# 电子数据证据收集系统保护机制的研究与实现

孙 波,孙玉芳,张相锋,梁 彬
(中国科学院软件研究所,北京 100080)

**摘 要**: 随着计算机犯罪的不断增加,电子数据取证技术(Digital Forensic Technologies)越来越受到人们的重视.目前对计算机取证技术的研究主要集中于证据提取及证据分析等方面,而对取证机制本身的安全没有考虑,这使得电子数据证据的完整性得不到充分的保障.在对相关研究工作进行分析的基础上,文中提出安全隔离环境是用于保护电子数据取证机制的有效方法,并设计和实现一个安全保护机制——FLOMAC,验证了以上方法是符合实际并有效的.

**关键词**: 电子数据取证; 电子数据证据收集系统; 访问控制; 完整性; 真实性
**中图分类号**: TP309 **文献标识码**: A **文章编号**: 0372-2112 (2004) 08-1374-07

## Research and Implementation of the Protection Mechanism for Digital Evidence Collecting System

SUN Bo, SUN Yu-fang, ZHANG Xiang-feng, LIANG Bin
*(Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)*

Abstract: Research regarding Digital Forensic Technologies has become more active with the recent increases in illegal accesses to computer system. Many researchers focus only on the techniques or mechanisms for evidence detecting and evidence analyzing, without considering the security of forensic mechanisms themselves, and the digital evidence can't be protected completely. Based on the analysis of relative researches, secure area is proposed to protect forensic mechanisms from attacking. A mechanism called FLOMAC has been designed and implemented to evaluate this method. The results demonstrate the advantage in protecting the forensic mechanisms.

Key words: digital forensics; digital evidences collecting system; access control; integrity; fidelity

## 1 引言

近些年来,随着人们对信息技术的依赖性越来越强,计算机犯罪也在不断的增加,此类犯罪造成的后果已经越来越严重[1].在侦破计算机犯罪以及清查系统安全漏洞的过程中,电子数据取证技术显得越发重要[2,3].对于计算机犯罪的取证,就是对计算机数据的取证.因此在对计算机犯罪进行侦察时,完整地从系统中提取电子数据证据以及对其进行固定与保全就显得尤为重要.

电子数据证据不同于传统的犯罪证据,它们更加容易消失和被破坏[4].这使得调查人员很难获得充分的电子数据证据,即使取得了一些证据,其完整性(integrity)和真实性(fidelity)也很难得到证明[5].电子数据证据的这些固有特性严重阻碍了计算机取证技术的发展.为了获得完整可信的电子数据证据,应在敏感主机中预先安装设置电子数据证据收集系统(Electronic digital evidence collecting system,EDECS),用来收集系统中的相关证据,尤其是那些易被损坏和易于消失的数据[6].

然而,EDECS 的某些模块往往存在于被攻击系统之中,因此其系统本身的安全性是首先应考虑的问题.也就是说,只有保证了整个 EDECS 的完整性,才能依据此系统所取得的数据进行后续的取证分析工作.基于这种实际需求,我们提出了一种新的保护机制(FLOMAC),用以确保 EDECS 免受攻击,从而能够正确的完成证据收集的任务.

## 2 现存的证据收集系统保护机制的分析

### 2.1 EDECS 概述

EDECS 系统一般多为 Client/Server 模式,客户端基本都包含了主机收集模块和网络收集模块.它们分别被安装在主机中和网络的入口处,以便收集主机以及网络信息.服务器端大体有与客户端进行通讯的收集模块,有依据相应规则对收集到的电子数据证据进行分类的分类模块,以及用于取证信息备份的备份模块等.

诚然,以上机制并不一定具有绝对的普遍性.但无论何种证据收集机制,其证据收集模块(如主机收集模块和网络收集模块)必须存在于被检测的环境中,这也正是最有可能遭到攻击的环境.因此,为了保证被收集证据的真实性与完整性,证

收稿日期:2003-08-08;修回日期:2004-03-10
基金项目:国家自然科学基金(No. 60073022);国家 863 高科技项目基金(No. 863-306-ZD12-14-2);中国科学院知识创新工程基金(No. KGCX1-09)

据收集模块的保护至关重要.

### 2.2　现存技术分析

**2.2.1　威胁模型**　EDECS 的保护机制使用了一个威胁模型,它包含了整个攻击过程. 最初,我们假设系统是可信的,并且正确配置了证据收集机制. 经过一段时间之后,攻击者掌握了足够的系统相关信息,并对系统进行攻击. 攻击者成功的获取了系统管理员的权限,并迅速攻破了系统中包括内核在内的所有软件,阅读和修改系统中的所有数据(包括存留在系统中的取证数据). 为了避免攻击被发现,攻击者往往使得各种服务正常运行以避免被察觉. 而且这种逃避的时间越长,系统所遭受的损害越大. 此时,系统管理员会通过某种方式发现此攻击. 同时为了避免攻击者对系统造成更大的破坏,将会使系统停止运行. 之后,通过获得的电子数据证据查找线索,分析攻击方法、攻击意图和造成的损失.

**2.2.2　保护需求**　由以上威胁模型可知,当攻击者已知系统中有取证机制(指运行于客户端的电子数据证据收集模块)正在运行时,攻击者很有可能会对取证机制进行攻击,使之失效. 而且当攻击者取得系统管理权限后,他完全可能修改取证模块. 为了避免此类攻击,我们需对以下客体实施保护:(1)对证据收集系统中的所有文件实施保护;(2)对证据收集系统中的所有进程实施保护.

**2.2.3　几种典型保护机制的分析**　尽管文[6]提出了预先在系统中做好证据收集工作的重要性,但并没给出具体的实现. 目前还没有可实际应用的 EDECS,更没有用于保护 EDECS 的机制. 但我们发现日志系统保护机制中的很多思想和方法可以引用到 EDECS 的保护机制中去,并且对日志系统保护机制的研究也比较深入. 因此本节将首先对现存的日志系统保护机制进行分析,最后给出可用于 EDECS 的保护机制.

**(1) Forward Integrity 保护机制**　对于日志系统的保护最早来源于对安全时间戳的研究. 一串时间戳顺序排列,任意时间戳的验证基于上一个时间戳[7]. 这一思想被 Bellare 和 Yee 提炼为一种特性,称之为"Forward Integrity(FI)"特性. 它保证了当系统被攻破后,系统日志的完整性不会被破坏[8].

FI 的主要设计思想是将日志信息流视为一个个具有时间序列的日志信息体$\{m_1, m_2, \cdots, m_l\}$,从而日志文件可分解为一个个具有时间序列的日志记录体$\{e_1, e_2, \cdots, e_l\}$,对日志系统的攻击也就是对 $e_i$ 的攻击(其中 $1 \le i \le l$),因而将日志系统的安全问题转换为每个日志记录体的安全问题. 每个日志体中不但包含了加密的日志信息,还拥有信息认证码(MACs). 即使在时间段 $\varepsilon_j$(其中 $j = 1, \cdots, n$)的密钥 $k_j$ 被攻破,所有 $k_i$(其中 $i < j$)也不会被泄漏. 从而保证了日志记录体的完整性不被破坏.

但此机制也有其无法避免的安全缺陷:首先它无法保护日志系统的进程不被恶意更改或删除;其次,也无法禁止对日志文件的恶意删除;另外,此机制需要一个远端可信系统来存储 FI 的基本密钥 $k_0$,以保护 $k_i$(其中 $i > 0$)不被泄漏以及对每个日志体的验证,然而可信系统本身就包含了不可信的因素.

**(2) S&K 保护机制**　这种由 Schneier 和 Kelsey 提出的机制与 FI 机制有许多相似之处[9]. 它们都将日志生成的时间分成连续的时间段 $\varepsilon_j$(其中 $j = 1, \cdots, n$),对于每个 $\varepsilon_j$ 都有唯一的认证密钥 $k_j$ 与其对应. 而且也使用了与 FI 保护机制相似的方法来对日志信息记录体进行校验. 然而其安全性在以下几方面有所改进:(a) S&K 保护机制中的密钥 $K_j$ 的生成不像在 FI 保护机制中那样直接生成,而是需要其它域的参与($K_j = H(W_j, A_j)$);(b) $Y_j$ 是用于验证第 $j$ 个日志记录体完整性的校验值. 由于 $Y_j = H(Y_{j-1}, E_{K_j}(D_j), W_j)$ 使得校验者看不到日志记录明文的情况下也可对其进行完整性验证. 因此即便校验者只是一位半可信主体(semi-trusted party),也可对日志记录进行校验;(c) 在 S&K 保护机制中,$Y_j$ 域提供了相对于 FI 保护机制更强的序列号结构,能更好的检测对日志记录体的删除和插入等操作.

尽管 S&K 相对于 FI 在安全性上有所加强,但是此机制也有其固有的弱点:首先,此机制虽然可以检验序列中是否有日志记录体被恶意删除,但对于整个日志文件的恶意删除还是无法保护. 其次,无法防止对于日志系统进程的恶意操纵. 最后,此机制也需要一个远端可信系统来存储它的基本验证密钥,这本身也包含了不可信的因素.

**(3) NIGELOG 保护机制**　NIGELOG 主要通过隐藏文件的方法来对日志信息实施保护[10]. 此机制将日志信息多重备份,并将它们隐藏在随机选定的目录中. 并且当原始的日志信息被更改或删除时,NIGELOG 可以自动通过备份文件将原始信息恢复.

此方法虽较好的保护了日志文件,使其不会轻易丢失. 但无法防止对日志系统进程的恶意修改以及恶意删除.

**(4) 访问控制保护机制**　使用访问控制来限制一般用户对日志系统的访问[12]. 然而,文[12]需要操作系统提供大量特殊的附加功能,与原系统的兼容性很差,并不适合一般情况.

**(5) Syslog-Sign 保护机制**　前面几种方法集中于在当前主机中保护日志信息的安全,而此方法则侧重于将日志信息传输到另一台可信主机中来保障日志信息的安全[11]. 此方法还提供了一些原始信息认证、信息完整性保护等功能,以防止信息在传递过程中被删除或更改. 但此方法对于 Denial of Service 等攻击方法却无能为力,而且也无法保证日志系统的进程不被恶意操纵.

## 3　FLOMAC 机制对 EDECS 的保护

### 3.1　安全隔离环境的提出

通过对上一节的分析,我们可以看到日志系统本身就可以包含自我保护的机制,并且此机制作为日志系统的一部分与它一起运行. 然而此类机制无法对整个日志系统实施保护. 对于 EDECS 也面临着同样的问题,即无法对运行于客户端的电子数据证据收集进程实施保护.

由此,我们提出了对 EDECS 实施保护的主要思想:提供一个安全隔离环境,使得 EDECS 运行于其中. 也就是说,EDECS 将运行于一个隔离的环境中,其它进程无法对此环境进行访问,从而也就无法对 EDECS 的进程和文件进行访问.

我们使用强制访问控制机制来构造安全隔离环境. 由于

在 UNIX 等系统上超级用户有权访问任何文件和进程,因此保护机制不能在应用层上构造;这样,在内核层实现强制访问控制机制,从而构造一个运行 EDECS 的安全隔离环境是必需的.

那么选择何种强制访问控制机制是我们此时要考虑的主要问题. 尽管可信计算基(TCB)对日志系统的保护由来已久[13],然而很多强制访问控制机制对于一般环境,尤其是商业现货(COTS)环境是不适用的[8]. 因此,我们采用了适于 COTS 环境的 LOMAC[14]作为 EDECS 的保护机制,使得 EDECS 在一个安全隔离环境中运行.

### 3.2 LOMAC 对 EDECS 保护的缺陷

LOMAC 是一种在内核层的、基于低水印访问控制(Low Water-Mark access control)的完整性保护机制. 在 COTS 环境中与现存软件、配置和应用高度兼容[15]. LOMAC 提供完整性保护的本质是防止潜在的有害数据从低完整性的客体流向高完整性的客体,从而威胁高完整性客体的完整性. LOMAC 定义了主体、客体和它们的安全级别. 主体的完整性级别将依据它访问的客体的完整性级别进行动态的调整,以此限制主体行为,提供完整性保护[14,15].

虽然 LOMAC 在 COTS 环境中提供了有效地完整性保护,然而它在对 EDECS 实施保护的过程中仍存在很大的问题:EDECS 客户端的证据收集进程无法和其它进程隔离. 也就是说,此进程无法运行于安全隔离环境中. 这是因为 EDECS 客户端的证据收集进程和其它进程可能都会访问一些公共对象,这些对象包括系统中的易丢失数据、描述系统运行状态的数据、网络数据、第三方的日志文件、库文件和 IPC 等. 如果这些对象存在于与 EDECS 相同的高安全级别,除 EDECS 外的任何其它进程都无法对其进行访问. 然而如果将这些对象设为低安全级别,EDECS 客户端的证据收集进程将无法隔离于其它进程,也无法维持其高安全级别. 这样,EDECS 客户端的证据收集进程既不能保护其自身的安全也无法向存在于高安全级别的 EDECS 中录入取证信息.

### 3.3 FLOMAC 对 EDECS 的保护

为了解决 LOMAC 在 EDECS 保护中存在的问题,本文对 LOMAC 的访问决策进行了调整,提出一个基于 LOMAC 的新的保护机制 FLOMAC.

**3.3.1 FLOMAC 的访问决策** 为了便于对 FLOMAC 的访问决策进行描述,我们首先将 LOMAC 中的部分定义及公理描述如下:

S:主体的全集;

O:客体的全集;

I:完整性级别的全集;

$il$:函数 $S \cup O \rightarrow I$,用于定义与主体和客体相关的完整性级别;

$\underline{leq}$:一种可逆的、对称的、可传递的关系($I \times I$ 的子集),此关系定义了有关 $I$ 的小于等于的关系;

$\underline{min}$:函数 POWERSET($I$) $\rightarrow I$ 返回指定子集 $I$ 的最大下界;

$\underline{o}$:一种主体观察(observed)客体的关系($S \times O$ 的子集),如果主体 $s \in S$ 观察客体 $o \in O$,有 $s \underline{o} o$;

$\underline{m}$:主体更改(modified)客体的关系($S \times O$ 的子集),如果主体 $s \in S$ 更改客体 $o \in O$,有 $s \underline{m} o$;

$\underline{i}$:主体调用(invoked)客体的关系($S \times S$ 的子集),如果主体 $s_i \in S$ 调用另一主体 $s_2 \in S$,有 $s_1 \underline{i} s_2$;

**公理 1:** $\forall s \in S, o \in O, s \underline{o} o \Rightarrow \underline{il}'(s) = \underline{\min}\{\underline{il}(s), \underline{il}(o)\}$

**公理 2:** $\forall s \in S, o \in O, s \underline{m} o \Rightarrow \underline{il}(o) \underline{leq} \underline{il}(s)$

**公理 3:** $\forall s_1, s_2 \in S, s_1 \underline{i} s_2 \Rightarrow \underline{il}(s_2) \underline{leq} \underline{il}(s_1)$

在 FLOMAC 中,我们保留 LOMAC 中的主体和客体的概念. 我们将和的概念应用于读操作(read),将的概念应用于写操作(write). 每一次主体对客体的访问操作都将由 FLOMAC 来裁决. 首先我们对 LOMAC 的定义进行了扩展:

$M$:访问模式的全集(访问模式包含 READONLY,WRITE,APPEND 等,详见 4.3 节);

$W$:访问职能的全集(访问职能包含证据识别职能、证据提取职能等,详见 4.3 节);

$\underline{il}'$:由 $\underline{il}$ 扩展的安全级别,$\underline{il}' = (\underline{il}, m, w)$,其中 $m \in M$ $w \in W$;在 FLOMAC 中,$\underline{il}$ 称为安全等级,$\underline{il}$ 称为安全级别;

$m_s, w_s$:分别为主体的访问模式和访问职能;

$m_o, w_o$:分别为客体的访问模式和访问职能.

基于 LOMAC,我们建立了 FLOMAC 的访问规则:

**规则 1:**读访问规则(IRAR)

IF $\underline{il}(s) \underline{leq} \underline{il}(o)$ AND $w_s \supseteq w_o$ THEN return (YES)

ELSE IF $\underline{il}(o) \underline{leq} \underline{il}(s)$ AND $w_s \supseteq w_o$ THEN $\underline{il}(s) = \underline{min}(\underline{il}(s), \underline{il}(o))$,return (YES)

ELSE return(NO);

**规则 2:**写访问规则(IWAR)

IF $\underline{il}(o) \underline{leq} \underline{il}(s)$ AND $w_s \supseteq w_o$ THEN return (YES)

ELSE IF $\underline{il}(s) \underline{leq} \underline{il}(o)$ AND $w_s \supseteq w_o$ AND $m_s = m_o$ THEN return (YES)

ELSE return (NO);

**规则 3:**读写访问规则(IRWAR)

IF $\forall \underline{il}(s) \underline{leq} \underline{il}(o)$ AND $w_s \supseteq w_o$ THEN call IWAR

ELSE IF $\underline{il}(o) \underline{leq} \underline{il}(s)$ AND $w_s \supseteq w_o$ THEN $\underline{il}(s) = \underline{min}(\underline{il}(s), \underline{il}(o))$,call IWAR

ELSE return (NO);

以上规则中,"return (YES)"表示访问操作被允许,"return (NO)"表示访问操作被拒绝.

**3.3.2 FLOMAC 访问决策的说明** 我们用两个例子对以上规则加以说明. 第一个例子中,将主体 $S_1$ 的安全级别表示为 $\underline{il}'_{s_1}$ = (LOW_LEVEL,WRITE,Fb),客体 $O_1$ 的安全级别表示为 $\underline{il}'_{o_1}$ = (HIGH_LEVEL,WRITE,Fb),其中 LOW_LEVEL < HIGH_LEVEL,WRITE 为访问模式中的"写模式",Fb 为访问职能中的"证据提取职能"(详见 4.3 节). 如果 $S_1$ 对 $O_1$ 进行写访问,由 IWAR 规则可知,尽管 $\underline{il}(s_1) \underline{leq} \underline{il}(o_1)$,但由于 $m_{s_1} = m_{o_1}$ 且 $w_{s_1} \supseteq w_{o_1}$,使得 $S_1$ 成功的对 $O_1$ 实施写操作. 也就是说,如果 $S_1$ 的安全等级低于 $O_1$,$O_1$ 只允许任何具有"写模式",且具有 Fb 访问职能的主体 $S_1$ 对其执行写操作,并且只能进行写操作.

第二个例子中主体 $S_2$ 的安全级别表示为 $\lambda'_{s_2}$ = (HIGH_LEVEL,READOLNLY,Fa),客体 $O_2$ 的安全级别表示为 $\lambda'_{o_2}$ = (LOW_LEVEL,READOLNLY,Fa),其中 LOW_LEVEL < HIGH_LEVEL,READOLNLY 为访问模式中的"只读模式",Fa 为访问职能中的"证据识别职能"(详见 4.3 节).如果 $S_2$ 对 $O_2$ 进行只读访问,由 IRAR 可得,由于 $w_{s_2} \supseteq w_{o_2}$,此次访问成功.也就是说,$O_2$ 只允许任何具有"只读模式",且具有 Fa 访问职能的主体对其执行只读操作,并且只能进行只读操作.又因为 $\lambda(o_2)$ leq $\lambda(s_2)$,所以 $\lambda(s_2) = \min(\lambda(s_2), \lambda(o_2))$.

FLOMAC 使得 EDECS 客户端的证据收集进程和其它进程都会访问的一些公共对象存在于低安全等级.这样,对其进行访问的证据收集进程也会降为低安全等级.但由以上描述可知,即使证据收集进程存在于低安全等级,在访问职能和访问模式的限定下,FLOMAC 不但可以充分的保证证据收集进程的正常运行,而且可向存在于高安全级别的 EDECS 中录入取证信息.这样,不但保证了 EDECS 运行于一个安全隔离环境中,而且有效地解决了 3.2 节中提到的问题.

## 4　FLOMAC 的实现

为了验证 FLOMAC 对 EDECS 保护的实际效果,我们在 Linux 平台上构建了 FLOMAC 的实现原型,作为全文研究的实验基础,是全文主要思想的应用实践.此原型依据一个被扩展的 ISO 访问控制框架(ISO Access Control Framework)构建[17],我们称之为 EDECS_RFSA.EDECS_RFSA 选用 ISO 访问控制框架作为基本框架的原因是它可灵活的支持多种访问控制策略.这样,就提供了一个灵活的、可扩展的 EDECS 保护机制之实验平台.在这一节,我们将解释 EDECS_RFSA 中的概念和术语,并对其实现进行具体描述.

### 4.1　整体框架

EDECS_RFSA 主要包括安全属性信息模块(SPIM)、安全策略裁决模块(SPDM)和安全策略执行模块(SPEM).SPIM 中记录了系统中所有主体和客体的安全属性信息.SPDM 依据安全策略和安全属性信息判断进程的请求是否符合安全策略,并做出裁决.SPEM 按照 SPDM 所做出的判断来执行系统调用.

现以打开文件为例,说明 EDECS_RFSA 如何处理安全相关的系统调用以及各模块的交互关系.首先 SPEM 向 SPDM 送出"open"请求,这个请求的参数包含访问模式(access mode),调用进程(calling process)和被访问文件的身份标识(ID).通过使用与"open"请求相关的策略规则以及此策略规则所需的安全属性信息,SPDM 对进程的请求做出裁决.之后,SPEM 执行

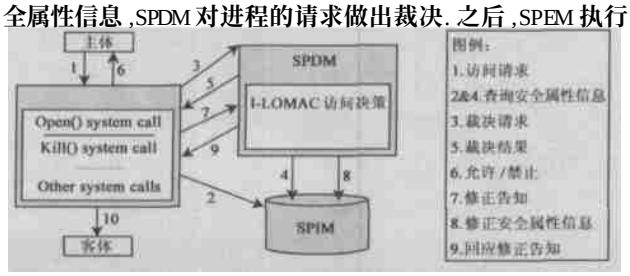


图 1　EDECS_RFSA 中各模块的交互关系

SPDM 的裁决,或者继续执行"open"的系统调用,或者调用进程返回错误信息.如果"open"请求被允许,且打开文件的操作正确执行,SPDM 将对相关的安全属性进行重新修正.图 1 显示了各模块间的交互关系.

### 4.2　SPEM 模块描述

因为在 Linux 中所有安全相关的访问都需通过系统调用,因此 SPEM 对安全相关的系统调用进行了"包装(Wrapper)",类似之"包装"功能的详细论述可见文献[18].对于每个安全相关的系统调用都会有一个"包装"与其对应,每个"包装"具有与其对应的系统调用相同的参数.起初,SPEM 将拦截安全相关的系统调用向量,并将安全相关的系统调用的地址替换为相应的"包装"的地址.这样,通过系统调用向量所生成的调用将直接调用"包装",而非内核中相应的原始系统调用.最后,"包装"向 SPDM 发送裁决请求,以判断允许或拒绝调用进程的请求.当 SPDM 的裁决结果为禁止时,SPEM 将给调用进程返回错误信息.否则,SPEM 将调用内核的原始系统调用以提供实际所需的服务.表 1 列出 Linux 相关系统调用与相应的由"包装"发出的裁决请求之对照表.

表 1　Linux 相关系统调用与"包装"发出的裁决请求之对照表

| "包装"发出的裁决请求 | Linux 相关系统调用 |
|---|---|
| ALTER | ipc,msgctl,shmctl |
| APPEND_OPEN | open,msgsnd |
| CHANGE_GROUP | chgrp,fchggrp,setgid,setfsgid,setregid,setgroups |
| CHANGE_OWNER | chown,fchown,setuid,setfsuid,setreuid |
| CREATE | creat,ipc,socketcall,mkdir,mknod,symlink,open,msgget,shmget |
| DELETE | ipc,socketcall,rmdir,unlink,msgctl |
| EXECUTE | exev_ve |
| LINK_HARD | link |
| MODIFY_ACCESS_DATA | utime |
| MODIFY_PERMISSIONS_DATA | chmod,fchmod,ioperm,iopl |
| MOUNT | mount |
| READ | readdir,readlink,getdent |
| READ_OPEN | ipc,open,msgrcv,shmat |
| READ_WRITE_OPEN | ipc,socketcall,open,shmat |
| RENAME | rename |
| SEARCH | chdir,fchdir |
| SEND_SIGNAL | kill |
| TRACE | ptrace |
| TRUNCATE | open,truncate,ftruncate |
| UMOUNT | umount |
| WRITE | rename |
| WRITE_OPEN | open |

### 4.3　SPIM 模块描述

SPIM 模块包含进程、用户以及所有资源的安全属性信息.它是安全策略所依赖的并且受安全策略控制.对 SPIM 的访问仅限于预先定义的函数调用.用户和文件等的安全属性信息的存储格式独立于所使用的文件系统.

表 2、表 3 列出了 SPIM 中所需的安全属性信息以及它们所代表的含义.其中的访问模式和访问职能是系统给出的默

认值,在不同的应用环境中,其含义和数量可以作相应的修改和扩充.

表 2 SPIM 中的安全属性信息

| 属性的名称 | 属性的含义 |
|---|---|
| id | 主体和客体的身份标识; |
| integrity_level | 主体和客体的完整性等级; |
| ilomac_*m* | 主体和客体的访问模式,指明主体可以通过何种访问模式来访问客体.我们定义了 8 种访问模式:READONLY,WRITE,APPEND,CREAT,DELETE,LINK,MODIFY,EXECUTE;它们分别代表了 readonly,write,append,creat,delete,link,modify,execute 的访问操作. |
| ilomac_*w* | 主体和客体的访问职能,指明主体访问客体时各自所具有的访问职能.我们定义了 6 种取证职能,详细描述可见表 3. |

表 3 取证访问职能描述

| 访问职能的名称 | 含义 |
|---|---|
| 证据识别职能(*Fa*) | 生成可作为证据的系统信息,即证据收集点的信息.当然也包括如防火墙等第三方所提供的证据信息; |
| 证据提取职能(*Fb*) | 从主机中的证据收集点提取证据信息; |
| 证据收集职能(*Fc*) | 从多台具有证据提取功能的主机中收集证据信息,并汇总; |
| 证据检查职能(*Fd*) | 对收集的证据进行有效性验证,并对验证后的数据进行过滤,以提取与证据直接相关的信息; |
| 证据分析职能(*Fe*) | 对过滤后的数据进行分类、比较和定位,实现犯罪过程重构. |
| 证据提交职能(*Ff*) | 接收最终电子数据证据的分析结果,并以可向法院提交的格式形成报表. |

表 4 裁决请求对应的裁决规则

| 客体 / 裁决请求 | FILE | DIR | DEV | PROCESS | IPC |
|---|---|---|---|---|---|
| ALTER | | | | | A |
| APPEND_OPEN | A | | A | | A |
| CHANGE_GROUP | A | A | | | A |
| CHANGE_OWNER | A | A | | A | A |
| CREATE | | A | | | |
| DELETE | A | A | | | A |
| EXECUTE | R+S | | | | |
| LINK_HARD | A | | | | |
| MODIFY_ACCESS_DATA | A | A | | | |
| MODIFY_PERMISSIONS_DATA | A | A | | | A |
| MOUNT | | W+S | W+S | | |
| READ | | R+S | | | |
| READ_OPEN | R+S | R+S | R+S | | R+S |
| READ_WRITE_OPEN | W+S | | W+S | | W+S |
| RENAME | A | A | | | |
| SEARCH | | R+S | | | |
| SEND_SIGNAL | | | | A | |
| TRACE | | | | W+S | |
| TRUNCATE | A | | | | |
| WRITE | | A | | | |
| WRITE_OPEN | A | | A | | A |

### 4.4 SPDM 模块描述

SPDM 从 SPEM 收到裁决请求,SPDM 将依据在 3.2.1 中所讨论的 FLOMAC 访问决策做出裁决.表 4 详细说明了对于所有由"包装"发出的裁决请求 FLOMAC 所对应的裁决规则.每行代表一个裁决请求,每列指明了主体需访问的目标.其中,"A","R","W"分别代表主体访问客体时需要遵循的 FWAR,IRAR,IRWAR 访问规则."S"表示一次成功的访问之后,主体相关的安全属性将被重新设置.

## 5 FLOMAC 实现的评价

这一节展示了对 FLOMAC 保护机制的评价结果,以及对其优缺点的讨论.我们对 FLOMAC 进行了保护功能和性能的测试,测试的详细结果如下.

### 5.1 保护功能测试

我们分别对五个含有 EDECS 的系统进行了攻击性测试,并对攻击结果进行了比较,以便正确评价 FLOMAC 的保护功能.这五个系统分别是:红旗 Linux 3.0,运行 FI 保护机制(2.2.3.1 节)的红旗 Linux 3.0(红旗 Linux 3.0 + FI),运行 S&K 保护机制(2.2.3.2 节)的红旗 Linux 3.0(红旗 Linux 3.0 + S&K),运行 NIGELOG 保护机制(2.2.3.3 节)的红旗 Linux 3.0(红旗 Linux 3.0 + NIGELOG)和运行 FLOMAC 的红旗 Linux 3.0(红旗 Linux 3.0 + FLOMAC).在实验中,我们使用五种攻击方法对这三个系统分别进行攻击,并测试这三个系统中的保护机制是否有效.使用的攻击方法如下:

(1)攻击方法 1:停止 EDECS 的进程

对 setuid 程序实施缓冲区溢出的攻击,并获得特权.然后用 kill 命令停止所有的 EDECS 进程.

(2)攻击方法 2:删除 EDECS 保存证据的文件

对 setuid 程序实施缓冲区溢出的攻击,并获得特权.然后删除所有 EDECS 用于临时存放取证信息的文件.

(3)攻击方法 3:伪造 EDECS 保存证据的文件

对 setuid 程序实施缓冲区溢出的攻击,并获得特权.然后篡改伪造 EDECS 的取证信息文件.

(4)攻击方法 4:删除 EDECS 的配置文件

对 setuid 程序实施缓冲区溢出的攻击,并获得特权.更改配置文件的访问权限,并删除 EDECS 的配置文件.

(5)攻击方法 5:查看取证信息

对 setuid 程序实施缓冲区溢出的攻击,并获得特权.查看客户端主机上已被收集的取证信息.

表 5 攻击测试结果对比

| | 红旗 Linux 3.0 | 红旗 Linux 3.0 + FI | 红旗 Linux 3.0 + S&K | 红旗 Linux 3.0 + NGELOG | 红旗 Linux 3.0 + FLOMAC |
|---|---|---|---|---|---|
| 攻击方法 1 | N | N | N | N | Y |
| 攻击方法 2 | N | N | N | Y | Y |
| 攻击方法 3 | N | Y | Y | Y | Y |
| 攻击方法 4 | N | N | N | Y | Y |
| 攻击方法 5 | N | Y | Y | N | Y |

("Y"代表可以抵御攻击;"N"代表无法抵御)

攻击测试结果如表 5 所示.表中“Y”代表可以抵御攻击,而“N”代表无法抵御攻击.由此可得:

(1) FLOMAC 对 EDECS 的进程实施了保护.因此攻击者无法停止或替换取证进程,保证了取证数据的真实性(Fidelity).

(2) 对 EDECS 的文件(取证信息文件和系统配置文件)实施了保护,攻击者无法阅读、篡改甚至删除取证数据,保证了数据的完整性(Integrity).

文[5]中提到衡量取证数据可信度的两个标准是:真实性和完整性.因此,FLOMAC 很好的实现了对 EDECS 安全隔离,使得生成的取证数据有较高的可信度.

### 5.2　性能测试

FLOMAC 运行时,系统性能将会下降.这是由于每个安全相关的系统调用都会触发 FLOMAC.为了检验 FLOMAC 对系统的影响程度,我们通过实验的方法进行测算.测试硬件实验平台为普通微机,CPU 为 Intel Pentium Ⅲ 667MHz,内存 256M,硬盘 20G.原 Linux 内核版本为 2.4.18.测试分为两部分:处理服务请求的时间开销测试和资源访问的时间开销测试.实验时,各安全机制处于工作状态.

**5.2.1　处理服务请求的时间开销测试**　FLOMAC 通过截获安全相关的服务请求的方式来进行安全控制.服务请求来自系统调用.Open 系统调用是最典型、使用最频繁的与安全有关的系统调用,我们以该调用为代表设计测算方案.

设计一个 time_evaluate 程序,它的主要功能是试图用 open 系统调用打开一个指定的文件并测出 open 产生的服务请求被处理的时间.打开文件前,open 首先发出访问文件的请求,请求得到允许后,才能打开文件.time_evaluate 程序记录请求处理的时间开销,它还能以指定的次数循环执行以上的操作,程序结束前,输出所记录的时间开销,精确到微秒.如果文件打开成功,还要关闭该文件.试验时,以 2 万次作为循环次数,连续运行 10 次,可得到一个请求被处理 20 万次的时间开销,以算术平均作为对一个请求处理一次的时间开销.

open 系统调用打开文件的过程中需要对文件路径名进行遍历,历经路径名的每个分量时都要判断主体对相应分量所对应的目录客体的访问权限,实际上该系统调用所提交的一个服务请求可能因起多个访问许可判断,这由路径名长度确定.实验时,作者对多达 20 级分量的多种子目录深度的路径进行测试,在普通 Linux(红旗 Linux 3.0)和普通 Linux + FLOMAC 的环境中分别进行同样内容的实验,实验数据如表 6 所示.

表 6　处理服务请求的时间开销

| 文件路径名深度 < 层数 > | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| 普通 Linux 的时间开销(微秒) | 4.8 | 5.9 | 7 | 7.9 | 9.6 | 11 | 12 | 13.9 | 15 | 15.8 |
| 普通 Linux + FLOMAC 的时间开销(微秒) | 5.7 | 7.1 | 8.5 | 9.7 | 12 | 13.9 | 15.4 | 18 | 19.6 | 20.7 |
| 性能损失(%) | 18.75 | 20.33 | 21.42 | 22.78 | 25.00 | 26.36 | 28.33 | 29.50 | 30.67 | 31.01 |

(每项开销是 20 万次请求的平均值)

表 7　资源访问的时间开销

| 文件路径名深度 < 层数 > | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| 普通 Linux 的时间开销(微秒) | 30360 | 31103 | 31274 | 31291 | 31458 | 31480 | 30719 | 30354 | 31296 | 30495 |
| 普通 Linux + FLOMAC 的时间开销(微秒) | 32523 | 32575 | 32621 | 32677 | 32790 | 32855 | 32957 | 33088 | 33407 | 33647 |
| 性能损失(%) | 7.12 | 4.73 | 4.30 | 4.42 | 4.23 | 4.37 | 7.29 | 9.00 | 6.75 | 10.34 |

(每项开销是 5 万次请求的平均值)

由表 6 可知,FLOMAC 对系统的性能影响较小,即便在目录层次大量增加的情况下,FLOMAC 对系统性能损失的增长较为缓慢.

**5.2.2　资源访问的时间开销测试**　服务请求的目的是访问资源,为了进一步测算 FLOMAC 对资源访问效率的影响,以最典型的资源访问操作——读文件为代表设计测算方案.

设计一个 read_evaluate 程序,该程序与 time_evaluate 程序相似,只是增加了一项功能,在打开文件操作成功之后,把相应的文件内容读入内存中,并测出读取文件的时间开销.read_evaluate 程序输出结果为资源访问的时间开销(由服务请求处理开销和实际的资源存取开销组成).实验时,选择一个大小约为 1M 字节的文件为访问对象,以 5 千次作为循环次数,连续运行 10 次,可得到对一个文件进行 5 万次访问的时间开销,以算数平均作为对该文件进行一次访问的时间开销.实验结果如表 7 所示.

实验结果表明,资源访问的时间开销总体上是随着路径名的深度增加的,其间有些波动,这显示出系统其它方面的影响(如文件在磁盘文件系统中的存放方式和系统调用方法等)会比安全相关服务请求及处理开销的增长具有更大的权重.经过估算,FLOMAC 引起的系统性能下降大致在 4%~11%的范围内.通常,系统中路径名深度超过 10 层的并不多;深度在 10 层以内的性能影响的算术平均值约为 4.96%;这样,读取一个约为 1M 的文件所增加的开销在千微秒的数量级内,这说明了 FLOMAC 中由于实施访问控制而引起的时间消耗在整个资源访问过程中所占比例很小.

### 5.3　兼容性分析

Linux 中的基本访问控制机制是基于拥有者、属组和其它人的自主访问控制(DAC),新的安全机制的引入不会与原有机制冲突.当 FLOMAC 的安全机制处于非工作状态时,系统的安全职能与原系统相同.当处于工作状态时,FLOMAC 与原有粗粒度 DAC 机制协同作用,FLOMAC 条件首先被判断.

## 6　总结

EDECS 的保护是电子数据取证技术中要解决的一个关键问题,本文通过对几种典型保护机制的分析,提出安全隔离环境是用于保护 EDECS 的有效方法.为了构造安全隔离环境,作者将访问控制机制引入了 EDECS 的保护功能中:以 LOMAC 作为基础,设计并实现了一种新的保护机制——FLOMAC.与 LOMAC 相同,它同样适合于在 COTS 环境中应用.通过实验测算,我们证实这种新的访问控制机制不但为 EDECS 提供了较为完备的安全隔离环境,而且对整个系统性能影响较小.

电子数据取证是一项正在发展的技术[5],对 EDECS 的保护之研究更是刚刚起步[6]. 本文从访问控制的角度讨论了 Linux 环境下对 EDECS 的保护,力图为今后取证系统保护机制的研究工作提供理论及实践的依据. 如何进一步形式化分析和证明电子数据证据在转换及传输过程中的完整性问题,是我们下一步要研究的主要问题.

## 参考文献:


[1] R D Hof. A New Era of Bright Hopes and Terrible Fears[R]. Business Week, Oct. 1999. 50 - 56.

[2] Brown Stallard. Automated Analysis for Digital Forensic Science[D]. USA: Univ. of California, Dec. 2002.

[3] Ahmed Patel. The impact of forensic computing on telecommunications [J]. IEEE Communications Magazine, 2000, 11: 64 - 67.

[4] Jesse Kornblum. Preservation of Fragile Digital Evidence by First Responders[R]. Digital Forensics Research Workshop, August 2002.

[5] Gary Palmer. A Road Map for Digital Forensic Research[R]. Digital Forensics Research Workshop, August 2002.

[6] John Tan. Forensic Readiness[Z]. http://www.atstake.com, July 2001.

[7] S Haber, W S Stornetta. How to time stamp a digital document[J]. Advances in Cryptology-Crypto '90, Springer-Verlag: 1991, 437 - 455.

[8] M Bellare, B S Yee. Forward Integrity For Secure Audit Logs[R]. 1997 University of California, San Diego: 1997.

[9] B Schneier, J Kellsey. Secure audit logs to support computer forensics [J]. ACM Transaction on Information and System Security, May 1999, 2 (2): 159 - 176.

[10] T Takada, H Koike. NIGELOG: Protecting logging information by hiding multiple backups in directories[A]. International Workshop on Electronic Commerce and Security (in conjunction with DEXA '99)[C]. IEEE CS Press, Sep. 1999. 874 - 878.

[11] J Kelsey, J Callas. Syslog-Sign Protocol DRAFT[R]. Network Working Group, June 2002.

[12] Hewlett-Packard Company. HP Praesidium/VirtualVault White Paper [R]. 1998, http://www.hp.com/security/products.

[13] U S Department of Defense, Computer Security Center. Trusted computer system Evaluation criteria[S]. December 1985.

[14] Tim Fraser. LOMAC: Low water-mark mandatory access control for linux[A]. The 8th USENIX Security Symposium[C]. Washington D. C., August, 1999.

[15] Tim Fraser. LOMAC: low water-mark integrity protection for COTS environments[A]. Proceedir 2000 IEEE Symposium on Security and Privacy[C]. Berkely, California, May 2000. 230 - 245.

[16] K J Biba. Integrity Considerations for Secure Computer Systems[R]. Technical Report ESD-TR-76-372, USAF Electronic Systems Division, Hanscom Air Force Base, Bedford, Massachusetts, April 1977.

[17] International Organization for Standardization (ISO). Information Technology-Open Systems Interconnection-Security Frameworks for Open Systems-Part 3: Access Control[R]. ISO/IEC 10181-3, 1995.

[18] T Fraser, L Badger, M Feldman. Hardening COTS software with generic software wrappers[A]. Proceedings of the 1999 IEEE Symposium on Security and Privacy[C]. Berkeley, California: May 1999. 2 - 16.


## 作者简介:


孙 波 男,1975 年 3 月生于辽宁省,博士研究生,主要研究方向为信息安全和系统软件. Email: sunbo@sonata.iscas.ac.cn



孙玉芳 男,1947 年 2 月生于江苏省,研究员,博士生导师,主要研究方向为系统软件和中文信息处理.