基于移动 Agent 的 DDoS 攻击综合防御模型

陶利民
TAO Li-min

杭州师范大学 信息科学与工程学院,杭州 310036
Information Science & Engineering School,Hangzhou Normal University,Hangzhou 310036,China
E-mail:TLM5460@163.com

TAO Li-min.Integrated model for defending against DDoS attacks based on mobile Agent.Computer Engineering and Applications,2008,44(4):147-150.

Abstract: The attack of Distributed Denial of Service(DDoS) has become one of the most harmful problems in network security. There are some limitations in traditional defendant for using the single method.This paper proposes an initiative defence model which adopts mobile agent and integrate multi-defense methods.The elements of the mobile agent in defense model are designed in detail to solve the limitation of the single defense method,passive defense and high false alarm.The model is robust and scalable.

Key words: mobile agent;Distributed Denial of Service(DDoS);detection;defense

摘 要:分布式拒绝服务(DDoS)攻击已成为网络最大的安全威胁之一,传统检测防御方法由于多采用单一防范措施很难对其彻底防范。利用移动 Agent 特性,在设计上整合多种防御方法,构造了一种综合的主动检测防御模型,并详细设计了模型中移动 Agent 的组成元件,力求解决存在的单点失效、被动防御等问题,使得该模型具有良好的健壮性和可扩展性。

关键词:移动代理;分布式拒绝服务攻击(DDoS);检测;防御
文章编号:1002-8331(2008)04-0147-04 **文献标识码**:A **中图分类号**:TP393.08

Internet 的普及为世界范围内实现资源和信息的共享提供了便利条件,但随之而来的信息安全问题日益成为社会关注的焦点。在网络入侵攻击方面,无论是数量、手段、性质,还是规模,已经到了令人震惊的地步。根据 CNCERT/CC(国家计算机网络应急技术处理协调中心)网络安全工作报告[1],分布式拒绝服务攻击(DDoS)从技术实现上而言开始通过成群受控主机进行分布式高强度攻击;同时产生非常随机的源 IP 地址;攻击数据包结构形式随机变化;利用网络协议缺陷与系统漏洞;攻击特征更不明显等。

由于 DDoS 攻击分布式特点,进行防御时理想方法是采用分布式防御策略,在尽量靠近攻击源处进行防御。分布式解决方案主要思想是当受害主机或网络检测到 DDoS 攻击并对其进行标识后,沿着攻击流量到达的多条路径反向追踪,在靠近攻击源的各个路由器/主机截获攻击流量并丢弃,从而扩大了防御范围,降低 DDoS 攻击的危害。

对传统的 DDoS 检测防御方法进行研究和分析后发现,单一的防范措施很难彻底防范 DDoS 攻击,一个完善的 DDoS 攻击防御方案必将是一个整合了预防、检测、响应等多种方法的综合机制[2]。针对上述问题,本文利用移动 Agent 的智能、移动、协作等特性,依托于路由设备和 ISP 域关键主机,在设计上整合监控、过滤、追踪等多种防御方法,提出了一种基于移动 Agent 的 DDoS 综合检测防御模型,力求解决当前检测防御系统存在的单点失效、被动防御等问题,使得检测防御系统具有更强的健壮性和良好可扩展性。

1 移动 Agent 技术

移动 Agent 是一个独立运行的软件实体,可以代表用户完成特定任务。它可以简明地定义为:包含代码、数据以及执行语境的软件包。所有的移动 Agent 系统都包括如下两部分:移动 Agent 服务设施和移动 Agent。移动 Agent 服务设施实现 Agent 在主机间的转移,并为其分配执行环境和服务接口。Agent 在服务设施中执行,通过 Agent 通信语言 ACL(Agent Communication Language)相互通信并访问服务设施提供的服务[3]。

从系统的角度看,一个移动 Agent 应包括代码、状态和属性三部分。代码用来实现移动 Agent 的各项功能。为了保证状态的连续性,移动 Agent 必须保存移动之前的一些状态信息。属性描述了移动 Agent 本身的特性,包括:Agent 标识符、所有者、起始点地址以及起始时间等;此外,移动 Agent 还必须带有认证信息,认证信息用来作为确认用户的凭证。

利用 Agent 的移动特性,可以将攻击检测及防御功能封装在 Agent 实体中,并分派到边界路由器、网关、服务器等关键网络节点上。在这些节点上过滤、处理、分析所采集到的可疑数据和异常行为,同时位于不同节点上的 Agent 可以交换信息,达到协同防御 DDoS 攻击的目的。由于 Agent 可以自由地迁移到

作者简介:陶利民(1975-),男,硕士,主要研究领域:计算机网络安全,移动 Agent。
收稿日期:2007-05-31 **修回日期**:2007-08-28

网络的各个节点,这样可以合理地均衡网络负载,并且更靠近攻击源,经过配置的 Agent 可以分布在网络的各部位完成检测防御工作,保证在攻击发生的最初阶段进行检测防御工作,有效克服了网络延迟,从而确保防御系统具有很好的实用性。

2 基于移动 Agent 的主动检测防御模型

本文参考 Stone 给出的体系结构[4],将网络划分为不同 ISP 域,用户连接在这些 ISP 域上。将路由器分两类:内部路由器和外部路由器。对某 ISP 域而言,内部路由器指该 ISP 域内的路由器,外部路由器指属于另一个 ISP 域的路由器。在单个 ISP 域内内部路由器又分为边界路由器和交换路由器。边界路由器与一个或多个外部路由器相邻;交换路由器仅与其它内部路由器相邻。假设网络拓扑中的路由器(关键主机)都存在移动 Agent 执行环境或拥有足够的资源来为移动 Agent 提供服务。

在每个 ISP 域中,基于移动 Agent 的主动检测防御模型主要由域内防御节点和边界防御节点两部分组成。每个 ISP 域内在某个时间段内仅存在一个域内防御节点,一般情况下在任意一个内部路由器或专用主机上实现;边界防御节点在 ISP 域的所有边界路由器或专用主机上实现。如果节点受到攻击已经无法执行任务,可利用移动 Agent 寻找合适的主机或路由器来替代该节点,执行完成相应的任务。

2.1 节点结构模型

基于移动 Agent 主动检测防御模型节点结构如图 1 所示。根据前文假设,路由器/关键主机拥有足够的资源和条件来搭建移动 Agent 的执行环境。移动 Agent 执行环境是支持 Agent 系统的关键部分。移动 Agent 的交互行为以及对本地资源的访问均由它来控制和支持,它主要包括认证服务、资源管理、通信模块几部分。

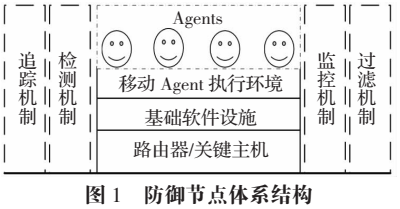

图 1 防御节点体系结构

目前有关 DDoS 攻击检测和防御方法的研究成果很多,例如过滤、监控、追踪、异常流量检测、数据包标记等,每种方法都有它的特点和应用局限。而在实际应用中,对 DDoS 攻击采取单一的技术难以做到全面有效的防御,需要不断更新综合防御体系[2]。本文提出的主动检测防御体系,对防御节点使用移动 Agent 技术结合现有监控、过滤、追踪、检测等技术完成 Ingress/Egress 流量监控和过滤,对 DDoS 攻击进行检测和追踪。再通过防御节点间的协同构筑 DDoS 攻击的主动检测防御体系。

2.2 节点防御方法集成

下面详细介绍边界节点的防御机制。

(1)流量监控:边界防御节点利用 Habib 提出的测量 SLA(Service Level Agreement)的三个参数:延迟、丢包率及吞吐量,以这三个参数改变值来进行分布式网络监控[5]。当延迟、丢包率及带宽消耗量同时超过门限,监控器报告 SLA 异常。结合拥塞控制和过滤机制对 Ingress/Egress 流量进行处理。

(2)过滤机制:过滤机制依据规则对 IP 包进行过滤,对 Ingress 流量,只允许源地址不属于域内的报文。对 Egress 只允许源地址属于域内的报文流出。过滤机制也可根据流量监控情况对异常流量进行过滤,阻止瞬间暴增的数据包。当边界防御节点接收到来自内部防御节点的任务 Agent 时,解析 Agent 任务并执行,同时将规则添加到规则库中。对于匹配规则的数据包,启动追踪机制和检测机制,对攻击进行过滤和协同防御。

(3)追踪机制:采用 Barros 提出的 ICMP traceback 的改进方法来追踪攻击源[6],路由器向正被处理的数据包源地址发送 ICMP 消息。根据收到的 ICMP 追踪数据包,构造出完整攻击路径,追踪 Agent 依据此攻击路径反向追踪,在靠近攻击源节点防御 DDoS 攻击。

(4)检测机制:将恶意流量或数据包的监控参数、过滤参数、来源 IP 地址封装在检测 Agent 中,而后检测 Agent 巡游 ISP 域各边界防御节点,达到单 ISP 域协同检测防御目的。也可根据恶意流量目的地址(对 Egress 流量)发往目的 ISP 域的边界防御节点,将相关参数存入该边界防御节点的规则库,达到多 ISP 域之间的协同防御。

(5)备份机制:备份 Agent 在边界防御节点受到攻击不能正常工作时,将所携带的资料存入替代受攻击节点的路由器或关键主机的规则库中,保证防御的持续性和鲁棒性。

3 模型中的移动 Agent 设计

3.1 移动 Agent 组成

移动 Agent 除了包含执行代码段、数据段及执行状态段外,还需要携带许多信息作为执行任务时的参考资料,本文在 Cubalesba[7]等人定义的移动 Agent 元件的基础上进行修改和扩展,给出相关移动 Agent 的元件组成:

(1)追踪 Agent(Tracing Agent):追踪 DDoS 攻击行程中,在某个 ISP 域边界防御节点 $Dnode_j$ 或域内关键主机/路由器 j 上执行检测和防御后的状态:

$$T\_Agent^j=(Owner,ID,Goal/Result,bc,uid^j,r,vc^{\#(Dnode_j)},Life\_T,States)$$

(2)检测 Agent(Detecting Agent):在边界防御节点 $Dnode_j$ 上执行检测和协同防御后状态:

$$D\_Agent^j=(Owner,ID,bc,Md^j,voc,uid^j,r,vc^{\#(Dnode_j)},Life\_T,States)$$

(3)任务 Agent(Assignment Agent):派发到边界防御节点 $Dnode_j$ 或域内路由器/关键主机 j 上执行任务后的状态:

$$A\_Agent^j=(Owner,ID,Goal/Result,bc,uid^j,r,Life\_T,States)$$

(4)备份 Agent(Backup Agent):在防御节点 $Dnode_j$ 上执行备份后的状态:

$$B\_Agent^j=(Owner,ID,bc,Md^j,rid_j,voc,Life\_T,States)$$

其中,*Owner*:Agent 的拥有者,记录移动 Agent 拥有者的数字签名、路由器/主机名称与 IP 地址等;

ID:标识移动 Agent,若移动 Agent 没有合法标识,则会被防御节点或其他路由器/主机拒绝服务;

Goal/Result:目标/结果,包含移动 Agent 预定完成的任务和完成任务后的结果;

bc:移动 Agent 执行的二进制代码;

r:$r=(Dnode_1,Dnode_2,\cdots,Dnode_j,\cdots,Dnode_n)$ 描述移动 Agent 的迁移路径,包含完成任务所必须造访的防御节点;

vc^j:描述已经访问过的防御节点序列,j 为目前为止访问过的防御节点数量。移动Agent 在始发防御节点(Host)迁移前,

$vc^{\#(Dnode_h)}=vc^0$,当它开始按照迁移路径 r 开始迁移时,在释放移动 Agent 前,经过防御节点 $Dnode_j$ 产生 $vc^{\#(Dnode_1)}=vc^1=Dnode_1$ 序列。

Life_T:*LifeTime*:生存时间,依据任务不同,移动 Agent 的生存时间有长有短,移动 Agent 拥有者派遣移动 Agent 时加入时间戳,以告知其它节点此移动 Agent 的生存时间,否则就可能在网络中存在许多已完成任务或失去效用的 Agent,消耗网络资源。一般 *Life_T* 可设为移动 Agent 的迁移次数,而移动 Agent 每迁移一次,*Life_T* 值减去 1。

States:移动 Agent 的状态;当移动 Agent 从一个防御节点迁移到下一个防御节点时都会记录下执行状态,移动 Agent 的内部状态包括读写的记录、最后执行动作等。

Md^j:移动 *Agent* 在防御节点 $Dnode_j$ 上执行任务所累计收集的资料(*Md*)。利用它来判断网络流量是否正常。$Md^j=(md^j, fd^j, td^j)$,md^j 是防御节点 $Dnode^j$ 上的监控值,fd^j 为防御节点 $Dnode^j$ 上过滤值,td^j 为 $Dnode_j$ 上过滤的恶意流量的源地址或恶意流量特征。

voc:在一个移动 Agent 的迁移行程中所进行的计算可以分为独立和非独立两种。利用访问节点限制(visiting node constraints)*voc* 来描述。$voc=(voc_1, voc_2, \cdots, voc_n)$,$n$ 是 r 中所列出的移动 Agent 迁移行程上的节点数量。若 $voc_j=0$,则移动 Agent 在防御节点 $Dnode_j$ 上执行任务时不需要其它节点资料。若 $voc_j=\{Dnode_m, Dnode_n\}$,$j,m,n=1,2,\cdots$,则移动 Agent 在防御节点 $Dnode_j$ 上执行任务时须参考 $Dnode_m$ 和 $Dnode_n$ 的资料,属于非独立计算。

uid^j 防御节点 $Dnode_j$ 产生回复确认,当 $Agent^j$ 迁移到下一个防御节点 $Dnode_{j+1}$ 时,给节点 $Dnode_j$ 发回 uid^{j+1} 以确认正确迁移。

rid_j 当防御节点 $Dnode_j$ 被攻击经过重新启动后,发送恢复确认 rid_j 给它的替代节点,告知替代者,防御节点 $Dnode_j$ 已经回复正常工作,结束替代者的备份功能。

3.2 追踪防御方法

利用 ICMP 追踪法,加载了跟踪机制的边界防御节点,以很低概率 p(默认为 1/20 000)发送对数据包的一个特殊形式拷贝,该拷贝是一种特殊定义的 ICMP 数据包,每一个 ICMP 追踪数据包都包括部分路径的信息。若收到足够多的 ICMP 追踪数据包,就可以构造出完整攻击路径。边界防御节点将将攻击路径封装在追踪 Agent(*T_Agent*)中作为追踪路径。

假设防御节点 $Dnode_h$ 已通过上述 ICMP 追踪方法构造出完整攻击路径。则 $Dnode_h$ 送出 T_Agent^j 给攻击路径上的节点,按攻击路径(迁移路径)$r=(Dnode_1, Dnode_2, \cdots, Dnode_n)$行走,在路径 r 上的节点接收到该 Agent 后,从 T_Agent^j 获取防御任务和迁移路径,依据 $vc^{\#(Dnode_i)}$ 和 *Life_T* 来决定是否将该 Agent 继续发送到路径 r 的下个节点,转发 Agent 前在 $vc^{\#(Dnode_i)}$ 中添加本节点标识,并将 *Life_T* 减 1,$vc^{\#(Dnode_i)}$ 和 *Life_T* 会随移动 Agent 迁移路径上不同的节点而改变。迁移路径上的节点从追踪 Agent 获取防御任务,并将 Agent 携带的攻击签名或流量特征加入本地规则库,供过滤机制参考。

上述追踪防御过程中,追踪 Agent 从一个节点迁移到另一个节点的行为主要包括两部分:节点间追踪 Agent 的"发送"和"接收"。分别描述如下:

追踪 Agent(T_Agent^j)的发送算法:

(1)Select next node from r
(2)Store a copy of T_Agent^j
(3)If ($Life\_T>0$){
(4) Send T_Agent^j to the selected node by step(1)
(5) Wait for confirmation uid^{j+1} until time_out
(6) If (confirmation received and valid)
Release the copy of T_Agent^j
(7) else If (not received uid^{j+1} or time_out)
Goto Step(1)
(8)} else
Release the copy of T_Agent^j
(9)end

算法首先从 T_Agent^j 的迁移路径 r 中选择下一个要迁移的节点,备份 T_Agent^j,若 $Life\_T>0$,发送 T_Agent^j 到下一个要迁移的节点,并等待下一个节点的确认;若收到下一个节点有效确认,则释放 T_Agent^j 的备份,算法结束,若在规定时间内没收到确认,则算法跳转回第一步继续运行,若 $Life\_T=0$,则释放 T_Agent^j 的备份,算法结束。

追踪 Agent(T_Agent^j)的接收算法:

(1)Receive T_Agent^j
(2)Check the *ID* of T_Agent^j
(3)If (valid *ID*){
(4) Create and Send confirmation uid^{j+1};
(5) Download Goal/Result form T_Agent^j to local database;
(6) Execute of T_Agent^j;
(7) $vc^{\#(Dnode_{i+1})}=vc^{\#(Dnode_i)}+Dnode_{j+1}$;
(8) $Life\_T-1$;
(9) If ($Life\_T-1>0$)
Form and Send T_Agent^{j+1}
(10) else
Release T_Agent^j
(11)} else
Drop T_Agent^j
(12)end

节点 $Dnode_{j+1}$ 接收到 T_Agent^j,首先检查是否拥有合法 *ID*,若不合法则直接丢弃。若合法则发送确认码 uid^{j+1};从 T_Agent^j 读取防御任务(包含攻击签名或流量特征),执行 T_Agent^j 携带的二进制代码,之后将本节点加入到已访问节点集合 $vc^{\#(Dnode_i)}$ 中,计算 $Life\_T-1$,若 $Life\_T-1>0$ 则继续发送 T_Agent^{j+1} 到下一个节点,否则,释放 T_Agent^j。

3.3 过滤防御方法

受害者一旦发现攻击,就会请求该域的域内防御节点对上游的攻击进行防御。假设域内防御节点和受害者之间已经建立经过认证的可靠通信,若进行非本地防御,则派发任务 Agent 到边界防御节点,在靠攻击源头处进行过滤。工作流程如下:假设域内防御节点 $Dnode_c$ 接收到来自某个受害主机 V_i 的防御请求,通过解析决策,向 n 个防御节点 $Dnode_1, Dnode_2, \cdots, Dnode_n$ 分别派出 n 个任务 Agent:$A\_Agent^1, A\_Agent^2, \cdots, A\_Agent^n$。$Dnode_c$ 分别为这 n 个防御节点分配唯一的标识符,作为任务 Agent 的 *ID*。$Dnode_c$ 维护 $Dnode_1, Dnode_2, \cdots, Dnode_n$ 和分配给它们的任务 Agent *ID* 号相关联的数据库。同时为本次防御任务(请求)随机产生一个 *ID* 号,作为所有此次派发的任务 Agent 的 *Owner*。攻击签名和受害主机 V_i 的相关信息保存在任务 Ag

ent 的 *Goal/Result* 字段中；因为每个任务 Agent 迁移的目标节点是唯一的，所以迁移路径 *r* 中仅有和该任务 Agent *ID* 相匹配的防御节点。*Life_T* 为常量 1。表明任务 Agent 从域内防御节点出发仅需迁移 1 次。收到任务 A_Agent^j 的防御节点 $Dnode_j$ 都会给域内防御节点 $Dnode_c$ 发回确认 uid^j。激活报文标记功能，使用 A_Agent^j 的 *ID* 号标记去往 V_i 的数据包。

3.4 节点备份方法

防御节点维护一个和它直接相连的邻居防御节点列表，当防御节点遭受攻击，各项监控值超过门限值时，生成备份 Agent，将本地防御状态封装在 Agent 中发送给邻居防御节点，防御节点处于被攻击期间，会定时发送备份 Agent 给邻居防御节点，若防御节点一直能够正常工作，在攻击结束，各项监控值恢复正常值时，或者被攻陷的防御节点重新启动恢复功能之后，都会发送恢复码 *rid* 给邻居防御节点，结束备份。邻居防御节点在接收到备份 Agent 时，先将此 Agent 携带的防御状态信息存入本地数据库，若超过定时设置没有收到更新，则载入最新保存的防御节点的监控信息，启动防御功能，替代该防御节点，由于是该防御节点的所有邻居防御节点共同完成替代防御任务，因此这个备份在功能上是完全的。当收到恢复码 *rid* 时，邻居防御节点结束备份，恢复原来设定值。

被攻击防御节点（B_Agent^j）发送备份 Agent 给邻居防御节点的算法：

(1)Select $Dnode_j$ from Neighbor_Dnode_List
(2)$r=Dnode_j$
(3)Store a copy of B_Agent^j
(4)Send B_Agent^j to the selected node $Dnode_j$
(5)Release the copy of B_Agent^j
(6)end

邻居防御节点（B_Agent^j）接收备份 Agent 的算法：

(1)Receive B_Agent^j
(2)Check the *ID* of B_Agent^j
(3)If(valid *ID*){
(4)Download Md^j form B_Agent^j to local database;
(5)Execute of B_Agent^j;
(6)*Life_T*-1;
Release B_Agent^j
(7)} else
Drop B_Agent^j
(8)If (not received next B_Agent^j or rid^j within UpdateTime)
//如果在更新时间间隔内没有收到新的备份 Agent 或恢复码
(9)Backup with the latest Md^j
(10)else If (a new update *B_Agent* arrive)
(11) goto step(1)
(12)else If (receive a rid_j) //如果收到恢复码立即结束备份工作
(13) Backup application terminate
(14)end

3.5 模型中移动 Agent 的原型

为了验证将移动 Agent 引入本文提出的主动检测防御系统的可行性，使用 JDK1.6+Aglets2.0.2 来实现防御机制中的移动 Agent 的原型。设计原型系统包括下面组成：移动 Agent、Agent 守护进程、监视器（monitor）三部分组成，其中移动 Agent 包括追踪 Agent、检测 Agent、任务 Agent 和备份 Agent。

（1）移动 Agent：基于移动 Agent 的主动检测防御模型中主要包括追踪 Agent、检测 Agent、任务 Agent 和备份 Agent 四种，是用 Java 语言编写的独立运行的软件实体。它们代表监视器或用户在网络中巡游并执行追踪、协同、检测、派发任务和节点备份等任务。

（2）Agent 守护进程：Agent 守护进程是整个系统的基础。在每个防御节点上必须有一个运行于 Java 虚拟机（JVM）上的代理守护程序。Agent 守护进程为移动 Agent 的传输和执行提供了必要的机制和环境，可以使用 IBM 日本公司开发的 Aglets2.0.2。

（3）监视器（monitor）：监视器主要负责管理节点上的移动 Agent，进行数据管理，依据本地知识库使用 Agent 守护进程提供的功能产生各种移动 Agent 完成追踪、检测、任务派发和备份等功能。依托于本地知识库，监视器提供了一个更高级的视图，有助于整合和协同各种 Agent 完成防御任务。

原型系统中移动 Agent 之间的通信，可用消息传递的方式来传递消息对象。由于移动 Agent 所在位置透明化，Agent 想要与远端 Agent 沟通时，只在本地主机的上下文环境中产生对应远端的 Agent 的代理，并与此代理沟通即可，不必直接处理网络连接和通信的问题。

使用 Aglet Workbench 所提供的可视化 Agent 管理界面 Tahiti，可以方便地监视和控制 Aglet 的执行。

4 结论

本文提出的基于移动 Agent 的主动防御模型整合了多种攻击检测防御方法，并可以使用移动 Agent 在边界防御节点之间实现知识共享和协同检测防御，使其具有很大的灵活性。该模型本身也具备抗 DDoS 攻击的特性，攻击者很难通过瘫痪其中某个结点来破坏整个防御体系。在实际应用上，由于防御模型是基于路由器的，可在骨干网络已有设备的基础上实现，只需配置软件（假设路由器具备足够的资源运行软件），而无需增加硬件设备，在一定程度上可以节约成本。但该模型的部署难度随着 ISP 域的大小，路由器的多少而变化，完全由人工部署将不可想象，未来可以考虑使用移动 Agent 技术实现参数的动态配置和软件的重编程能力。

参考文献：

[1] CNCERT/CC 2005 年上半年网络安全工作报告[EB/OL].http://www.cert.org.cn/upload/2005CNCERTCCAnnualReport.pdf.
[2] Chang R K C.Defending against flooding-based distributed denial-of-service attack:a tutorial[J].IEEE Communications Magazine,2002,40(10):42-51.
[3] 张云勇.移动 Agent 及其应用[M].北京:清华大学出版社,2002:17-18.
[4] Stone R.Center Track:an IP overlay network for tracking DoS floods[C]//Proceedings of Nineth Usenix Security Symposium,August,2000.
[5] Habib A,Fahmy S,Bhargava B.On monitoring and controlling QoS network domains[C]//ACM Computer Communication Review,Aug,2004.
[6] Barros C.A proposal for ICMP traceback message.Internet Draft [EB/OL].http://www.research.att.com/lists/ietf-itrace/2000/09/msg00044.html.
[7] Cubaleska B,Schneider M.Detecting Dos attacks in mobile agent systems and using trust policies for their prevention[C]//The 6th World Multi conference on Systemics,Cybernetics and Informatics SCI,2002.