.html.