基于密码隔离的恶意代码免疫模型*)

陈泽茂[1] 沈昌祥[2] 吴晓平[1]

(海军工程大学信息安全系 武汉 430033)[1] (海军计算技术研究所 北京 100841)[2]

摘 要 基于访问控制的恶意代码防御模型对恶意代码实行的是逻辑隔离,它可能被旁路且防御效果受限于访问控制策略的有效性。本文基于密码学原理建立了一个恶意代码免疫模型,以克服逻辑隔离的脆弱性;定义了代码植入规则、保护规则和执行规则,实现代码存储和执行的安全;证明了在系统初态安全且代码加、解密密钥安全的条件下,任何时刻恶意代码都不会被执行和传播。

关键词 安全模型,恶意代码,安全操作系统

A Crypto-based Immunization Model against Malicious Code

CHEN Ze-Mao[1] SHEN Chang-Xiang[2] WU Xiao-Ping[1]

(Naval University of Engineering, Wuhan 430033)[1] (Naval Institute of Computing Technology, Beijing 100841)[2]

Abstract Access control based malicious code defending model has vulnerabilities because it may be circumvented in certain circumstances. In addition, its effectiveness depends on that of the access control policy applied. A crypto-based security model against malicious code is proposed which defined secure operation rules for executable code planting, protection, and execution. It is proved that if the initial system state is secure and the crypto keys for code protection are secure also, the proposed model will be able to protect system from being infected and be able to prevent subjects from invoking malicious codes, thus to stop malicious code from spreading.

Keywords Security model, Malicious code defending, Secure operating system

1 引言

基于访问控制的恶意代码防御模型主要是通过限制主体的权限来控制恶意代码的传播范围,其实质是对恶意代码实行逻辑隔离,以防止其在不同安全域间任意扩散[1~4]。这种防御策略起作用的前提是可信计算基(Trusted Computing Base,TCB)[5]的引用验证机制不被旁路。而当今流行的操作系统为了获得较高的运行效率,又多采用大内核的结构,把文件系统、设备驱动等模块都包含在TCB的安全边界内(系统内核)。TCB安全边界的扩大给系统安全带来了更多威胁,因为安全边界内的任一程序出现安全漏洞都可能导致TCB的安全机制被旁路,从而使得恶意代码得以绕过系统的身份认证机制和访问控制机制对系统实施破坏。

本文研究一种完全不同于逻辑隔离的恶意代码防御模型。该模型不是建立在访问控制的基础上,而是基于密码学原理[6],通过对可执行代码的标识和加密,在此基础上定义若干安全操作规则对可信代码域和不可信代码域进行密码隔离,防止各种已知和未知的不可信代码流入可信代码域,从而达到恶意代码免疫之目的。

2 模型机理

加密是利用密码学的方法对信息进行变换,使未授权者不能识别和理解信息的真正含义,也无法伪造和篡改数据。任何一个加密系统都至少包括如下几个部分:

①明文空间:它是所有待加密的消息集合,记为 M。

②密文空间:它是所有加密后的消息集合,记为 C。

③密钥空间:它是所有密钥的集合,记为 K。$\forall k_i \in K, k_i = (k_e, k_d)$,其中 k_e 是加密密钥,k_d 是解密密钥。

④加密算法:它是一组由 M 到 C 的数学变换,记为 E。

⑤解密算法:它是一组由 C 到 M 的数学变换,记为 D。

对于每一对确定的密钥 (k_e, k_d),加密算法将确定一个具体的加密变换,解密算法将确定一个具体的解密变换,而且解密变换是加密变换的逆过程。在加密密钥 k_e 的控制下,加密算法把明文变换成密文:

$$C = E_{k_e}(M)$$

在解密密钥 k_d 的控制下,解密算法把密文变换成明文:

$$M = D_{k_d}(C) = D_{k_d}(E_{k_e}(M))$$

在现代密码学中,算法的安全性都是基于密钥的安全性,即只要保证解密密钥的安全,就可以确保明文信息的安全。因此,可以通过不同安全域间的密钥分割达到控制信息共享的目的,而通过控制信息共享就有可能实现恶意代码防御。下面描述基于密码隔离的恶意代码免疫模型的基本机理。

公理 信息只有在能够被正确解析的情况下才有价值。

不论是可以直接由操作系统装载执行的二进制代码,还是由特定解释程序解释执行的应用级代码,它们能够被正确执行的前提都是能被正确解析。目前常见的操作系统都没有对可执行代码进行加密保护,这导致了如下的安全问题:

①攻击者有可能利用操作系统的安全漏洞向系统中植入并执行恶意代码。

②通过解析可执行文件的格式,攻击者可以向合法程序

*)本文得到国家"八六三"高技术研究发展计划项目(2002AA144020)的资助。陈泽茂 讲师,博士;沈昌祥 工程院院士,博士生导师;吴晓平 教授,博士生导师。

注入恶意代码，并可以设法使用户在执行合法程序时首先触发恶意代码，从而导致该用户的权限被恶意代码窃取。

基于密码隔离的恶意代码免疫模型假设只有经过标识的已知代码才是安全的，该假设的正确性可以通过技术手段来保证。在这个假设的基础上，把系统中的所有可执行客体分为两类：一类是已知的并经过标识的可执行程序；除此之外的所有其他可执行客体都归为另一类。对前者进行加密，把它同其他未经标识的可执行客体用密码隔离的方法分割开来。对可信的可执行客体按照上述方法进行了标识和加密保护之后，为了确保它们能够被正确执行，需要在操作系统的程序装载器中增加代码解密模块，如图1所示。

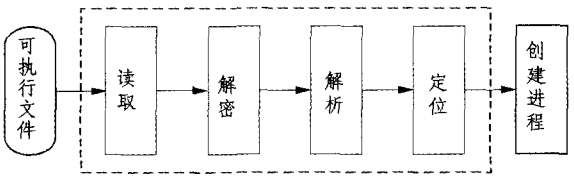

图1 加密程序的执行流程

在程序的执行流程中增加了解密环节之后，由于只有解密结果正确的代码才可能被操作系统执行，因此只要确保解密密钥的安全，就可以实现：

①防止合法程序被植入恶意代码；

②防止主体触发恶意代码。

3 模型描述

把所有客体构成的集合记为 O，把所有客体划分为数据(Data)和代码(Code)两类。无论是数据还是代码，都可以把它看作是一组字节按特定顺序排列而成的，即可以把它表示成向量形式。

定义1 $T=\{t_c, t_d\}$ 是客体类型的集合，其中 t_c 表示代码，t_d 表示数据。

定义2 $\forall o\in O$，$content(o)$ 表示 o 中存储的信息，令 $content(o)=\langle a_1, a_2, \cdots, a_n\rangle$，其中 a_i 表示单个字节，$i\in[1, n]$，n 为自然数。

定义3 $k_c=(k_e, k_d)$ 是代码保护密钥，其中 k_e 是加密密钥，k_d 是解密密钥。

定义客体的类型函数，规定在经过加密变换之后客体的类型保持不变，即一个可执行程序被加密之后，虽然其可执行格式信息已经不可见，但仍然要把它标识成代码类型。

定义4 $f_t: O\to T$，$\forall o\in O$，若 $o_e=E_{k_e}(o)$，则有 $f_t(o_e)=f_t(o)$，称 f_t 为客体的类型函数。

定义5 $P=\{o\mid f_t(o)=t_c\}$，令 $P=P_u\cup P_e$，其中 P_u 是 P 中未加密客体的集合，P_e 是 P 中已加密客体的集合。经过一次状态转换之后，这三个集合分别记为 P'、P'_u、P'_e。

规则1 $New(p):=$ if $p\notin P$, then $p_e=E_{k_e}(p)$，$p'_e=p_e\cup\{p_e\}$，$P'_u=P_u$，$P'=P\cup\{p_e\}$

规则2 $Protect(p):=$ if $p\in P_u$,

then $p_e=E_{k_e}(p)$，$P'_e=P_e\cup\{p_e\}$，$P'_u=P_u-\{p\}$，$P'=P'_u\cup P'_e$

规则3 $Execute(p):=\forall p\in P_u\cup P_e$，$p'=D_{k_d}(p)$，*execute* p'

规则1称为代码植入规则，该规则要求在向系统中引入新的可执行程序时必须先对该程序进行加密保护；规则2称为代码保护规则，该规则定义了如何对程序进行加密保护以及在该过程完成之后各可执行程序集的变化；规则3称为代码执行规则，该规则要求在执行任何代码之前都要先对其作无条件的解密变换。这里规则1和规则2是正确实施规则3所必需的。

4 安全定理

定义6 任何一个完整的程序或一个代码片段，如果它可被操作系统执行，则称其被正确植入系统。

定义7 p_1、p_2 是两个可执行程序，$p_1=p_2\Leftrightarrow content(p_1)=content(p_2)$。

定理1 $k_c=(k_e, k_d)$ 是代码保护密钥，若能确保 k_e 不被窃取或非法使用，则规则3可防止攻击者把恶意代码正确植入系统。

证明：假设 p 是一恶意代码，它可能已经被加密，也可能未被加密，下面分这两种情况证明：

(1) p 已被加过密

假设 p 是使用密钥 k_1 对可执行代码 p_1 进行加密后得到的，即 $p=E_{k_1}(p_1)$，根据规则3，p 的执行过程为：先无条件地用解密密钥 k_d 对其实施解密操作，解密后的结果为 $p'=D_{k_d}(p)$，然后把 p' 作为真正的执行对象。显然，当且仅当 $p'=p_1$ 时 p 才能被正确执行。但根据假设知攻击者不掌握代码加密密钥 k_e，据此有

$$k_1\neq k_e\Rightarrow E_{k_1}(p_1)\neq E_{k_e}(p_1)\Rightarrow p\neq E_{k_e}(p_1)\Rightarrow D_{k_d}(p)\neq D_{k_d}(E_{k_e}(p_1))\Rightarrow D_{k_d}(p)\neq p_1\Leftrightarrow p'\neq p_1$$

所以 p 不可能被正确执行，定理得证。

(2) p 未被加过 s 密

根据规则3，p 的执行过程为：先无条件地用解密密钥 k_d 对其实施解密操作，解密后的结果为 $p'=D_{k_d}(p)$，然后把 p' 作为真正的执行对象。显然，当且仅当 $p'=p$ 时 p 才能被正确执行，而 $p'=p\Leftrightarrow D_{k_d}(p)=p$，但显然 $D_{k_d}(p)\neq p$，故 p 不可能被正确执行，定理得证。

推论1 $k_c=(k_e, k_d)$ 是代码保护密钥，若能确保代码加密密钥 k_e 不被攻击者窃取或非法使用，则规则3可防止恶意代码被触发。

证明：由定理1，若能确保 k_e 不被攻击者窃取或非法使用，则规则3可确保恶意代码不可能被正确植入系统，再根据定义6中关于“代码正确植入”的定义知本命题结论成立。

推论2 $k_c=(k_e, k_d)$ 是代码保护密钥，若能确保代码加密密钥 k_e 不被攻击者窃取或非法使用，且第一次植入的代码都不包含恶意代码，则在规则3下，恶意代码不可能被触发。

证明：根据假设，代码加密密钥 k_e 不会被攻击者窃取或非法使用。由定理1知，规则3可确保在系统运行过程中恶意代码不可能被正确植入系统；又因为第一次植入的代码都不包含恶意代码，所以在任何时刻都不会有恶意代码被正确植入系统；再根据定义6中关于“代码正确植入”的定义知本命题结论成立。

下面证明本模型对计算机病毒的免疫作用。计算机病毒是一种依附在其他计算机程序中的具有自我复制能力的代码片段。计算机病毒要把自身传播给某个可执行程序，首先必须正确解析该文件的格式，然后才可能把病毒代码注入其中。

假设系统中的可执行代码已被加密保护，此时计算机病毒若要把自身传染给其他可执行程序，则其首先必须有能力正确解密出所要传染的程序。为此，它需要获得解密密钥，否

(下转第293页)

执行之前找出可能的问题，减少不必要的损失。一个活动的时序一致性直接依赖于活动的可调度性，一个 TCPNs 模型的时序一致性验证则依赖于模型的可调度性。这样时序一致性验证主要包含两方面的范畴：

对一个独立的活动 t_i 来讲，确保 $LFET(t_i)-EFBT(t_i)\geqslant FIRE_{dur}(t_i)$，则 t_i 是强可调度的。

·对整个 TCPN 模型来说，模型执行满足所有预设时间约束的前提条件且该模型中所有变迁在时间约束下都可调度。

定义 7 一个 TCPNs 模型中所有变迁基于附加时间约束都强可调度，则 TCPNs 模型一定能够顺利执行完成。

定理 6 一个 TCPN 模型是时序一致的当且仅当这个模型中所有活动基于现有时间约束都是可调度。

证明：在一个建模好的 TCPN 模型中，因为所有变迁基于附加时间约束都强可调度，因此对于每一个变迁 t_i，都有 $LFET(t_i)-EFBT(t_i)\geqslant FIRE_{dur}(t_i)$，这样每一个变迁也都存在决策域度。在事先期望的时间约束条件下，只要每个变迁在相应的可调度决策范围内开始执行，则整个 TCPN 模型一定能够顺利地完成执行工作。

总结与展望 本文的工作主要包括：

1. 扩展 TCPNs 的原始定义，对其变迁的强可调度性重新进行了定义，并给出单个变迁以及变迁序列的强可调度判定定理；

2. 变迁序列的组合分析策略以分析含有循环结构的复杂变迁序列；

3. 基于 TCPNs 可调度分析策略，提出时序一致性概念。

如何扩展组合分析策略来分析含有不规则结构的序列将是我们今后需要继续的工作。

致谢 在此，我们向对本文提出宝贵意见的评审专家表示感谢。

参 考 文 献

1 Tsai J J P, Yang S J, Chang Y H. Timing Constraint Petri Nets and Their Application to Schedulability Analysis of Real-Time System Specifications. IEEE Trans. Software Eng., 1995, 21(1): 32～49

2 Xu Dianxiang, He Xudong, Deng Yi. Compositional Schedulability Analysis of Real-Time Systems Using Time Petri Nets. IEEE Trans. Software Eng., 2002, 28(10): 984～995

3 Murata T. Petri nets: properties, analysis and application. Proc. IEEE, 1989, 77(8): 541～580

4 Van der Aalst W M P. The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems, and Computers, 1998, 8(1): 21～66

5 Adam NR, Atluri V, Huang WK. Modeling and Analysis of Workflow Using Petri Nets. Journal of Intelligent Information Systems, 1998, 10(2): 131～158

6 Ramamoorthy C V, Ho G S. Performance evaluation of asynchronous concurrent systems using Petri nets. IEEE Trans. Software Eng., 1980, SE(6): 440～449

7 Ciardo G, German R, Lindemann C. A characterization of the stochastic process underlying a stochastic Petri net. IEEE Trans. Software Eng., 1994, 20(7): 506～515

8 Merlin P M, Farber D J. Recoverability of communication protocols implications of a theoretical study. IEEE Trans. Comm., 1976, 24(4): 1036～1043

（上接第 289 页）

则便无法正确解析要传染的可执行程序，从而也就无法实施传染操作。也就是说，掌握解密密钥是计算机病毒实施传播的前提，据此有如下结论：

定理 2 $k_c=(k_e,k_d)$ 是代码保护密钥，若能确保代码解密密钥 k_d 不被窃取或非法使用，则计算机病毒不可能在系统中传播。

如果计算机病毒掌握了解密密钥或可以非法使用解密密钥，那么它就能正确解密受保护的代码并把病毒代码传染给它。即便如此，如果它不掌握加密密钥或不能非法使用加密密钥，就无法把被传染后的文件重新加密。这样，根据规则 3，并由定理 1 知计算机病毒代码仍然不可能被正确植入系统。这个结论由如下定理描述：

定理 3 $k_c=(k_e,k_d)$ 是代码保护密钥，若能确保代码加密密钥 k_e 不被窃取或非法使用，则在规则 3 下，计算机病毒不可能在系统中传播。

逻辑隔离的主要脆弱性是一旦引用验证机制被旁路，那么它将彻底或局部失去恶意代码防御能力。下面的定理指出了基于密码隔离的恶意代码防御策略较之逻辑隔离的优越性。

定理 4 $k_c=(k_e,k_d)$ 是代码保护密钥，假设在初始时刻系统中不存在恶意代码，那么如果能够确保代码加、解密密钥 k_e、k_d 不被窃取或非法使用，且规则 3 总是起作用，那么系统将具有如下恶意代码防御能力：

(1)在任意时刻可执行程序都不会被恶意代码传染；

(2)在任意时刻主体都不会触发恶意代码；

(3)在任意时刻恶意代码无法在系统中传播。

由前面所述的各个定理和推论知定理 4 的结论是显然的。定理 4 结论成立，要求规则 3 总是起作用，即要求实现规则 3 的安全机制不能被旁路。实现这一点较之确保逻辑隔离的有效性要容易得多，因为只要在操作系统的程序装载器中增加一个解密模块即可实现。

结论 目前安全操作系统所实现的恶意代码防御策略从本质上说都是基于访问控制策略的逻辑隔离，这类防御策略具有脆弱性：首先，TCB 的引用验证机制存在被旁路的可能；其次，所实施的访问控制策略自身可能存在局限性，从而限制了其恶意代码防御能力。本文基于密码隔离机制建立了一个恶意代码免疫模型，它可以克服逻辑隔离的脆弱性。该模型定义了代码植入规则、代码保护规则和代码执行规则，证明了在系统初态安全且代码加、解密密钥安全的条件下，任何时刻系统中都不会有恶意代码的执行和传播，从而达到对恶意代码免疫的防御效果。

参 考 文 献

1 Cohen F. Computer Viruses: Theory and Experiments. Computers and Security, 1987, 6(1): 22～35

2 Biba K J. Integrity Considerations for Secure Computer Systems. NTIS AD-A039324, Electronic Systems Division, Air Force Systems Command, April 1977

3 赵庆松. 安全操作系统的恶意代码防御技术的研究和实施:[学位论文]. 北京:中国科学院软件研究所, 2002

4 杨涛. SUNIX 安全操作系统:[学位论文]. 长沙:国防科技大学, 1993

5 Department of Defense of USA. Trusted Computer System Evaluation Criteria. DoD 5200.28 - STD, Aug 1983

6 Schneier B. 应用密码学 - 协议，算法与 C 源程序. 北京:机械工业出版社, 2000