一种基于J2EE应用服务器的容侵自治愈方法

周睿鹏 郭渊博 刘 伟 韩磊磊
(解放军信息工程大学电子技术学院 郑州 450004)

摘 要 针对目前容忍入侵技术和软件自治愈技术的不足,在研究现有JANTM平台的基础上,提出了一种基于J2EE应用服务器的容侵自治愈模型,并在该模型下提出了一种容侵自治愈方法;和传统单一方法相比,该方法不仅解决了容忍入侵技术中存在的隐性入侵、软件老化以及容侵前提条件易遭破坏的问题,也改善了自治愈技术中未解决的外界入侵等问题;最后通过将容侵自治愈集群、JANTM集群和JBoss4.0集群进行比较测试,验证了该容侵自治愈方法可以使基于J2EE应用服务器的容侵自治愈集群有更高的可靠性和生存性。

关键词 容忍入侵,软件自治愈,J2EE应用服务器,容侵自治愈
中图法分类号 TP311 **文献标识码** A

Self-healing Intrusion Tolerant Method Based on J2EE Application Server

ZHOU Rui-peng GUO Yuan-bo LIU Wei HAN Lei-lei
(Institute of Electronic Technology, the PLA Information Engineering University, Zhengzhou 450004, China)

Abstract Aiming at the limitation for intrusion-tolerant and self-healing, this paper presented a self-healing intrusion-tolerant model based on J2EE application server, and presented an self-healing intrusion-tolerant method for this model. Compared with the traditional method, this method not only solved the limitation for intrusion-tolerant, for example, the hidden intrusion, software aging, and the vulnerable prerequisite of intrusion tolerance, but also solved the problem of intrusion in the self-healing. Finally, comparison tests of the self-healing intrusion tolerance clusters, JANTM clusters and JBoss4.0 clusters verify the self-healing intrusion tolerance method to make self-healing intrusion tolerance clusters higher reliability and survivability.

Keywords Intrusion tolerance, Self-healing, J2EE application server, Self-healing intrusion tolerance

随着国家信息化建设的不断深入,市场对基于J2EE(Java 2 platform enterprise edition)中间件规范的应用服务器产品的需求不断增加,而市面上的J2EE应用服务器产品,如IBM WebSphere,BEA WebLogic和JBoss等,虽然内置了安全服务如认证及授权等,但其在服务可生存性方面还存在着不同程度的缺陷。对此,郭渊博等人根据容忍入侵相关理论,设计开发了基于J2EE中间件规范的容忍入侵应用服务器平台—JANTM(J2EE based Adaptive iNtrusion Tolerant Middleware)[1],成功弥补了J2EE应用服务器在可生存方面的不足。

容忍入侵[2]是一种融合密码技术和容错技术的全新网络安全技术,强调系统的某些部分即使已经受到攻击者破坏或被攻击者成功控制时,系统如何继续对外提供服务,并保证系统中关键数据的秘密性和完整性,被认为是现代信息安全纵深防御中的最后一道防线。

容忍入侵J2EE应用服务器也有一定的安全隐患。首先,入侵若具有一定连续性和成功性,则会破坏容忍入侵屏蔽机制的前提条件,例如,JANTM中应用的大数表决机制的前提条件,即一共有$2f+1$台J2EE应用服务器,最多只能有f个J2EE应用服务器发生错误;其次,因为容忍入侵总是假定系统存在一些未知的脆弱点,当入侵者利用这些脆弱点发起攻击时,入侵有可能不会被检测出来和屏蔽掉,即发生隐性入侵,这时系统将进入一种无法预测的危险状态,即未知状态;另外随着J2EE应用服务器软件的长时间运行,自然会出现软件老化现象[3],使整个系统的性能下降,错误增多。在这三种情况下,系统的生存性将会大大降低,失去对外界提供正常服务的能力,甚至发生系统崩溃。

针对上述问题,本文提出了一种基于J2EE应用服务器的容侵自治愈模型,并在该模型下提出了一种容侵自治愈方法,该方法将软件自治愈技术与容忍入侵技术相结合,弥补各自缺陷,满足了用户对J2EE应用服务器的高可靠性和高生存性的要求。下面首先给出基于J2EE应用服务器的容侵自治愈模型—J-SHIRT(J2EE application server based Self-Healing IntRusion Tolerant model)的设计方案,然后提出了容侵自治愈方法,之后通过集群对比测试性能测试,验证了容侵自治愈集群的高可靠性和高生存性,最后对全文进行了总结和展望。

1 J-SHIRT模型

1.1 设计思路

大多数自治愈系统都是由检测部件、执行部件和管理部

到稿日期:2010-04-30 返修日期:2010-07-16 本文受国家863计划项目(2007AA01Z405)和河南省基础与前沿技术计划项目(082300413206)资助。

周睿鹏(1984—),男,硕士生,主要研究方向为网络安全、分布式应用。

件三部分组成的，它们与应用服务器构成一个自治愈环状结构，如图 1 所示。检测部件检测入侵和故障，并将结果上交给管理部件；管理部件对检测结果进行分析，给出治愈方案，发送到执行部件中；执行部件根据治愈方案对应用服务器进行治愈。

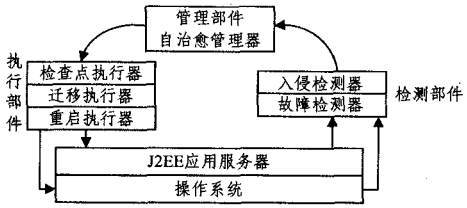

图 1　治愈功能组件间交互示意图

JANTM 平台是项目组基于 J2EE 规范下对开源应用服务器 Jboss[4] 进行容忍入侵功能扩充完成的。设计思路如下：首先利用 J2EE 规范中的组件技术和拦截器(interceptor)技术，对单个容忍入侵应用服务器进行自治愈功能扩展，设计加入的自治愈功能组件包括检测部件、管理部件和执行部件，再分别对每个部件进行具体的功能设计，如图 1 所示；然后通过扩展后的容忍入侵自治愈应用服务器进行主从集群协作，以在单个服务器层面和整个系统层面实现自治愈目标。

1.2　模型总体结构设计

基于上述设计思路，J-SHIRT 模型是由 J-SHIRT 应用服务器和 J-SHIRT 平台两部分组成的。J-SHIRT 应用服务器(如图 2 所示)包括容忍入侵功能组件、自治愈功能组件、安全群组通信系统及 J2EE 应用服务器基础平台。其中容忍入侵功能组件包括请求过滤器、复制分发器、表决器、主席选举器和日志审计器，它为应用服务器提供了以表决技术为主的容忍入侵支持，其表决结果作为自治愈功能组件的输入。

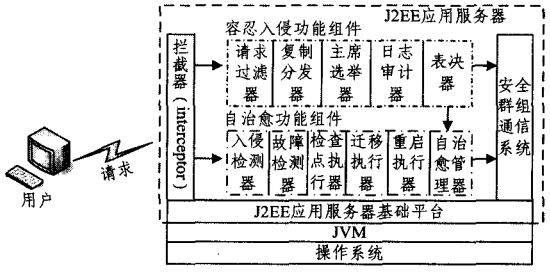

图 2　J-SHIRT 应用服务器结构结构图

自治愈功能组件包括检测部件、执行部件和管理部件。检测部件由入侵检测器和故障检测器组成，用于检测客户调用请求和应用服务器及其组件运行，发现入侵和故障，该部件的输出作为管理部件的输入；管理部件由自治愈管理器组成，用于分析检测部件的输出结果，根据定义的自治愈算法，给出针对此输出结果的自治愈方案，并根据方案，向各个执行部件发送命令，管理部件输出作为执行部件的输入；执行部件由检查点执行器、迁移执行器和重启执行器组成，用于接收管理部件的执行命令，并根据执行命令对发生入侵或故障的应用服务器及其组件进行自治愈处理。其中检查点执行器用于对正常应用服务器的业务组件进行检查点设置，保存当前正常的业务状态，以备治愈时使用；迁移执行器用于将正常应用服务器的业务组件状态迁移到无业务状态的应用服务器中继续运行；重启执行器用于重启应用服务器或其组件，使之恢复到初始化状态。

安全群组通信系统[5] 替换原 J2EE 应用服务器的群组通信系统，其通过 API 接口与自治愈功能组件和容忍入侵功能组件进行交互，为它们提供一套安全可靠的群组通信机制和组成员管理机制。

如图 3 所示，J-SHIRT 平台是由 N 模冗余容侵自治愈应用服务器组成的一个对外提供统一服务的 J2EE 应用服务器集群，运行在不同操作系统上的各应用服务器作为集群内的组成员，各组成员通过安全群组通信系统进行通信，并拥有唯一的标识。此时 S_1 为主应用服务器，$S_2 \cdots S_{n+k}$ 为从应用服务器；在线的应用服务器为 $S_1 \cdots S_n$，而 $S_{n+1} \cdots S_{n+k}$ 为此刻离线恢复的应用服务器。

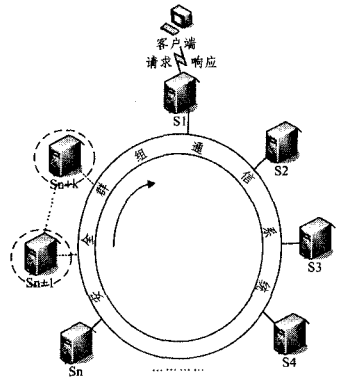

图 3　J-SHIRT 平台结构示意图

2　容侵自治愈方法

本文提出了将容忍入侵和软件自治愈两项技术相结合的容侵自治愈方法。其基本思想是在应用服务器集群系统中以周期性恢复为主，对未被容忍入侵技术检测和屏蔽的入侵进行周期性的治愈，防止了应用服务器进入未知状态，同时预防了应用服务器的软件老化；以反应式恢复为辅，对被容忍入侵技术检测和屏蔽的入侵和故障进行反应式的治愈，而且面对不同类型的入侵和故障时，系统会自动选择相应的治愈策略和恢复技术进行处理。这种方法进一步提高了应用服务器的安全性和可靠性，并有效地降低了自治愈的开销。另外在自治愈后，原有的应用服务器对相同的入侵攻击会具有一定的免疫力。

2.1　约定及前提条件

首先做如下约定：

N 表示集群中定义的容侵自治愈应用服务器的总数，$N \geqslant 4$ 且 N 为自然数，$N=2f+k+1$；

n 表示集群中定义的在线的容侵自治愈应用服务器的总数，$n \geqslant 3$ 且 n 为自然数且 n 为素数，$n=2f+1$；

f 表示集群系统定义的最多可以容忍的入侵和故障应用服务器的个数，f 为自然数；

k 表示集群中定义的离线容侵自治愈应用服务器的个数，k 为自然数；

i 表示集群中容侵自治愈应用服务器的 ID 号码；

S_i 表示第 i 个容侵自治愈应用服务器；

S_L 表示主应用服务器；

S^i 表示第 i 个容侵自治愈应用服务器的状态，其中包括 failure，online，offine，standby 等状态；

Clusterview 表示集群视图，其中存有各个容侵自治愈应用服务器的状态；

〈Failure〉表示被检测出发生故障和入侵的应用服务器的

集合；

|{Failure}|表示{Failure}中的容侵自治愈应用服务器个数，|{Failure}|为自然数且|{Failure}|$<f+k$；

{Online}表示当前在安全群组通信系统中被设为在线的应用服务器集合；

{Offline}表示当前在安全群组通信系统中被设为离线，且其正在进行自治愈的应用服务器集合；

{Standby}表示当前在安全群组通信系统中被设为离线，且已经完成自治愈过程，能够上线进行服务的应用服务器集合；

Leader()表示主席选举函数；

R_i 表示 S_i 对客户端请求的反馈结果；

majorityvoting()表示大数表决函数；

F_i 表示 S_L 的表决器对 S_i 发出故障信号；

Synclock()表示同步时钟函数；

getclock()是用来返回当前时间的函数；

Heal()表示应用服务器自治愈操作；

Online()表示应用服务器上线操作；

T_H 表示一台容侵自治愈应用服务器自治愈所需的最大时间；

T_{off} 表示主动式自治愈中容侵自治愈应用服务器的离线时间；

t_h 用来保存触发容侵自治愈应用服务器的瞬时时间。

前提条件：

1. 消息在安全群组通信链路中的传输延迟可以忽略不计；

2. 集群中的同步时钟通过时钟同步协议确保同步时间准确不误；

3. 自治愈管理器为可信组件，且不发生故障；

4. 在执行主动式自治愈的 T_H 时间段内发生入侵和故障的容侵自治愈应用服务器个数不会超过 f。

2.2 容侵自治愈步骤

(1)容忍入侵步骤

1a)由所有在线应用服务器的主席选举器根据安全群组通信系统提供的 Clusterview 在其{Online}集合中执行 Leader()，选举产生 S_L，$L\in\{1,2,\cdots,n\}$，并将选举结果通告给群组中所有组成员，其他剩余的应用服务器皆为从应用服务器；

1b)S_L 接收客户端请求，并通过拦截器触发请求过滤器对客户端请求进行过滤检测，如果合法则继续执行 1c)步，否则拦截非法请求，重复 1b)步；

1c)S_L 的复制分发器对合法客户端请求进行复制分发，$S_i(i\in\{1,2,\cdots,n\}$且 $i\neq L)$的拦截器截获此复制品，建立相应的 EJB 组件实例进行处理；

1d)$S_i(i\in\{1,2,\cdots,n\}$且 $i\neq L)$将 R_i 经安全群组系统发送到 S_L 的表决器中，执行 majorityvoting()，将正确结果返回给客户端，此时如果全部表决结果一致，则转到第 1b)步，否则往下执行；

1e)S_L 的表决器将 F_i 发送给 S_i 的自治愈管理器，并在群组中将其状态 S^i 标为 failure 状态，分入{Failure}集合中，此时如果 $i=L$，则所有在线应用服务器重新执行 Leader()，选举产生新的 S_L。

(2)主动式自治愈步骤

2a)每台应用服务器的自治愈管理器通过调用 Synclock()来建立一个虚拟全局时钟，使所有自治愈管理中的同步时钟同时启动，并将初始值设为 0；

2b)在时间同步之后，自治愈管理器根据 S_i 自身应用服务器的 ID 号码 i 与当前全局时间，得到 $getclock()=t_h$ 时，触发第一组 k 个应用服务器离线，S^i 标记为 offline，分入{Offline}集合中，并进行 Heal()，每一个应用服务器的 Heal()都会在 T_H 内完成。此时，若 S_L 其将要进行离线自治愈，则 S_L 将会通知所有应用服务器进行新一轮的 Leader()，选出 S_L' 来代替 S_L 成为新的主应用服务器；

2c)S_i 执行完 HeaL()之后，会将自己在群组中的 S^i 设置为 standby，并分入{Standby}集合中，等待上线进行服务；

2d)根据 S_i 自身应用服务器的 ID 号码 i 与当前全局时间，得到 $getclock()=\left\lceil\frac{i}{k}\right\rceil T_{off}$ 时，进行 Online()操作；

2e)执行完 Online()操作后，使 $i=i+N$，以此循环。

(3)反应式自治愈步骤

3a)在应用服务器级别上，首先通过服务器层故障检测器和入侵检测器对应用服务器进行检测，将发生故障的应用服务器状态 S^i 标为 failure，分入{Failure}集合中，此时如果 $i=L$，则所有在线应用服务器重新执行 Leader()，选举产生新的 S_L；

3b)首先判断集群中|{Failure}|与|{Standby}|的大小，若|{Failure}|$\leqslant$|{Standby}|，则转到 3c)步，否则转到 3d)步；

3c)S_L 的自治愈管理器通过安全群组通信系统提前向{Standby}集合中的|{Failure}|台应用服务器发出上线通知，以代替发生故障的|{Failure}|台应用服务器，确保在线 J2EE 应用服务器总数为 n，避免表决机制的前提条件遭到迅速破坏；

3d)S_L 的自治愈管理器通过安全群组通信系统通知{Standby}集合中的 k'台应用服务器提前上线，$k'<$|{*Standby*}|，使当前在线应用服务器总数 $n'=n-$|{*Failure*}|$+k'$为此刻群组中能够形成的最大素数，满足表决机制中表决总数应为素数的前提条件；

3e)对{Failure}中的应用服务器执行 Heal()，完成自治愈后，执行 Online()；

3f)在组件级别上，通过每台应用服务器自身的组件层故障检测器对其组件进行跟踪检测的反馈结果，自治愈管理器锁定发生故障的应用服务器组件，对其进行分析，如果此组件没有其它关联组件，则通过组件层重启器对其发起重启自治愈；如果此组件有关联组件，则通过组件层重启器对其和其它关联组件发起递归重启自治愈。其具体过程为：首先摧毁相应的组件实例，终止这些实例的守护线程，释放相关的引用资源，删除维组件持组件的服务元数据，最后重新实例化和初始化组件。

其中 Heal()的具体过程如下：

首先通过该应用服务器中的服务器层重启器关闭应用服务器软件，然后，触发系统层重启器关闭并删除整个操作系统，之后从一个干净的只读介质中随机考取一份系统代码镜像，最后通过重启执行器重启系统以及应用服务器程序，使应用服务器系统还原到相对安全的初始化状态。

其中 Online()的具体过程如下：

首先 S_i 重新加入安全群组通信系统并获得新的通信密钥，同时其迁移执行器会向 S_L 的检查点执行器发送业务状

态迁移请求，S_L 通过检查点执行器保存正确的业务状态后，利用迁移执行器将 S_L 正确的业务状态迁移至 S_i 中继续执行。

2.3 容侵自治愈算法

Intrusion-tolerant Self-healing Algorithm//容侵自治愈算法

Init;

//容忍入侵

1a:Leader();//主应用服务器选举

$S_i=S_L, S_i\in$ {Online};

Broadcast();//在集群中进行广播

1b:while(request=illegal)

{requestfiltering()};//请求过滤

1c:replicationdistribution();//S_L 对合法客户端请求进行复制分发

1d:majorityvoting();//混合型大数表决

Send the R_L to the client;

if($R_1=R_2=\cdots R_n$)

{go to 1b};

1e:send F_i to S_i;

S^i=failure, $S_i\in$ {Failure};

if(i=L)

{Leader()};

//主动式自治愈

proactive heal()

{2a:Synclock();//同步全局时钟

$t_h=0$;

2b:$t_h\leftarrow$getclock()+($\lceil \frac{i}{k} \rceil -1)T_{off}$;

loop;

wait until getclock()=t_h;

Si=offline, $S_i\in$ {Offline};//离线操作

if(i=L)

{Leader()};

heal();//自治愈操作

2c:S^i=standby, $S_i\in$ {Standby};

2d:wait until getclock()=$\lceil \frac{i}{k} \rceil T_{off}$;

Online();//上线操作

2e:i=i+N;

end loop;}

//反应式自治愈

reactive heal()

{3a:detect();//故障和入侵检测

S^i=failure, $S_i\in$ {Failure};

if(i=L)

{Leader()};

3b:if(|{Failure}|≤|{Standby}|)

{go to 3c;

else go to 3d;}

3c:while(|{Online}|≠n)

Online the |{Failure}| server in the {Standby};

S^i=online, $S_i\in$ {Online};

|{Online}|=|{Online}|+|{Standby}|;

3d:while(|{Online}|≠n′,

{n′|n′<n+|{Standby}|−|{Failure}| and n=max prime})

Online the k′ server in the {Standby};

S^i=online, $S_i\in$ {Online};

|{Online}|=|{Online}|+ k′,

k′<|{Standby}|;

3e:Heal();

Online();}

end;

2.4 算法分析

本文提出的容侵自治愈方法，其基本思想是在所设计的 J-SHIRT 平台中以主动式自治愈机制为主，对未被容忍入侵屏蔽技术检测出和屏蔽掉的入侵或故障进行周期性的自治愈，防止应用服务器轻易进入未知状态，也预防在应用服务器中软件老化现象的发生；同时以反应式自治愈为辅，对被容忍入侵技术检测和屏蔽的入侵和故障进行反应式的自治愈；面对发生在不同级别上的入侵和故障，算法会选择相应的自治愈执行器进行处理。

将本文提出的算法与国内外相关研究的容侵自治愈算法进行比较分析可以看出，本文算法能够有效地弥补单一反应式或主动式算法的缺陷，适用于强攻击、隐性入侵和软件老化存在的环境，具有更好的实用性和更高的安全性。具体如下：

李庆华、张胤等人提出的一种基于 CORBA 分布式对象容忍入侵系统模型自治愈算法[6]，是一种单一的反应式自治愈算法，当系统面对隐性入侵和软件老化的威胁时，该算法将无法处理。

周华、孟相如等人基于容忍入侵系统的可信实时计算基所提出的主动式自治愈算法和 Yih Huang, David Arsenault 等人基于其设计的 SCIT 系统所提出的容侵自治愈算法[7]，都是一种周期性自治愈算法，可以在一定程度上解决隐性入侵和的问题，但是简单的周期性自治愈很容易被攻击者所利用，发动周期性间隔式攻击会使其失效。

Paulo Sousa, Alysson N. Bessani 等人基于其设计的 CIS 所提出的容侵自治愈算法[8]与本文所提出的容侵自治愈算法一样，结合了反应式和主动式算法的优点，是一种混合型的容侵自治愈算法，但其与本文所提出的容侵自治愈算法相比存在着以下不足：

1. 在该算法中，容忍入侵系统的冗余副本一旦被确认发生入侵和故障，整个副本会立刻自治愈，这样攻击者并不需要费很多精力去破坏整个副本，只需要使副本中的某个重要组件发生异常，就可以让系统本身自己去隔离该副本进行自治愈处理，其效果与破坏整个副本一样都成功减少了容忍入侵系统中的可用副本数，从而达到破坏整个容忍入侵系统的目的。

本文提出的容侵自治愈算法采取了分级自治愈的思想，即一旦冗余副本出现异常行为，首先自治愈冗余副本的异常组件，如不成功，再一级一级向上递归进行自治愈，最终使副本恢复正常。

2. Paulo Sousa 提出的算法应用在其设计实现的容侵自治愈防火墙中，并指出此防火墙是不保存其运行状态的。所以该算法无法解决自治愈后的容忍入侵系统节点的状态同步问题。本文研究的 J-SHIRT 服务器是有运行状态的，而本文提出的容侵自治愈算法成功地解决了这个问题。

3. Paulo Sousa 的算法，主要针对 Byzantine 失效进行处理，并没有全面考虑到容忍入侵系统中存在的其它入侵或故障，与本文提出的容侵自治愈算法相比，主要未对容忍入侵系统中存在的隐性入侵威胁给出解决方案。

3 性能测试

测试环境：为了测试所设计的 J-SHIRT 在处理各种故障时的性能表现，本文基于 Jboss5.0，JDK1.5.0 构建了一个 J-SHIRT 平台原型，在该原型平台中完整地实现了本文所提出的容侵自治愈算法。与此同时，为了更好地体现出本文设计的平台的性能指标，同样配置了 JANTM 平台和 Jboss 平台。测试工具采用专业的 J2EE 应用服务器基准测试工具 ECperf，版本为 ECperf Kit 1.1，此时定义的 J-SHIRT 服务器总数 $N=7$，在线的应用服务器总数 $n=5$，最多可以容忍的入侵和故障应用服务器的个数 $f=2$，离线的容侵自治愈应用服务器的个数 $k=2$，其中 $N=2f+k+1$，$n=2f+1$。由于自治愈时间测试的需要，每台 J-SHIRT 服务器的机器配置均相同，CPU 为 Intel Pentium4 2.5GHz，内存为 Kingston 2G，而底层操作系统是在 Ubuntu 9.04，RedHat Linux 5，Windows XP 和 Windows 2003 server 等洁净系统中随机选取的。另外，为了方便测试，在测试实验中以上所有的系统硬盘镜像的大小均为 1GB。它们通过 Cisco 百兆以太网交换机相连，组成 100Mbps 以太局域网，而客户端(Client)通过路由器与此局域网相连。

这里需要指出的是，进行对比实验的 Jboss 平台和 JANTM 平台，是由 5 台应用服务器组成的集群。之所以只选用 5 台应用服务器，是因为本文设计的平台中有两台应用服务器是“离线”的，它们是为了实现本文所提出的容侵自治愈方法设置的，作为硬件冗余，解决隐性入侵和软件老化等问题，整个平台中真正提供在线服务的应用服务器数量为 5 台。如果这两个对比测试平台也是由 7 台应用服务器组成，其处理业务的性能会与本文所设计平台的性能有差别，所以在本文的测试实验中，选择了相应的 5 台应用服务器组成对比测试平台。

3.1 自治愈时间测试

本文需要在第一个测试实验中得出，在目前这种配置下，J-SHIRT 平台中参数 T_H 的大小，即任意一台 J-SHIRT 服务器完成自治愈 heal()操作过程所需的最大时间为多少。测试求出 T_H 的大小后可以进一步得出 T_{off} 的值，也就得出了平台中主动式自治愈步骤所需的大体时间周期是多少，以便为以下要进行的测试实验做好准备。

本文通过对每一台 J-SHIRT 服务器的完整治愈过程进行 10 次的重复实验得出了以下数据。表 1 中显示了每一台 J-SHIRT 服务器的操作系统的平均关闭时间、平均自治愈时间、应用服务器及操作系统重新的平均启动时间以及平均总共耗时。其中自治愈 Heal 操作是通过随机拷贝本地硬盘中的洁净系统镜像文件来完成的，系统镜像为 Ubuntu 9.04，RedHat Linux 9.0，Windows XP 和 Windows 2003 server 等不同的洁净镜像文件。系统镜像文件中包括 J-SHIRT 服务器的相应软件。另外，为了方便测试，在本测试实验中使用的所有系统镜像文件的大小均被定为 1GB。

从表 1 可以看出，虽然每台 J-SHIRT 服务器的硬件配置均相同，但是由于其应用服务器程序运行的底层操作系统并不一样，因此每台应用服务器的总自治愈过程耗时有所偏差。如表 1 所示应用服务器的整个自治愈过程最大耗时为 168.8 秒(2.8 分钟)。另外，从表 1 还可以看出自治愈过程的主要耗时集中在从本地硬盘中随机拷贝预先配置好的洁净系统镜像和启动新的系统镜像以及应用服务器程序上。所以说，如果我们能够选择一个更小的系统镜像，并且在启动时关闭其他并不需要的系统服务，就可以有效地降低自治愈最大耗时，为以后的实际应用创造条件。

表 1 J-SHIRT 自治愈时间表(单位为秒)

Server	Shutdown	Heal	Restart	Total
S1	31.6	74.2	60.1	165.9
S2	31.8	74.3	61.2	167.3
S3	28.5	73.8	55.7	158.0
S4	30.2	73.9	59.9	164.0
S5	27.6	73.0	54.6	155.2
S6	32.4	74.9	61.5	168.8
S7	27.9	74.0	54.8	156.7
Max	32.4	74.9	61.5	168.8

根据以上测试实验得到的数据和本文提出的平台工作方法可以得出，T_H 必须取自治愈耗时的最大值。另外，考虑到运行时其他各种因素，实际设置的 T_H 应比实验得出的最大值 168.8 秒还要长一点，所以在本文的以下测试实验中将 T_H 设置为 180 秒，即 3 分钟。

已知 $T_H=180$s，下面再确定一下 T_{off} 的值，由本文提出的容侵自治愈方法可以得出，T_{off} 值的大小要充分考虑到反应式自治愈的要求，即为在两次主动式自治愈执行之间的时间段内可能发生故障和入侵的应用服务器进行反应式自治愈留出足够多的时间，所以 $T_{off}>2T_H=360$ 秒 $=6$ 分钟。此外，本文设计的 J-SHIRT 平台，还要考虑软件老化的影响，所以 T_{off} 的值又不易过小。综合两方面的因素，将本文测试实验中的 T_{off} 设置为 600 秒，即 10 分钟。

3.2 DoS 攻击测试

在本次测试实验中将会增加一个新的恶意客户端来模拟来自外部网络的 DoS 攻击，以测试本文设计平台的可靠性和生存性。恶意客户端将向实验平台发动一个流量从 0Mbps 到 100Mbps 的 DoS 攻击。而平台原有的正常客户端也将同时向实验平台源源不断地发送数据包，平均每秒 200 个包，每个数据包大小为 2048bytes，以模拟平时正常的网络流量。这里需要注意的是，本实验平台设置了一个合法的流量前提条件，即每秒接收 800 个数据包为本平台的最高合法流量。

下面将在本文设计的 J-SHIRT 平台，JANTM 平台和 Jboss 平台下，分别进行吞吐量和响应延迟时间测试，以获得各平台的相应数据，通过相互对比发现其中的问题。前者是通过测量客户端收到的反馈信息率来获得平台的吞吐量，后者是通过计算客户端发送数据包和接收反馈信息之间的往返时间来得到响应延迟时间。以下测试实验结果都是通过 100 次反复实验测试所得，图 4 分别显示了各平台的响应延迟时间的平均值和吞吐量的最大值。

从图 4(a)中可以看出，在恶意客户端发动的 DoS 攻击流量达到 60Mbps 之前，本文设计的 J-SHIRT 平台的运行是基本正常的，没有太大波动，而 JANTM 平台在 DoS 攻击流量达到 50Mbps 之后就增加很长的响应延迟时间，甚至普通 Jboss 应用服务器平台在 DoS 攻击流量为 30Mbps 时，延迟时间就会相当大。J-SHIRT 平台一直到 DoS 攻击达到 70Mbps 时，平台的运行才优雅地降级，而 JANTM 平台达到 60Mbps 服务就会出现降级，普通 Jboss 应用服务器平台更差。

在图 4(b)中显示，随着恶意客户端发动的 DoS 攻击流量从 0Mbps 到 100Mbps，三个平台的吞吐量也会随之下降。但是本文设计的 J-SHIRT 平台下降得最平缓，而且下降临界点出现得最晚。

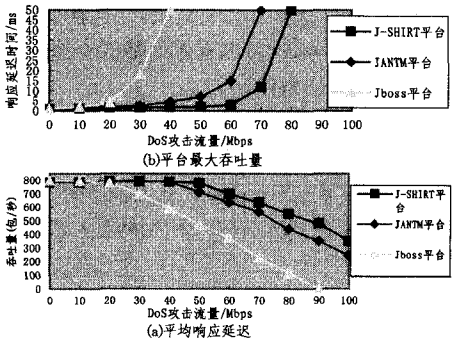

图 4 DoS 攻击下三种应用服务器平台的响应延迟时间和吞吐量的结果对比图

另外，从上述两图中可以看出，在未发生 DoS 攻击或者 DoS 攻击较弱时，本文所设计的 J-SHIRT 平台和 JANTM 平台，比起商用的 Jboss 应用服务器平台来说，其最大吞吐量较低，平均相应时间较慢。这是因为 J-SHIRT 平台和 JANTM 平台中的表决算法和容侵自治愈本身具有一定的通信复杂度，减缓了平台的响应速度。但是从上图中也可以看出，它们的通信复杂度对平台整体性能的影响不大。

以上测试实验结果说明，本文设计的 J-SHIRT 平台会增加大约 2ms 的延迟时间，并且在一个适度的网络负载下不会发生最大吞吐量下降的情况。同时也可以看出，当 DoS 攻击流量大于 60Mbps 之后，本文设计的平台最大吞吐量和平均响应延迟时间也会大幅增加。但是，总体来看，相对于其他两个测试平台，本文设计的原型平台具有更高的可靠性和生存性。

3.3 软件老化测试

本测试实验将对 J-SHIRT 平台和 JANTM 平台进行对比测试，以验证本文设计的平台是否具有预防软件老化现象发生的能力，同时也可以证明所设计的平台具有更高的可靠性和生存性。

软件老化测试实验所需的测试时间较长，通过调查研究，本次测试实验将软件老化测试时间定为 48 小时。在这 48 小时之内，客户端将向实验平台源源不断地发送数据包，平均每秒 800 个包，每个数据包大小为 2048bytes，即为平台的最高合法流量。

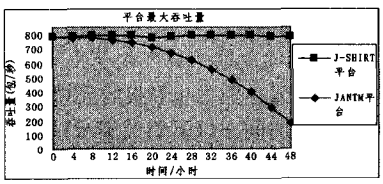

图 5 不同应用服务器平台的软件老化测试结果对比图

下面将对 J-SHIRT 平台和 JANTM 平台分别进行吞吐量对比测试。由于软件老化测试实验周期较长，以下测试实验结果只是通过 5 次反复实验测试所得，图 5 分别显示了 J-SHIRT 原型平台和 JANTM 原型平台的吞吐量的测试结果。

从图 5 中可以看出，JANTM 平台由于经过长时间不间断的运行，各应用服务器系统的资源不断消耗，资源碎片大量增加，数据污染以及数值累积错误也同时出现。最终，随着时间的推移，整个应用服务器平台不断老化，导致性能衰退，安全性降低，平台吞吐量由最初的 785 包/秒下降到 48 小时后的 180 包/秒，如果继续运行，原型平台甚至有可能面临崩溃。而本文所设计的平台针对软件老化现象有着良好的抗衰特性，在 48 小时的实验测试时间内，虽然面对高负载流量，但是依然能够保持较高且稳定的吞吐量，能够从容地面对用户的业务请求和经常发生的网络入侵，体现了本文设计的平台的高可靠性和生存性特征。

结束语 随着 J2EE 应用服务器的广泛应用，企业也对 J2EE 应用服务器系统服务的可生存性和可靠性要求越来越高，单纯地使用一种安全技术很难防范所有的攻击，本文提出一种基于 J2EE 应用服务器的容侵自治愈方法，它有效地针对容忍入侵状态中的未知状态，进行周期性恢复，使每个应用服务器防止了隐性入侵的发生，也避免了软件老化现象的出现；同时又针对具有一定连续性和成功性的入侵，引入了反应式恢复，以确保应用服务器集群能满足容忍入侵表决机制的前提条件。该设计和实现方法满足了用户对应用服务器的高可靠性和高生存性的要求。今后的工作包括：针对不同入侵和故障设计具体的自治愈算法；研究自治愈管理器中的周期性恢复时间；构建一个可信的自治愈管理器。

参 考 文 献

[1] 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[P]. 国家技术发明专利，申请号 200710019118.9. 2007.11.20

[2] 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[J]. 信息安全与通信保密，2005(7)：337-341

[3] Huang Y，Kintala C，Koletis N，et al. Software Rejuvenation：Analysis，Module and Applications[C]// Proc. 25th IEEE on Fault Tolerant Computing. CA：Los Alamitos，1995：381-390

[4] Scott S. JBoss Administration and Development(Third Edition)[M]. JBoss Group，LLC，2003

[5] 袁顺，刘伟，彭亮. 容忍入侵的 J2EE 应用服务器设计与实现[J]. 计算机工程，2008，34(19)：137-140

[6] 李庆华，张胤，赵蜂. 一种基于 CORBA 分布式对象的容侵恢复策略[J]. 计算机工程，2006，32(20)：138-140

[7] 周华，孟相如，张立，等. 分布式入侵容忍系统的主动恢复算法研究[J]. 西安电子科技大学学报：自然科学版，2009，36(2)：378-384

[8] Sousa P，Bessani A N，Dantas W S，et al. Intrusion-Tolerant Self-Healing Devices for Critical Infrastructure Protection[C]// Proceedings of the 39th IEEE International. 2009：374-380

(上接第 27 页)

[6] RealNetworks，Inc.：RealSystem Media Commerce Suite Technical White Paper[S]. http://www.realnetworks.com

[7] 庄超. 一种新型的 Internet 内容版权保护的计算机制[J]. 计算机学报，2000，23(10)：1088-1091

[8] 马兆丰，冯博琴. 基于动态许可证的信任版权安全认证协议[J]. 软件学报，2004，15(1)：131-140

[9] 俞银燕，汤帜. 数字版权保护技术研究综述计[J]. 计算机学报，2005，28(12)：1957-968

[10] Microsoft Media Rights Server. Microsoft Corp [S]. http://drmlicense.one.microsoft.com/indivsite/en/indivit.asp

[11] 郑晓琳，荆继武. 一种基于 PKI 的 DRM 系统[J]. 计算机工程与应用，2006，4：128-131

[12] 李平，卢正鼎，等. 一个面向家庭网络的数字版权管理系统[J]. 计算机科学，2009，36(11)：116-119