高技术通讯 2010 年 第 20 卷 第 7 期:661 ~ 667

doi:10.3772/j.issn.1002-0470.2010.07.001

基于中间件规范的容侵应用服务器研究与实现①

郭渊博② 刘 伟 袁 顺 周睿鹏

(解放军信息工程大学电子技术学院 郑州 450004)

摘 要 针对现有的容忍入侵应用系统的构建需要针对不同的业务类型进行不同的个性化设计和开发以及系统或部件的可重用性较差的问题,在研究基于规范的容忍入侵中间件方法的基础上,提出了一种基于拦截器的容侵中间件模型,从逻辑上将系统分为容忍入侵服务提供者和容忍入侵服务管理者,可在满足既有规范情况下实现用户应用的业务逻辑与容忍入侵特性所依赖的非功能性服务的分离。对涉及的容忍入侵框架、容忍入侵策略组件、安全群组通信管理器、安全群组通信系统等进行了详细设计,在一个开源的 J2EE 应用服务器 JBoss 中实现了对容忍入侵功能的支持,可利用 Java 类加载机制完成容忍入侵服务的动态加载。

关键词 容忍入侵,应用服务器,中间件,J2EE,拦截器

0 引 言

容忍入侵是一种新型网络安全防护技术,它强调系统在某些部分受到攻击者破坏或被攻击者成功控制时如何继续对外提供服务,并确保系统中关键数据的秘密性和完整性。这种技术假设系统中不可避免地存在着一些无法被检测到的脆弱点,并且随着时间的推移,其中某些脆弱点可能会被入侵者利用。它立足于当系统遭到入侵和故障突然发生时能够利用"容忍"方法来解决系统的"可生存"问题,以确保信息系统的保密性、完整性、真实性、可用性和不可否认性,因而被认为是现代信息安全纵深防御中的最后一道防线。

然而,尽管研究人员在容忍入侵方法研究和技术应用等多个领域取得了长足进步[1],但现有技术在构建容侵系统时常存在一些问题,例如针对已有的商用现货(commercial-off-the-shelf,COST)系统增加容侵功能时,通常需要对原有系统进行大量修改,而在构建全新的容忍入侵业务系统时,针对不同的业务类型和性能需求要进行不同的个性化设计和开发,系统或部件的可重用性很差,要消耗大量的人力物力。针对上述问题,本文研究了一种基于规范的容忍入侵中间件实现方法,即将基于规范的中间件技术和容忍入侵方法相结合,给出满足容侵需求的应用服务器结构扩展设计。本文分析了容侵应用服务器中间件运行平台的工作原理,介绍了容忍入侵应用服务器的关键部件实现。

1 满足容侵需求的系统结构设计

1.1 设计思路

为了在中间件层实现用户应用的业务逻辑与容忍入侵特性所依赖的非功能性服务相分离,从而在满足既有规范的情况下屏蔽容忍入侵技术在应用层面实现时的高复杂性,确定了如下设计思路:首先对单个 J2EE (Java 2 platform enterprise edition)应用服务器进行容侵功能扩展,利用组件技术分析各种应用系统中的公共容侵服务需求,使容忍入侵的相关策略和措施的实现独立于应用系统的业务逻辑,提出基于规范的容忍入侵功能服务接口,使之成为通用的松耦合系统部件,在流行的中间件平台规范架构中无缝嵌入容侵功能,然后通过扩展后的 J2EE 应用服务器进行主从集群协作,以在中间件层面实现整体的容侵目标。

基于上述要求,提出了一种基于拦截器的容侵应用服务器中间件设计方法。该方法对单个 J2EE 应用服务器进行容侵功能扩展是基于 EJB、WEB 容

① 863 计划(2007AA01Z405)和河南省基础与前沿技术计划(082300413206)资助项目。
② 男,1975 年生,博士,副教授,硕士生导师;研究方向:容忍入侵与无线网络安全;联系人,E-mail:Yuanbo_g@hotmail.com
(收稿日期:2009-06-25)

器原有拦截器链进行扩充,根据容侵服务功能需求增加相应的拦截器,同时设计实现容侵功能的容侵组件,将其注册到 MBean Server,并对 MBean 的类和接口进行设计,达到容侵服务与具体业务逻辑相分离的目的。拦截器通过 MBean 代理调用相应的 MBean 的类或接口,完成相应的容侵功能接口调用。同时利用 Java 类加载机制,实现在程序执行的过程中动态地加载所需要的类文件,通过自定义的类加载器实现容侵服务的动态加载。

1.2 基于拦截器的容侵应用服务器中间件结构

基于以上设计思路,我们在现有 J2EE 架构下无缝地实现了两种扩展:一方面是对现有 Web 容器、EJB 容器及原有 JMS、JNDI、JDBC 等服务的功能扩充;另一方面是对容忍入侵框架、容忍入侵策略组件、安全群组通信管理器、安全群组通信系统的扩展及在此基础上的对容忍入侵服务功能部件的扩展。扩展后的 J2EE 应用服务器系统层次结构如图 1 所示,其中深色部分是新加入的部件,灰色部分是在现有部件基础上实现了相应的功能扩充。

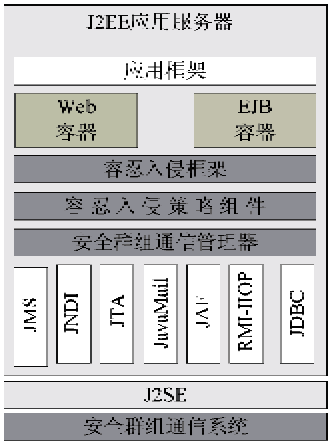

图 1 扩展的 J2EE 应用服务器层次结构图

扩展的 J2EE 应用服务器的功能实现图如图 2 所示。其中容忍入侵框架是基于 J2EE 应用服务器自身拦截器实现对 J2EE 应用服务器系统结构的扩展,它向上层应用提供容忍入侵服务。各扩展部件之间通过专门设计的 Java 接口进行交互,且保持松散耦合的关系。扩展部件间的交互过程如图 3 所示,可简单描述如下:首先,客户端向主应用服务器发送一个超文本传输协议(HTTP)请求或远程方法调用(remote method invocation,RMI)(以下简称调用请求),当该调用请求到达主应用服务器容器时,被容器调用者拦截,并将其转化成一个上下文对象 ContextObject 传送给相应的容器。这时容器装载拦截器,并调用位于拦截器链上第一个拦截器的 invoke()函数,并将上一步产生的 ContextObject 作为参数传递给该函数。然后拦截器使用 ContextObject 获取关于这个调用的相关信息,并调用相应的函数对 ContextObject 进行处理。处理完毕后,拦截器对 ContextObject 进行修改并将其作为参数传递给拦截器链中的下一个拦截器。当执行到 IT interceptor 时,IT interceptor 中的 invoke()函数将触发相应的容忍入侵服务的加载。如果此时容忍入侵服务不存在,则 IT framework 将根据配置文件 Eventmapping.xml 注册相应的 ITS component 使其成为一个组件实例。如果相应的 ITS component 实例存在但目前被钝化于实例池中时,则 IT framework 将其激活。而 ITS component 实例为了实现容忍入侵的功能,有时需要获取本机的信息,以及其它成员服务器的一些信息,则安全群组

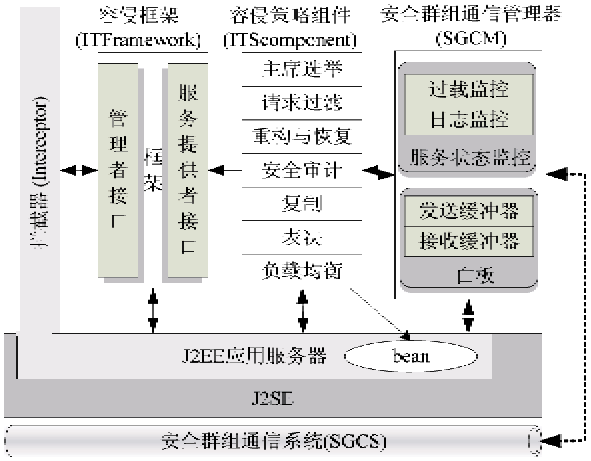

图 2 基于拦截器的应用服务器容侵功能实现图

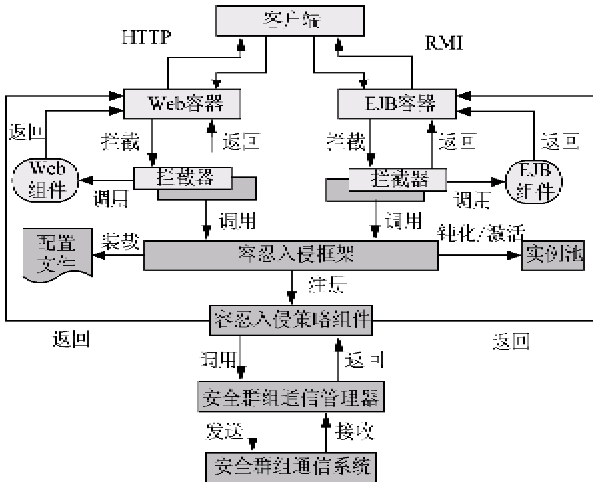

图 3 扩展部件之间的交互示意图

通信系统(SGCM)通过日志监控接口、过载监控接口、发送接口、接收接口向上层应用提供这些信息或参数,这种设计的优点是有利于减轻 ITS component 的计算开销,同时降低 SGCM 与 ITS component 的耦合度。之后,下一个 IT interceptor 接受 ContextObject,重复上面的处理过程,直到拦截链的最后一个 interceptor,最后一个拦截器是容器本身,ContextObject 被传递给容器组件的实例,由组件实例负责完成相应的业务逻辑。这样包含业务逻辑处理结果的 ContextObject 对象在 Interceptor 链中一步步返回,直到容器调用者。最后,容器调用者把调用请求的处理结果返回客户端。

可见,扩展后的 J2EE 应用服务器在完成原有的全部 J2EE 企业计算功能的基础上,将所需的容侵功能透明地集成到了现有 J2EE 应用服务器中,只要根据规范实现框架中的相关接口,用户就可以在不影响原有的 J2EE 计算服务的同时,来定制、开发、升级或扩展系统中的容忍入侵策略组件。

2 容侵应用服务器中间件平台的运行工作原理

容侵应用服务器中间件运行平台由 n 台经扩展后的 J2EE 应用服务器通过底层通信系统互联组成,采用半主动服务模式对外提供服务,其具体的工作过程如图 4 所示,工作原理如图 5 所示。其中 S_M 为主 J2EE 应用服务器(如图所示浅色部分),$S_1,\cdots,S_{M-1},S_{M+1},\cdots,S_n$ 为从应用服务器(图中深色部分,由于空间所限只画出一台从服务器,其余从服务器的工作方式与此从服务器完全相同),虚线部分代表各 J2EE 应用服务之间需要通过安全群组通信系统进行通信,平台内各成员服务器消息传递的可靠性、有序性及安全性由安全群组通信系统(SGCS)保证。S_M 与 $S_1,\cdots,S_{M-1},S_{M+1},\cdots,S_n$ 在结构上等同,并且可以互换,如果 S_M 在运行期出现故障,则平台将在 S_C 中重新选出一个新的主应用服务器 S'_M 以替代 S_M。图中 bean 表示 J2EE 组件。

① Init;

② 由 $S_1,\cdots,S_n$ 中的主席选举管理器,根据集群视图协商选出主应服务器 S_M,$M\in[1,2,\cdots,n]$,$n\in N$,选举结果通告给平台所有成员;

③ S_M 接受客户端调用请求,S_M 拦截器截获此调用请求并触发请求过滤操作;

④ S_M 中请求过滤器根据攻击特征库对调用请求进行分析,如果该调用请求不合法转到第③步,否则往下执行;

⑤ S_M 的复制管理器用来实现对客户端请求的复制以及请求处理过程中的相关状态复制,并根据服务列表将复制品发送到所有正确的应用服务器成员 S_C 中;

⑥ $S_i(i\in n$ 且 $i\neq M)$的拦截器截获此复制品,将其递交给目标组件 bean 的实例;

⑦ S_M 的组件实例开始处理客户端调用请求,$S_i(i\in n$ 且 $i\neq M)$开始处理客户端调用请求的复制品;

⑧ $S_j(j\in[1,2,\cdots,n],n\in N)$的安全审计管理器将此次客户端调用请求的处理过程记录到 D_j 中;

⑨ $S_i(i\in n$ 且 $i\neq M)$将 a_i 经 SGCM 发送到 S_M 的表决器中;S_M 的表决器对接收到的处理结果执行 $Majority()$,如果 $a_1=a_2=,\cdots,=a_{\lceil(n+1)/2\rceil}$ 或相差小于 t,则 S_M 将其中的一个返回给客户端,同时将 $\{f_1,f_2,\cdots,f_n\}$ 在平台中广播。如果有 n 个表决结果一致,则转到第③步,否则往下执行;

⑩ *if* $S_i(i\in n$ 且 $i\neq M)$产生了错误

then S_M 将本机的日志记录 D_M 的副本 D'_M 发送到平台中,产生故障的 S_i 在收到 D'_M 后,首先结合 D_i 进行攻击特征信息的提取,并将此特征在平台中广播,其它机器在收到此消息后,对本地攻击特征库进行更新;其次对本机进行重构与恢复操作,先是将本机与平台隔离,其次根据收到的 D_M 对本机进行恢复操作,待恢复完成后,重新加入到平台中; *endif*

if 至少($\lceil(n+1)/2\rceil$)个 $S_i(i\in n$ 且 $i\neq M)$发现 S_M 发生故障

then 转到第②步,新产生的主应用服务器 S'_M 将正确的 D'_M 在平台中广播,S_M 在收到 D'_M 后,首先结合 D_M 进行攻击特征信息的提取,并将此特征在平台中广播,其它应用服务器在收到此消息后,对本地攻击特征库进行更新;其次对本机进行重构与恢复操作,先是将本机与平台隔离,然后根据收到的 D'_M 对本机进行恢复操作,待恢复完成后,重新加入到平台中; *endif*

图 4 容侵应用服务器中间件平台工作过程

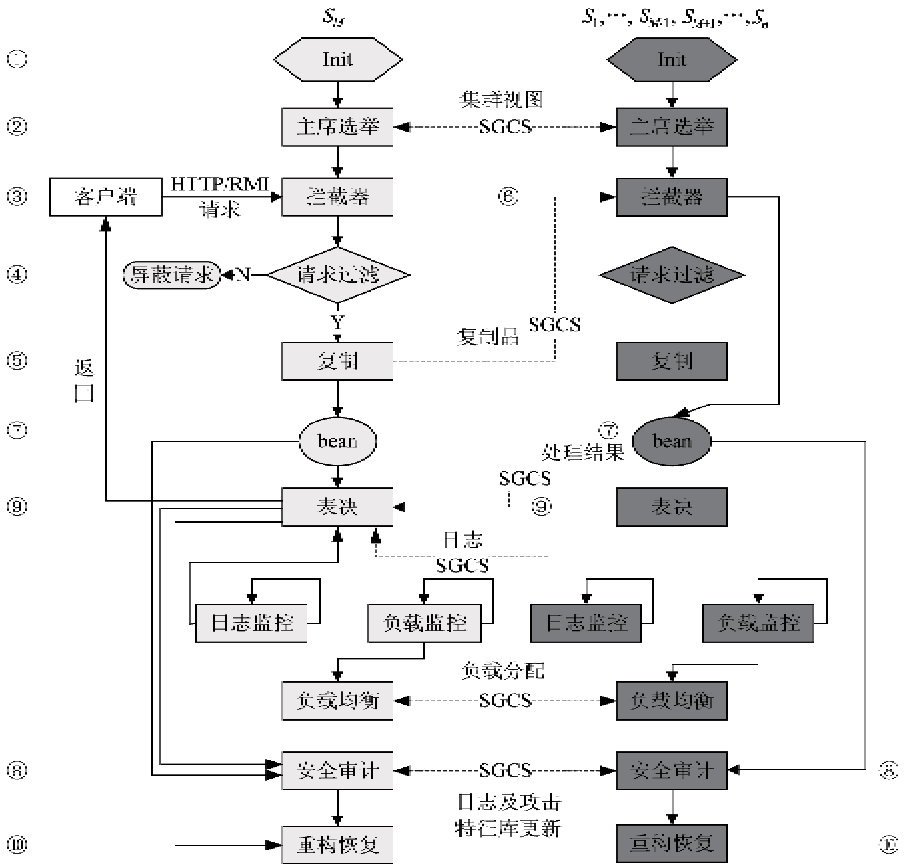

图 5 容忍入侵方法的实现原理图

3 容忍入侵应用服务器的实现

我们在开源的 J2EE 应用服务器 JBoss[2,3] 中利用 Java 类加载机制完成容侵服务的动态加载。扩展后的容忍入侵应用服务器由安全群组通信层、数据共享层、容侵框架层、应用管理层、数据层组成，如图 6 所示。其中群组层负责服务器群组底层可靠通信；数据共享层负责在群组范围内的数据共享服务；容侵框架层负责提供认证、加密签名、复制服务、表决服务、选举服务以及对这些服务的管理，其好处在于可以将基于现有或未来新开发的容侵技术所实现的容侵策略组件集成到容侵框架中；应用管理层完成系统的业务逻辑；数据层作为数据的载体为应用提供服务。从层次结构上来看，各层次逻辑上相对独立，扩展后的系统并没有改变 JBoss 本身的业务逻辑处理流程。

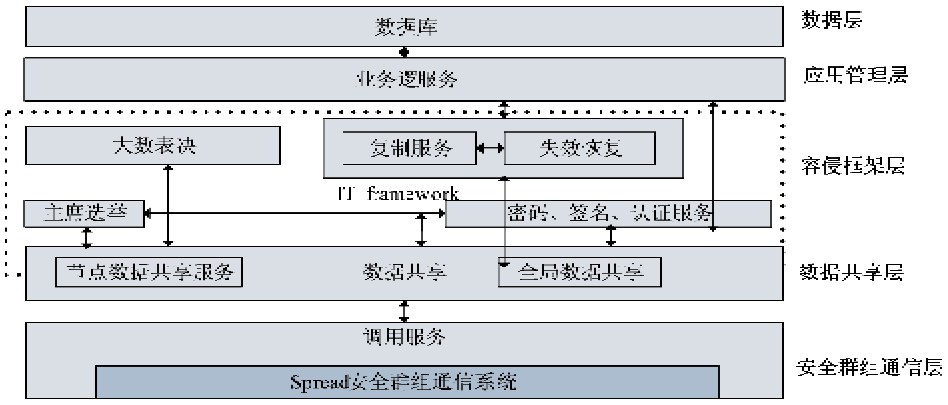

图 6 容侵服务器层次结构图

在安全群组通信层,我们用 Spread[4] 替代原有的 JGroup 作为群组通信工具,并完成以下功能:组的建立和删除、成员动态进入和离开时的消息通知、动态检测崩溃成员并删除该成员、发送和接收单点或多播消息、支持 TCP、UDP 等协议传输。同时在安全群组层中通过接收基本通信层的消息对上提供消息方法调用的传递,为群组节点提供一个关系控制服务,负责处理组内节点间一致关系。

数据共享层提供节点相关数据和全局数据的共享服务,对上层隐蔽实现细节,提供统一接口。节点通过该服务可访问各个节点的相关数据,但只有所属节点才能修改自身数据。用二维表来存储节点相关数据。若有节点加入,则将相关数据增加到表中,若有节点退出,则将该节点的数据删除。当表内的某个信息改变后,则通过底层的通信系统使其他节点上表中对应的数据同步更新。

容侵框架层是容侵服务执行的主体,是容忍入侵服务器设计的核心模块,实现容侵功能的集成与调用,容侵功能组件在容器中实现。其基本组成结构设计如图 7 所示。

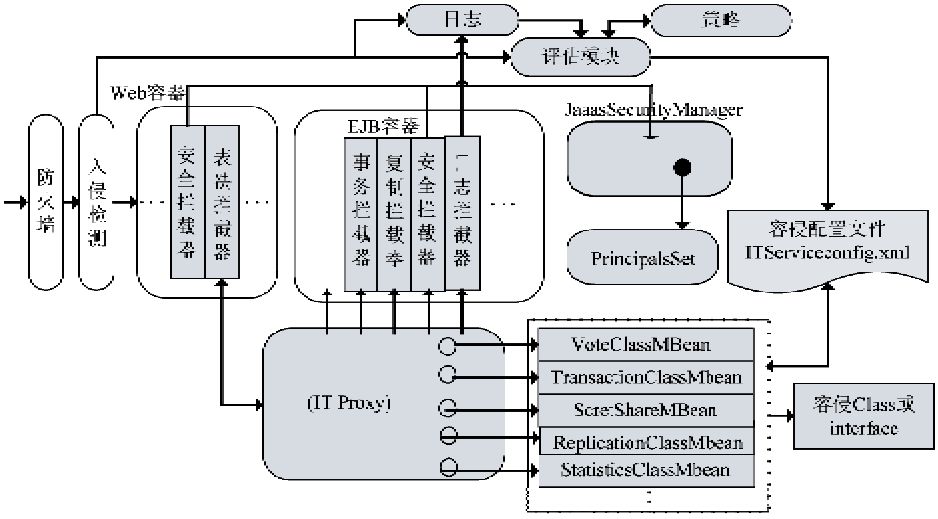

图 7　服务器端实现框架

首先在服务器端对原有的拦截器链进行扩充,增加相应功能的拦截器,其中表决和安全拦截器在 Web 容器,便于请求的转发和身份验证。EJB 容器中增加了日志拦截器、复制拦截器等。拦截器的功能是在方法 public void invoke (org.jboss.invocation.Invocation)中实现的,拦截器的配置在文件 standardjboss.xml 中完成。容侵配置文件中包含容侵服务的相关配置,包括采用何种容侵服务、何种算法等。评估模块对当前安全形势进行评估,从策略库提取策略,然后对容侵配置文件相关项进行动态配置。这里可以使用第三方的评估和策略库。MBean Server 充当容侵服务代理的角色,拦截器通过 MBean Server 获得具体容忍入侵 Mbean 的类名或接口函数名,再调用具体的类或接口函数。容忍入侵 MBean 对具体的容忍入侵服务提供所有必要的信息和操作功能,以便管理程序能够管理这些服务,在完成具体的功能时会查询容侵配置文件。系统的类加载架构在 JBoss 架构的基础上进行了扩充,增加 ITServiceClassLoader,以 UnifiedClassLoader 为父加载器,同时具体的容忍入侵服务类 ITClass 以 ITServiceClassLoader 为父类加载器。

应用管理层为上层提供业务逻辑服务,完成应用系统的相应业务,并对相应业务进行有效管理。

数据层以数据库为数据载体,根据系统的设计目标,数据库需要支持并发操作。

4　容忍入侵应用服务器的性能测试

为了测试设计的容忍入侵应用服务器中间件平台在各种故障发生时的性能表现,我们基于 JBoss4.0、JDK1.5.0 构建了一个容忍入侵 J2EE 应用服务器原型。测试工具采用专业的 J2EE 应用服务器基准测试工具 ECperf Kit 1.1,中间件平台由 5 台容忍入侵 JBoss 应用服务器通过以太网交换机相连,组成 100Mbps 的以太局域网,客户端通过路由与此局域网相连。

测试容忍入侵应用服务器中间件平台的性能的方式是根据应用服务器发生任意故障的个数来计算多个请求的平均执行时间,继而给出系统在发生故障时的性能情况,发生故障的服务器可能会以任意的结果对客户请求进行响应。我们将测试结果与JBoss4.0集群作了比较。表1给出了任意个故障发生时应用服务器中间件平台的性能情况。

表1 容忍入侵应用服务器中间件平台与JBoss4.0集群性能比较测试结果

故障个数	响应时间(μs)	
	容侵JBoss4.0平台	JBoss4.0集群
0	183.3	160.2
1	185.2	170.9
2	186.7	200.4
3	188.3	240.6

从表1可以看出,产生故障的应用服务器个数越多,平台响应客户端的时间就越长。具体情况如下:当故障增多时,JBoss4.0集群的执行时间开始逐渐多于容忍入侵的JBoss平台的执行时间。这是因为即使没有故障发生,容忍入侵的JBoss平台也要执行表决、散列、EVENODD编译码等操作,导致其性能低于JBoss4.0集群。但是随着故障个数的增多,JBoss4.0集群的性能开始明显降低,这是因为JBoss4.0集群此时需要做主/备服务器切换,因此增加了执行时间,并且故障越多服务器切换所需要的时间就越长。在相同情况下,容忍入侵JBoss中间件平台由于可以自动屏蔽故障机器,因而服务性能没有明显下降,因而其可生存性更高。

5 相关研究

中间件作为应用程序和操作系统之间的"桥梁",能够屏蔽底层的通信差异,在中间件层面上实现对容忍入侵机制的支持具有极高的成本收益比。当前国内外出现过一些与此相关的产品或系统设计,但大都基于CORBA中间件或针对web services类服务本身,下面简述几种具代表性的研究。

在DARPA资助下,Gregg Tally等人基于CORBA中间件开发了一个容忍入侵的分布式对象系统ITDOS[5,6]。其思路是从系统级入侵入手,使用代理防火墙、Byzantine一致性协商、表决技术进行防范。实现时以CORBA接口与上层应用和下层对接,接收到应用层的CORBA命令后,依照容忍入侵策略,使用CORBA语言与其它机器上的ITDOS部件协调,完成容侵功能。然而,ITDOS除了有限几种容忍入侵方法外几乎不可更改或添加,系统没法进行自适应。

Amjad Umar等人也在DARPA资助下开发了一个智能的补偿中间件ICM[7]。采用的技术主要有FRS(分割-冗余-分散)和门限密码技术,ICM定义了入侵触发器以及知识库用于各种情况下的入侵行为的识别,并通过拦截器向系统中加入容侵服务功能,而设计的调度程序则用来具体实现容侵功能的调用。但是由于容侵功能部分与COTS中间件其它部分在结构上存在紧密耦合关系,使得系统容忍入侵功能的添加和扩展不够灵活。

Carnegie Mellon的研究人员在美国军方资助下开发了一种基于Web技术的容忍入侵中间件Thema[8],Thema由客户端Library(Thema-C2RS)、BFT服务Library(Thema-RS)和外部服务Library(Thema-US)三个部分构成,各部分通过SOAP协议进行通信。C2RS、RS以及US主要用于请求响应信息的捆绑,BFT服务器则用于执行Byzantine一致性协商,以此增强Web服务的可生存性,Thema在实现容忍入侵功能时需要客户端的参与,使得整个系统的安全性具有一定的局限。

国内王树鹏、云晓春等人在分析大规模分布式应用的特点以及建立容灾系统存在问题的基础上,设计和实现了一种容灾中间件DTM[9]。通过采用虚拟服务端的方式实现了与客户应用衔接的透明化,通过加入配置更新部件增强了该中间件的灵活性和可配置性,将失效检测部件加入到容灾中间件中使得DTM可以及时了解服务端的状况。

6 结 论

目前已在流行的J2EE应用服务器JBOSS内部实现了对容忍入侵功能的支持,可在遵循J2EE规范的开发模式下进行应用程序的开发、部署和管理,而不用关心容忍入侵功能的实现细节,能够极大地提高容忍入侵应用系统开发、部署和维护的效率,具有极高的成本收益比。平台的实现可改变现有容忍入侵方法只是针对具体应用,对于不同的应用系统都要进行不同的系统开发的问题。

当前已在广域网络环境中实现了系统容忍入侵所需要的基本功能,可面向中小型规模的应用场合提供容忍入侵的Web应用服务开发。尽管在高安全的应用系统支持方面目前的版本中尚未嵌入实用

的密码技术增强,但先进的体系结构设计使得这种增强能够很容易地实现。

参考文献

[1] 郭渊博,马建峰.容忍入侵的国内外研究现状及所存在的问题分析.信息安全与通信保密,2005(7):337-341

[2] The JBoss Application Server. http://www.jboss.org/products/jbossas:JBoss Inc,2004

[3] Scott S. JBoss Administration and Development Third Edition. JBoss Group, LLC, 2003

[4] Amir Y, Nita-Rotaru C, Stanton J, et al. Secure spread an integrated architecture for secure group communication. *IEEE transactions on dependable and secure computing*, 2005, 2(3):248-261

[5] Sames D, Matt B, Niebuhr B, et al. Developing a heterogeneous intrusion tolerant CORBA system. In: Proceedings of the 2002 International Conference on Dependable Systems and Networks, IEEE Computer Society. Washington, DC, USA, 2002.387-396

[6] Tally G, Sames D, Matt B, et al. Intrusion tolerant distributed object systems: project summary. In: Proceedings of the DARPA Information Survivability Conference and Exposition, IEEE Computer Society. 2003.149-151

[7] Umar A, Anjum F, Ghosh A, et al. Intrusion tolerant middleware. In: Proceedings of the DARPA Information Survivability Conference & Exposition II, IEEE Computer Society. Anaheim, CA, USA, 2001.242-256

[8] Merideth M G, Iyengar A, Mikalsen T, et al. Thema: byzantine-fault-tolerant middleware for web-service applications. In: Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems, IEEE Computer Society. 2005. 26-28

[9] 王树鹏,云晓春,余翔湛,胡铭曾.一种容灾中间件的设计与实现.通信学报.2005,26(7):68-75

[10] 郭渊博,王亚弟,袁顺等. 基于J2EE中间件规范的容忍入侵应用服务器及容忍入侵方法. 国家技 术发明专利,申请号200710019118.9. 2007.11.20

Design and implementation of an intrusion-tolerant application server middleware-based

Guo Yuanbo, Liu Wei, Yuan Shun, Zhou Ruipeng

(Institute of Electronic Technology, the PLA Information Engineering University, Zhengzhou 450004)

Abstract

In consideration of the fact that the existing implementation of intrusion tolerance mechanisms is in the application layer, with the problem that it does not divide the business logic from the intrusion tolerance function, the paper proposes an intrusion-tolerance middleware model based on the interceptor on the basis of analyzing intrusion the tolerance requirements and the architecture of the J2EE application server, which logically divides the intrusion-tolerance functions into service providers and service management. A system was designed by using the intrusion tolerance security model and the element technology, the theory of intrusion tolerance mechanism and the strategy of implementation were described, and the intrusion tolerance technology and the implementation method were extracted. Finally, an intrusion-tolerance J2EE middleware was realized and built on the JBoss, an open source J2EE Application Server Platform, with the dynamic load of the intrusion-tolerance class being implemented through Java class load mechanism.

Key words: intrusion tolerance, application server, middleware, J2EE, interceptor