文章编号:1001 - 9081(2005)11 - 2521 - 03

安全中间件——SSPM 管理模块的设计

曾令华[1],余 塑[2],欧阳开翠[1],周明天[2]
(1. 温州大学 计算机科学与工程学院,浙江 温州 325027;
2. 电子科技大学 计算机科学与工程学院,四川 成都 610054)
(zhl@wznc.zj.cn)

摘 要:公共安全服务管理器(CSSM)是安全中间件核心模块。讨论了它的体系结构及其安全服务的运行流程。设计了对安全服务提供者模块(SSPM)的管理模块,给出了使用的数据结构及算法。

关键词:安全中间件;公共安全服务管理器;安全服务提供者模块
中图分类号: TP393.08 **文献标识码**:A

Design of SSPM management module of security middleware

ZENG Ling-hua[1], SHE Kun[2], OUYANG Kai-cui[1], ZHOU Ming-tian[2]
(1. *School of Computer Science & Engineering, Wenzhou University, Wenzhou Zhejiang* 325027, *China*;
2. *School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 610054, *China*)

Abstract: The kernel module of security middleware is the Common Security Service Manager (CSSM). Its architecture and process of the security service were discussed. The management module of Security Services Provider Module (SSPM), part of CSSM, was designed and implemented. The data structure and the algorithm used in SSPM management module of CSSM, were also introduced.

Key words: security middleware; CSSM (Common Security Services Manager); SSPM (Security Service Provider Module)

0 引言

目前信息安全技术的基础部分已日趋成熟,但它的高代价、易用性差、互操作能力弱等问题成为信息安全问题的瓶颈。在现有的大多数应用系统中,安全部分的软件实现往往与整个系统密不可分,是一种紧耦合的关系,而将这二者分离开来,使它们成为一种松耦合的关系,可以使安全系统部分从整个应用系统中分离出来,成为共用的软件,使其具有可重用性、可移植性,达到缩短开发时间、降低开发成本、提高软件开发效率的目的。

可采用安全中间件技术来屏蔽安全的复杂性,实现复杂的网络安全问题与应用系统相分离。在安全中间件的管理下,实现各种安全服务如加密、解密、签名、验证、证书管理等的动态加载和卸载,以增强系统的灵活性和易用性。

Intel 公司推出的 CDSA 虽然还存在效率、通用性等方面的问题,但确实是一个结构比较完善的安全中间件[1,2]。

在国内,电子科技大学率先提出了安全中间件的概念,并设计了安全中间件的体系结构及公共安全服务管理器(Common Security Service Manager, CSSM)[3]。本文在此基础上,进一步探讨用于对安全服务提供者模块 SSPM(Security Service Provider Module)实施管理的模块的设计与实现。

1 安全中间件的构成

安全中间件不是一个简单的概念,而是实现安全策略、实现安全服务的基础架构。它由 6 大部分、5 个层次和 4 级接口组成[3]。6 大部分别是:应用程序层、组件服务层、安全服务层、公共安全服务管理器、安全服务提供者和资源信息服务器。其中除资源信息管理服务器外,其他 5 个部分构成了分层次的结构。如图 1 所示。

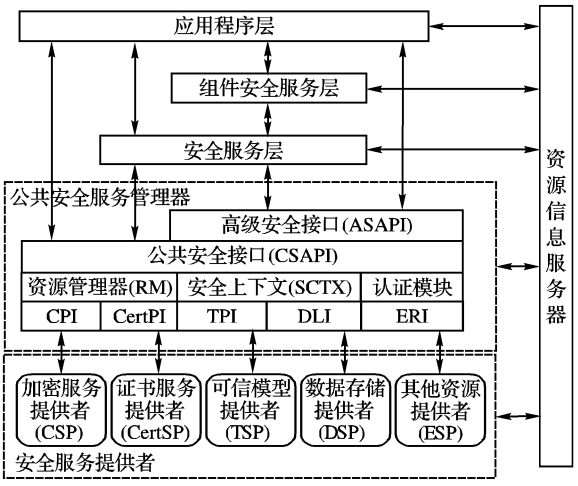

图 1 安全中间件系统结构

安全服务提供者 由基础安全提供商(如加密机厂商、智能卡厂商等)以基础安全服务接口的方式提供。本文将各种安全服务分为加密服务、证书服务、可信策略服务、数据存储服务和其他可扩展的服务,各种服务所对应的相应模块,由

收稿日期:2005 - 07 - 05;**修订日期**:2005 - 08 - 24

作者简介:曾令华(1968 -),男,贵州黄平人,讲师,主要研究方向:网络与信息系统安全、中间件技术; 余塑(1967 -),男,四川成都人,教授,博士,主要研究方向:网络与信息安全、分布式计算、中间件技术、电子商务、电子政务; 欧阳开翠(1967 -),女,贵州黄平人,讲师,主要研究方向:中间件技术; 周明天(1939 -),男,广西容县人,教授,博士生导师,主要研究方向:计算机网络、中间件技术、网络与信息系统安全、并行分布处理、计算机系统与软件.

通用安全服务管理器提供的动态加载机制灵活加载和卸载。

公共安全服务管理器 由安全中间件平台提供商提供,是安全中间件的核心。它屏蔽了基础安全服务算法复杂性和实现的细节,提供平台无关性和开发/运行高效性,使用应用与具体的实现无关。各高层模块利用它提供的接口,使用安全服务提供者提供的各种安全服务。

安全服务层 由体系结构或安全服务厂商提供,如常用的 TLS,SET,CORBA 安全服务和 J2EE 安全服务。该层延伸了现有的常规的安全中间件的概念,使安全中间件成为真正的"中间件",它提供了分布式环境下安全模块之间的互操作,不仅简化了体系结构级安全的开发和维护,还提供了基于对象的安全服务。

组件安全服务层 应由组件提供商提供,并以组件级接口方式出现,如商场组件工具、XML 工具等。组件安全服务常常建立在 CORBA 安全服务[1]和 J2EE 安全服务之上。

应用程序层 由具体的应用开发厂家提供,可使用如 C,C ++, Java 等语言开发应用程序。如果在组件级、安全服务级之上开发,它可以不用关心加密模块和网络分布的细节。如果在基础安全接口或通用级接口之上开发,该用户可以是安全专家。

资源信息服务器 为 5 层安全模块的灵活组态提供实质性支持,因为所有的安全服务、通用安全管理器和基础安全提供者都必须注册,提供其各种能力表。另外,如果为各个模块发布属性证书,安全中间件本身能保证其核心的安全性,特别对动态加载的扩展安全提供者,基于 PKI 的认证算法提供了强鉴别功能。

4 级接口分别是:组件级接口、体系结构接口、公共安全接口和基础接口。

在安全中间件的 6 大部分中,公共安全服务管理器(CSSM)处于核心位置。安全中间件屏蔽安全的不同实现、提供统一接口、调度安全任务等基本功能都是在这个模块中实现的。可以说 CSSM 本身就是一个原始的安全中间件。在其他几个部分中,安全服务提供者模块 SSPM 提供了最基本的安全服务,而 CSSM 负责管理 SSPM;安全服务层和组件服务层直接或间接地建立在 CSSM 的服务之上;应用程序使用安全中间件时必然要使用 CSSM 的服务;资源信息服务器是为其他模块提供服务的支撑模块;在安全中间件所提供的各级接口中,CSSM 提供的公共安全编程接口也是最核心的接口。

2 公共安全服务管理器

2.1 CSSM 体系结构

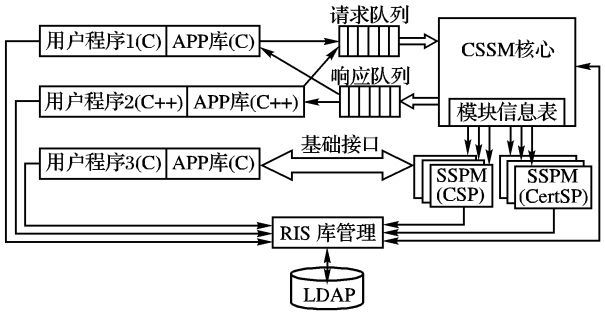

图 2 CSSM 体系结构

CSSM 的任务包括:管理执行具体加密任务的安全服务提供者模块,负责这些模块的加载和卸载,确保模块的可信性;向上层提供统一的接口,屏蔽各种安全服务实现之间的差异,使应用与具体实现无关;接收任务请求,并根据请求的不同将其分派到不同的模块,保证任务能并发高效地执行。安全中间件运行的基本功能和特性都在 CSSM 上得到解决。

狭义的公共安全服务管理器是指处于安全服务提供者模块和通用安全接口之间的一层软件结构,它通过请求/响应队列的同步机制、状态映射索引表和与队列有关的一些数据结构对请求/响应队列进行管理;利用模块配置文件、模块信息表对安全服务提供者模块进行管理、调度;在加载 CSP(加密服务提供者)、CertSP(证书服务提供者)等安全服务提供者模块时对它们作签名检查,在调用 SSPM 功能时对 SSPM 作相互的身份认证,保证 SSPM 的安全可信;并在调用者和 SSPM 之间进行通信,根据用户进程的请求调用各种服务提供者模块的相应服务函数,并对用户进程作出响应。

广义的 CSSM 还应该包括公共安全编程接口。它是安全中间件的核心接口,用户可通过该接口直接与 CSSM 交互。

2.2 CSSM 的安全服务过程

用户进程在开始请求安全服务前,首先向资源信息服务器(Resource Information Server, RIS)查询能够提供所需能力的安全服务提供者模块的信息,然后通过函数库的功能向 CSSM 发出调用某一功能的请求。函数库为用户在请求/响应队列中找到一个空闲的单元用于存放用户请求的功能类型、请求服务的数据和参数以及运行结果。

通过如信号量等同步机制,CSSM 核心得知请求队列中有用户请求到达,再利用队列状态索引表找到用户请求使用的单元,获取用户请求的类型。然后 CSSM 的 SSPM 管理模块根据请求的类型获取相应 SSPM 的句柄。CSSM 核心线程从线程池中获取一个空闲的子线程,并把这个子线程和指定的 SSPM 绑定在一起。子线程调用安全服务提供者模块中的运行任务函数实现用户要求的功能。如有必要,CSSM 可与 SSPM 进行相互的可信性检验[4~6]。

SSPM 的运行任务函数从请求队列中读取用户调用的数据、参数,完成用户要求的服务,把返回数据存放在请求/响应队列中的相应数据区,然后返回结果给 CSSM。CSSM 通过同步机制通知用户进程取回结果数据。用户进程取回数据后,一次安全服务过程结束。

在安全服务的过程中,安全服务由 SSPM 模块实际完成,而 SSPM 的查找、加载、绑定、引用记数、卸载等项工作由 CSSM 核心中的 SSPM 管理模块来完成,CSSM 核心本身并不提供安全服务。

调用 SSPM 模块时,若有必要,可采用 X.509 标准的三次握手来完成与安全服务提供者模块的交叉认证,确定双方的身份,在认证失败的时候阻止下一步的操作。

3 SSPM 管理模块的设计与实现

在 CSSM 体系结构中,CSSM 本身并不直接提供加密、解密、签名、验证、证书管理等安全服务功能。所有这些安全服务都是由各种 SSPM 来实现的,CSSM 只是提供对 SSPM 的管理,把任务分发给合适的 SSPM 处理。CSSM 利用模块配置表文件、RIS、模块信息表等结构实现对模块的管理。

3.1 模块信息表

模块信息表是 CSSM 中保存的一组数据,由 CSSM 在启动时创建,它保存了加载到内存中的各安全服务提供者模块的信息及服务函数的入口地址,CSSM 在接收到用户进程的请求时,首先在此表中查找模块,并最终用此模块中的服务函

数地址调用安全服务提供者模块的服务功能。每个表单元中主要包括以下数据:

m_hDll: 模块句柄;
m_Name: 模块名;
m_Type: 模块类型;
m_dispatchtype: 服务运行方式(线程或进程);
m_nMaxCount: 该模块能并行处理的最大请求数;
m_nCount: 目前该模块并行处理的请求数;
m_num: 该模块目前运行的总次数;
m_ok: 该模块运行成功次数;
m_fail: 该模块运行失败次数;
m_Index: 当前请求在队列中的索引;
m_nTTL: 该模块在生存时间;
m_Function: 访问该模块功能的入口函数;

3.2　模块配置表文件

模块配置表文件是一个文本文件,指明了 CSSM 在初始化时,需要加载的安全服务提供者模块的名称及其他具体信息。CSSM 在初始化时,读取该文件的内容,按照模块配置表文件的设置把指定的模块加载到内存中,再将相关的信息存入模块信息表。

模块配置表文件的主要内容包括:

name: 模块的名字;
path: 模块文件的实际路径;
Desc: 对模块的描述;
Dispatchtype: 模块的运行方式,0 为以线程方式运行;
Max: 模块能运并行处理请求的最大个数;
Init: 模块初始化时启动的个数;

3.3　RIS 中的模块表

所有的模块在安装时都需要在 RIS 的模块表、能力表中填写自己的资料信息。能力表中存放该模块的能力描述,用户进程可以查询该表,通过自己需要请求使用的服务,获得需要使用的模块名字。模块表中装有模块的配置信息及完整性检验所需的公钥证书和数字签名。模块表主要用于:1)CSSM 或模块可以通过该表查询判断某指定的模块是否已安装在本系统中; 2)模块在对自身进行完整性自检的时候,需要使用该表中的签名; 3)CSSM 在接到对某模块的请求而在模块信息表中未发现该模块时,将在 RIS 的模块表中查找该模块,如果找到,使用模块表中的模块文件路径加载该模块,并把相应的信息填入模块信息表。

模块表的主要内容包括:

Mod: 模块名字;
Path: 模块所在的路径和模块文件全名;
ServiceType: 模块的服务类型;
Version: 模块版本信息;
Manifest: 模块的公钥证书和数字签名;
Description: 对模块的描述信息;

3.4　模块加载过程

模块在安装到系统时,在 RIS 中的模块表中注册自己的名字、模块文件的全路径名称、自己的公钥证书[8,9]和数字签名[6]等信息;在 RIS 的能力表中注册自己拥有的能力。

CSSM 核心在启动的时候,读取模块配置表文件的内容。根据模块配置表文件中模块名在 RIS 的模块表中查找该模块的全路径文件名,将相关信息存入内存中的模块信息表。如果模块配置表文件中该模块的初始化个数大于零,则利用文件路径指定模块,把模块加载到内存中,并把此模块的服务函数入口地址保存到对应的模块信息表单元中。如图 3 所示。

当用户进程需要安全服务时,用户进程首先在 RIS 中查找自己需要的模块名;或是已知模块名,在 RIS 中确认该模块已经安装。然后通过库函数向 CSSM 核心提出对此模块的请求。这一请求包括 Load 和 Attach 两步。

当 CSSM 核心收到 Load 命令时,在模块信息表中查找该模块,找到后检查该模块是否加载到内存中。如果模块尚未加载到内存中,则根据模块信息表中的模块文件路径将该模块加载。如果未在模块信息表中找到该模块,则在 RIS 的模块表中查找该模块,找到后,将相关信息填入模块信息表,然后把模块加载入内存。如果在 RIS 中也未找到该模块或者是模块不能加载到内存中,都会向用户进程返回错误信息。

当 CSSM 核心收到 Attach 命令时,首先检查该模块是否已加载。如果没有加载则向用户进程返回出错消息。然后根据模块信息表中该模块的单元中,当前处理请求的个数是否已达到最大值判断,如果还能接受更多的请求,则将当前处理请求的个数加一,并将该模块信息表单元的索引值返回给用户进程,此后用户进程可以用此索引值请求更多的服务。如果有错误发生,则向用户进程发送出错消息。如图 4 所示。

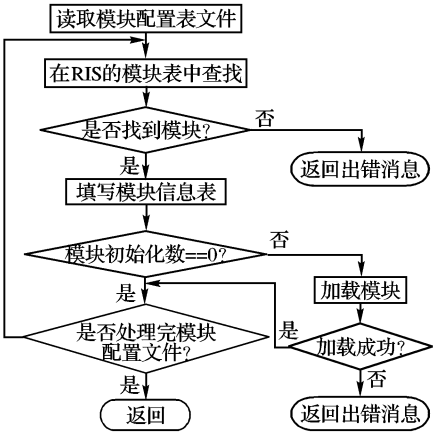

图 3　SSPM 的初始加载

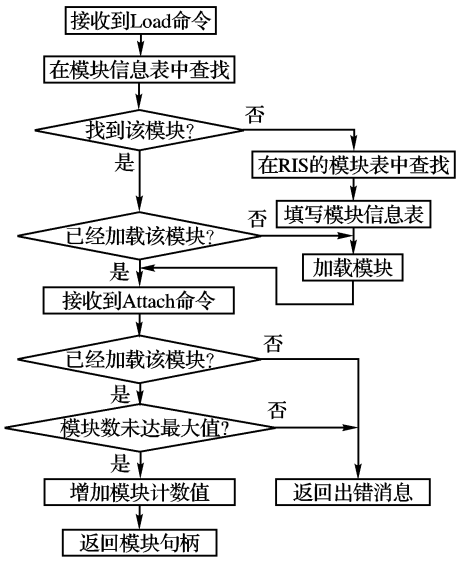

图 4　加载 SSPM 的过程

3.5　模块卸载过程

模块卸载过程相对于模块的加载过程要简单一些。用户进程在使用完所需的安全服务后,向 CSSM 核心发送 Detach 和 Unload 命令,以释放自己占用的服务资源。卸载过程如图 5 所示。

CSSM核心在收到Detach命令时,如果此模块当前处理

(下转第 2526 页)

由图5可知,随着预先免疫节点比例的增加,不仅感染节点数的峰值随之下降,且最终的稳定值也有所降低,说明了预先做好免疫工作是十分重要的,我们不能坐等病毒上门,而是要重视病毒的预防,积极升级系统补丁和病毒库,将病毒拒于计算机之外。

二为已免疫但一段时间后又失去免疫的节点数量。现实中,有部分已免疫的计算机由于病毒产生变种而自身未能及时升级病毒库,又重新转变为易感状态S。这部分节点的多少,同样反映了人们对后续病毒防治的重视程度,对于病毒的传播同样有重要的影响。图6分别显示了5%,10%,20%的原免疫节点失去免疫情形下病毒的传播情况。

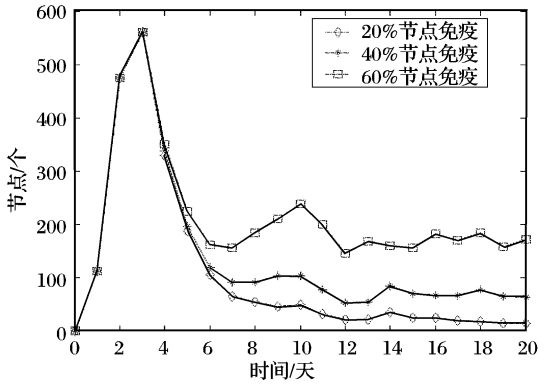

图6 不同比例节点失去免疫情形下的感染数曲线

由图6可知,失去免疫的节点比例增加,感染的节点在趋于一平衡值的数目也在增加,且波动也更剧烈。说明我们决不能放松对病毒的警惕,要经常升级更新系统和病毒库,以防病毒变种又一次入侵。

参考文献:

[1] 戴汝为, 操龙兵. Internet———一个开放的复杂巨系统[J]. 中国科学E辑, 2003, 33(4): 289-296.

[2] WATTS DJ, STROGATZ SH. Collective dynamics of 'small world' networks[J]. Nature, 1998, 393: 440-442.

[3] BARABASI AL, ALBERT R. Emergence of scaling in random networks[J]. Science, 1999, 286: 509-512.

[4] FALOUTSOS M, FALOUTSOS P, FALOUTSOS C. On Power-Law relationships of the Internet topology[J]. ACM SIGCOMM Computer Communication Review, 1999, 29(4): 251-262.

[5] 张成阳, 穆志纯, 王振花. 基于复杂性研究的Internet安全模型[J]. 北京科技大学学报, 2003, 25(5): 483-486.

[6] KEPHART JO, WHITE SR. Directed-graph epidemiological models of computer viruses[A]. Proceedings of the 1991 IEEE Symposium on Security and Privacy[C]. Oakland, California, USA: IEEE Computer Society Press, 1991. 343-359.

[7] PASTOR-SATORRAS R, VESPIGNANI A. Epidemic spreading in scale-free networks[J]. Physical Review Letters, 2001, 86(14): 3200-3203.

[8] PASTOR-STATORRAS R, BESPIGNANI A. Immunization of complex networks[J]. Physics Review E, 2002, 65(3): 036104-1-036104-8.

[9] LI X, CHEN GR. A local-world evolving network model[J]. Physica A, 2003, 328: 274-286.

(上接第2523页)

请求的个数大于0,则将它当前处理请求的个数减1。CSSM核心在收到Unload命令时,如果模块的当前处理请求的个数和模块初始化的个数都为0,就把该模块从内存中卸出,否则忽略该命令。

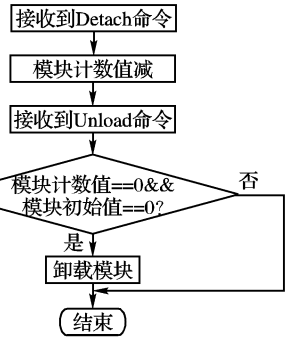

图5 卸载SSPM的过程

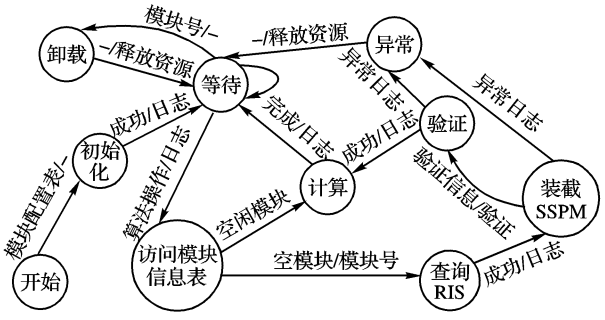

图6 SSPM管理模块状态变迁图

4 结语

公共安全服务管理器(CSSM)在安全中间件中起着承上启下的作用,通过使用CSSM中的SSPM管理模块对各种安全服务提供者模块实施管理,使应用软件与安全服务提供者相分离,从而屏蔽了不同安全服务的差异,实现各安全服务模块的加载/卸载自动化,提高安全系统的可移植性、可用性和可伸缩性。

另外,通过向上提供统一的编程接口,实现底层复杂的安全系统对应用软件的透明化,应用软件只需使用安全中间件提供的接口,即可方便地使用各种网络信息安全服务,从而缩短了应用软件的开发时间,降低了软件的开发成本,提高了软件的开发效率。

参考文献:

[1] The Open Group. Common Security: CDSA and CSSM, Version 2 (with corrigenda)[S], 2000.

[2] RAJAN A, WOOD M, BOWLER D. Mechanics of the Common Security Services Manager (CSSM)[Z], 2000.

[3] 佘堃, 周明天. 安全中间件核心——公共安全服务[J]. 小型微型计算机系统, 2003, 24(7).

[4] KRAWCZYK H, BELLARE M, CANETTI R. RFC 2104, HMAC: Keyed-Hashing for Message Authentication[S], 1997.

[5] DIFFIE W, HELLMAN ME. New Direction in Cryptography[J]. IEEE Transactions on Information Theory, 1976, (6): 644-654.

[6] RIVEST RL, SHAMIR A, ADLEMAN L. A Method for Obtaining Digital Signature and Public Cryptosystem[J]. Communication of the ACM, 1978, 21(2): 120-126.

[7] QUISGUATER JJ, COUVREVUR C. Fast Decipherment Algorithm for RSA Public-key Cryptosystem[J]. Electroinc Letters, 1982, 18(21): 905-907.

[8] RSA Laboratories. PKCS #1: RSA Cryptography Specifications Version 2.0[S], 1998.

[9] IEEE Std 1363-2000, IEEE Standard Specifications for Public-Key Cryptography[S], 2000.