移动 Agent 安全机制研究

The Research on Security Mechanisms of Mobile Agent

罗 飞 邱玉辉

(西南师范大学计算机与信息科学学院 重庆 400715)

Abstract The key elements of the security mechanisms of mobile agent are: (a)the host against potentially hostile actions of mobile agent; (b)the mobile code against tampering attempts by the malicious host. Many techniques for the first problem have been developed. The second problem seems to be more difficult. In recent years, several methods have been discussed for protecting mobile agent against malicious host. In this paper, the intentions of mobile agent are protected, so the hosts are disturbed and unable to tamper the mobile agent.

Keywords Mobile agent, Security, Malicious host, Intention spreading, Intention shrinking

1. 引言

近年来,移动 agent 技术已成为计算机网络和分布式系统最具活力的发展方向,同时移动 agent 系统也为分布式应用的开发提供了便利。

现在,为用户工作站提供各种信息服务的方式以客户/服务器方式为主;将来,移动 agent 系统将成为主要的信息提供者,它能动态地为 agent 和网络服务提供支持。

从现在的客户/服务器环境到将来的移动 agent 系统的转变是一个重大的进步,但这种转变存在的最大障碍就是:移动 agent 和其执行状态在执行环境中移动时的安全问题。

安全性是移动 agent 系统中最重要的问题之一,是移动 agent 系统被广泛接受的关键。它也是最复杂的问题,因为它把传统的大型分布计算安全问题和多用户操作系统的安全问题集中到了一起。移动 agent 系统出现的安全问题是对计算机安全系统的设计和实现提出的新挑战。更准确地说,这需要用以提供访问控制和通讯安全服务的程序和程序段的设计技术和机制来满足移动性的要求。

本文主要讨论如何保证移动 agent 的信息及运行的真实性。

2. 移动 agent 的信息熵

2.1 假设

· 移动 agents 都是模块化的,或它们的一部分可认为是模块化的。

· 每一模块表示一个任务,并将这些模块分为三类:

1. 公有类:所有意图都公有的模块。
2. 共享类:仅被某些意图的子集所共享的模块。
3. 私有类:仅被一个意图所拥有的模块。

· 移动 agents 和远程主机相互均不信任。

· 每一个移动 agent 仅在同一个远程主机上运行一次。运行完成之后,agent 被更新,并增加了在这个主机上收集的信息。然后 agent 移动到另一个主机上。

2.2 移动 agent 的信息熵

1948 年 Shannon 提出并发展了信息论,研究以数学的方法度量并研究信息。熵是信息论中描述事物不确定性的程度的一个概念。如果一个随机变量 x 只取 a 与 b 两个不同的值,比较下面两种情况:

$$p(x=a)=0.98, p(x=b)=0.02 \quad (1)$$

$$p(x=a)=0.48, p(x=b)=0.52 \quad (2)$$

很明显,(1)的不确定性要比(2)的不确定性小得多。

定义 1 移动 agent 的意图用随机变量 x 表示。令 X 表示由移动 agent 所有可能意图组成的集合,$X=\{x_1, x_2, \cdots, x_n\}$,其中 $x_1, x_2, \cdots, x_n$ 是 agent 的 n 个意图。任意 $x_i \in X (1 \leqslant i \leqslant n)$,$x_i$ 发生的概率由 $p(x_i)$ 表示,则 $\sum_{1 \leqslant i \leqslant n} p(x_i)=1$。

对于任意移动 agent,首先从概率的角度来确定它的意图。假设移动 agent 可分为 K 个模块 $M_1, M_2, \cdots, M_k$,其中,$M_i \subseteq X, 1 \leqslant i \leqslant k$。

要得到意图 x_i 在移动 agent 中出现的概率,则需要计算这 K 个模块中的包含意图 x_i 的模块数量。

如果所有的 K 个模块都仅包含意图 x_i,即全是私有类,则意图 x_i 出现的概率为 1,即意图 x_i 必然出现。如果 K 个模块都是公有类,即共同属于所有意图,那么每一个意图出现的概率都是相等的。如果 K 个模块中的一部分是共享类,则在意图 x_1 到 x_n 中的某些意图出现的概率较高。

一般来说,恶意主机并不是对任何移动 agent 都进行相同的攻击,它对某些移动 agent 的攻击频率总是比较高一些。例如,收集价格信息的移动 agent 通常比收集学术作品信息的移动 agent 容易受攻击。因此,知道移动 agent 的任务是恶意主机的最终目的。主机越早知道移动 agent 的意图,移动 agent 就越不安全。

定义 2 设 x_i 表示移动 agent 的意图,$p(x_i)$ 表示意图 x_i 出现的概率,则 x_i 出现的不确定性为 $H(x_i)=-\ln(p(x_i))$,$H(x_i)$ 称为意图 x_i 的熵值。移动 agent 的熵值 H,即集合 $\{x_1, x_2, \cdots, x_n\}$ 的平均熵值为 $H=-\sum_{1 \leqslant i \leqslant n} p(x_i)\ln(p(x_i))$。

定理 1 设随机变量 x 只取有限个值 $x_1, x_2, \cdots, x_n$,相应的概率记为 $p_1, p_2, \cdots, p_n$,则集合 $\{x_1, x_2, \cdots, x_n\}$ 的平均熵值 H 取得最大值的充分必要条件是:$p_1=p_2=\cdots=p_n=1/n$。

证明:充分性:考虑 $G(p_1, p_2, \cdots, p_n)=-\sum_{1 \leqslant i \leqslant n} p_i \ln p_i+\lambda(-\ln p_i-1)$,为求其最大值,将 G 对 p_i 求偏微商,并令偏微商为 0,得方程:$0=\frac{\partial G}{\partial p_i}=-\ln p_i-1+\lambda (i=1, 2, \cdots, n)$。求得 $p_1=p_2=\cdots=p_n$。又因为 $\sum_{1 \leqslant i \leqslant n} p(x_i)=1$,所以 $p_1=p_2=\cdots=p_n=1/n$。此时相应的熵值 $H_{max}=\ln N$。

必要性：当 $p_1=p_2=\cdots=p_n$ 时，$G(p_1,p_2,\cdots,p_n)$取得最大值。

熵 H 表示了移动 agent 所包含的信息量。熵值越高，移动 agent 所能提供的信息也越多。

假设，移动 agent 最初只包含一个意图，则熵值为 0。随着熵值的增加，主机得到了更多的信息，那么主机获得 agent 的最初意图也随之变得困难。

假设移动 agent 最初带有一些意图，它的熵值小于等于最大熵值。当熵值减少时，主机得到的信息也随之减少，那么主机要知道 agent 的最初意图也同样变得困难。

因此，可用以下方法来保护移动 agent 的原始意图被恶意主机发现：

- 当原始意图是集中的时候，用意图扩展法来增加熵值。
- 当原始意图是分散的时候，用意图收缩法来减少熵值。

3. 基于意图的移动 Agent 保护方案

3.1 意图扩展法

使用插入法，在移动 agent 中插入干扰代码，实现意图的扩展。

假设一个移动 agent 的意图由集合 X 表示。在 X 中插入干扰码，得到另一意图集 Z。$Z=\{z_1,z_2,\cdots,z_n\}$，n 为正整数。

算法 1 意图扩展算法

(1)令 a=1，b=1。

(2)移动 agent 模块化。将移动 agent 划分为 K 个模块 $M_1,M_2,\cdots,M_k$，其中 $M_i\subseteq X$，$1\leqslant i\leqslant k$。

(3)给定算法结束时要达到的熵值 H_{end}，且 $0\leqslant H_{end}\leqslant \ln N$。

(4)计算意图 x_i 的概率 $p(x_i)=\frac{t_i}{k}/\sum_{1\leqslant i\leqslant n}\frac{t_i}{k}$，其中 t_i 为包含意图 x_i 的模块数量。并将意图按概率的降序排序，得意图序列 $x'_1,x'_2,\cdots,x'_n$。

(5)计算熵值 $H=-\sum_{1\leqslant i\leqslant n}p(x_i)\ln(p(x_i))$。

(6)若熵值 $H<H_{end}$，转第(7)步；若 $H\geqslant H_{end}$，或 a+b>n 时，结束。

(7)令 h=a，$l=n-b+1$，计算意图 x'_h 与意图 x'_l 的相似度 $Sim(x'_h,x'_l)=\frac{1}{1+d(x'_h,x'_l)}=\frac{1}{1+d_{hl}}$，其中 $d_{hl}=x'_h-x'_l$。

(8)若 $Sim(x'_h,x'_l)\geqslant 0.5$ 时，将意图 x'_l 加入到满足下列条件的模块 M_i($1\leqslant i\leqslant k$)中，$x'_l\notin M_i$ 且 $x'_h\in M_i$，令 a=a+1，b=1，转第(4)步；若 $Sim(x'_h,x'_l)<0.5$ 时，令 b=b+1，转第(7)步。

当移动 agent 熵值在 0 和最大值 $\ln N$ 之间时，移动 agent 的意图是集合 X 中的某些意图(意图的个数大于 1 小于 n)，并且包含的模块既有私有类、共享类又有公用类。如果我们在这些私有类和共享类模块中插入干扰码，那么这些模块就会降低为非私有类。

如果在熵值为 0 的范围内插入干扰码，那么范围就会扩展开。当理想状态的完全扩展范围形成时，熵值达到最大值。

如果 agent 初始态的 K 个模块都是意图 x_i 的私有类，那么将其中 n 个模块扩展之后，(K−n)个模块仍然属于私有类，而另外 n 个模块则属于共享类或公有类。这样，通过插入干扰码扩展了 agent 的初始意图范围，将私有类转换为共享类，或进一步转换为公用类。

在这变化中，模块的范围扩展了，它们包含的意图的范围也随之扩展了。意图 x_i 的私有类扩展成了其它多个意图的共享类，其它意图发生的概率增加了。因此扩展初始意图的目的也达到了。

3.2 意图收缩法

使用分裂法，将原移动 agent 分裂成一组相互协作的移动 agent，实现意图的收缩。

假设一个移动 agent 的意图由集合 X 表示。将 agent 分裂为一组协作移动 agent，其中每一个移动 agent 的意图由集合 Y_i 表示，i=1,2,…,m，则 $X=Y_1\cap Y_2\cap\cdots\cap Y_m$，m 为正整数。

算法 2 意图收缩算法

(1)令 a=1。

(2)移动 agent 模块化。将移动 agent 划分为 K 个模块 $M_1,M_2,\cdots,M_k$，$M_i\subseteq X$，其中 $1\leqslant i\leqslant k$。

(3)给定算法结束时要达到的熵值 H_{end}，且 $0\leqslant H_{end}\leqslant \ln N$。

(4)计算意图 x_i 的概率 $p(x_i)=\frac{t_i}{k}/\sum_{1\leqslant i\leqslant n}\frac{t_i}{k}$，其中 t_i 为包含意图 x_i 的模块数量。

(5)计算意图集 X 的熵值 $H=-\sum_{1\leqslant i\leqslant n}p(x_i)\ln(p(x_i))$。

(6)若 $H_x>H_{end}$，转第(7)步；若 $H_x\leqslant H_{end}$时，结束。

(7)创建新移动 agent，并将其模块化。初始化其意图集 Y_a，令 $Y_a=X$。将移动 agent 划分为 K 个模块 $M_{a1},M_{a2},\cdots,M_{ak}$，其中 $1\leqslant i\leqslant k$，$M_{ai}=\Phi$。

(8)在意图集 X 中取意图 x_i，要求 $p(x_i)>0$。令 a=a+1，$M_j=M_j-\{x_i\}$，$M_{aj}=\{x_i\}$，其中 $x_i\in M_j$，$1\leqslant j\leqslant k$。转第(4)步。

在意图收缩法中，通过与意图扩展法相反的模块转换方向，将意图的范围缩小。

假设原移动 agent 有多种意图，而且某些意图发生的概率大于 0。如果按某种策略将原移动 agent 分裂为多个移动 agent，并且使原 agent 的共享类和公用类向私有类方向转换，则原 agent 意图的范围被划分为多个熵值为 0 或约为 0 的子范围。

意图收缩法将原移动 agent 的意图分散到一组协作 agent 中。恶意主机除非截获全部 agent，且分析出它们之间的关系，否则不能得到正确的数据。

在分裂后，每个移动 agent 的熵值都低于原 agent。为了避免恶意主机将这些分裂后的 agent 集合起来，得到原移动 agent，这些 agent 之间的关系必须非常模糊。总的来说，当以下条件成立时意图收缩法将发生效果：

移动 agent 是可分裂的。

分裂后的移动 agents 在表面看起来是不相关的。可用匿名 agent 来断开它们之间的关系。

4. 有效性分析

4.1 意图扩展分析

假设移动 agent 包含 N 个相互独立的意图，即要收集 N 个相互独立的信息。通过意图扩展，为每个意图添加上干扰码。此时，如果随机选出 N 个意图，那么它们全为原意图的概率为 $1/2^N$。若每个意图分别插入 M−1 个干扰码，那么恶意主机得到移动 agent 的原意图的概率为 $1/M^N$。在上例中，M=3，N=1，则移动 agent 的原意图被发现的概率为 1/3。

当带有干扰码的移动 agent 回到自己的主机时，主机从移动 agent 取得的信息中提出正确的信息。直到此时移动 agent的任务完成了。但在这一过程中，由于主机在最后参与

了选出正确信息的工作，因此移动 agent 的自治性没有很好地体现出来。

由于插入了干扰码，移动 agent 本身的性能相应降低。假设在移动 agent 的 N 个相互独立的意图上，分别添加 M－1 组干扰码，则移动 agent 在这些意图上花费的时间为原来时间的 M 倍。因此移动 agent 性能的降低可表示为：

相互独立的意图数目×插入的干扰码数目，即 N×M。移动 agent 性能的降低主要体现在计算、通信和数据传输上。

4.2 意图收缩分析

假设移动 agent 被派出收集一些项目的信息，并且用一个评估函数来评估这些信息。这个评估函数能通过对各个属性进行加权，找到这些被访问主机中的最优者。评估函数为：$F(x_1,x_2,\cdots,x_{n-1},x_n)=a_1f_1(x_1)+a_2f_2(x_2)+\cdots+a_{n-1}f_{n-1}(x_{n-1})+a_nf_n(x_n)$。

其中，变量 x_i 表示移动 agent 的意图 x_i，a_i 表示意图 x_i 在移动 agent 的决策中所占的权值，且 a_i 满足条件：$\sum_{1\leqslant i\leqslant n}a_i=1$。

可通过对评估函数的分裂实现对 agent 的分裂。如下所示：

$$F(x_1,x_2,\cdots,x_{n-1},x_n)=\sum_{1\leqslant j\leqslant M}F_j(x_1,x_2,\cdots,x_{n-1},x_n)$$

$$a_i=\sum_{1\leqslant j\leqslant M}a_{j,i}$$

$$F_j(x_1,x_2,\cdots,x_{n-1},x_n)=\sum_{1\leqslant i\leqslant M}a_{j,i}f(x_i)$$

将函数 F()分裂后得到的函数 F_i()代替原移动 agent 中的函数 F()。

假设移动 agent 中有评估函数：

F()= 0.2(price)+ 0.3(weight)+0.2(battery-life)+0.3(additional features)

通过意图分裂，可将此原 agent 分裂为 3 个相互独立的 agent，每个 agent 分别有评估函数 F_1()，F_2()，F_3()：

F_1()=0.3(price)+0(weight)－0.1(battery-life)+0.2(additional features)

F_2()=0(price)－0.1(weight)+0(battery-life)+0.1(additional features)

F_3()=－0.1(price)+0.4(weight)+0.3(battery-life)+0(additional features)

F()=F_1()+F_2()+F_3()

由于 agent 在不同的时间到达远程主机，远程主机仅能对它们进行单个的分析，而不能把它们联系起来得到原始的求和因子，因此即使远程主机分析了单个 agent 的评估函数，但它得到的评估算法也是不完全的。如果远程主机要修改 agent 收集的信息，那么除非对这 3 个看来相互独立的 agent 进行的修改是完全相同的，否则，当所有的 agent 返回其主机后，主机会发现在这个远程主机上获得的信息是不可信的。

假设有 t 个原 agent，每个 agent 分裂为 M_i 个 agent(i=1,2,…,t)，即总共有 L 个 agent：

$L=M_1+M_2+\cdots+M_t$

在 L 个 agent 中，主机分辨出其中 M_i 个 agent 为相关 agent 的概率 p(i)为，$p(i)=1/C_L^{M_i}$。当 $M_1=M_2=\cdots=M_L=L/2$ 时，p(i)达到最大值。

与意图扩展法相同，由于原移动 agent 的分裂，移动 agent 本身的性能也相应降低。假设原 agent 中有 K 个意图，原 agent 分裂为 M 个相互独立的 agent，每个 agent 中也带有 K 个意图，则移动 agent 性能的降低为：K×M。

小结 如何保护移动 agent 免受恶意主机的攻击，是移动 agent 安全问题中最困难的部分。在本文中，采用基于意图的方法来保护移动 agent 数据机密性，根据概率学的方法证明能起到一定的作用。但这种方法也存在着不足之处：(1) 必须要以牺牲移动 agent 的部分性能为代价；(2)当有多个恶意主机同时对移动 agent 进行攻击时，主机获得移动 agent 意图的概率就会大大增加。我们今后的工作将围绕如何解决以上问题展开。

参 考 文 献

1 Sander T, Tschudin C F. Protecting Mobile Agents Against Malicious Hosts, in G. Vigna(ed.), Mobile Agents and Security, 1998

2 Sau-koon NG, Kwok-wai Cheung. Intention Spreading: an Extensible Theme to Protect Mobile Agents from Read Attack Hoisted by Malicious Hosts, in Jiming Liu, Ning Zhong(ed.), Intellingent Agent Technology: System, Methodologies and Tools, World Scientific, 1999. 406～415

3 Hohl F. A Protocol to Detect Malicious Hosts Attacks by Using Reference States, 1999

4 董红斌，石纯一．移动 agent 系统的安全问题．计算机科学，2000，27

5 罗飞，邱玉辉．基于意图的移动 Agent 安全机制研究．中国人工智能进展 2001．见：中国人工智能学会第 9 届全国学术年会论文集，2001，12

(上接第 29 页)

中的 eventRegister，也只是当对感兴趣的元组操作发生时(如写、删除)调用客户端的 Callback 方法，在 Callback 方法中客户可对取到的元组进行处理。

可编程反应的元组空间使元组空间表现出一定的智能行为，对提高多 Agent 系统的适应性、健壮性大有益处，例如对移动 Agent 来说，一个关键问题是确定它应该移到哪台主机，一种可行的方案如下：在每个 Agent 平台创建一个固定 Agent，它每隔半小时向元组空间更新元组，元组的内容可以是当前的 CPU 利用率、CPU 速度、内存资源等，利用元组空间服务器端的 Reaction 方法(该方法是在 MARS 系统中的元组空间软件包中定义)可把元组空间的反应编程为：若该平台的负载过多、资源紧张时，查询剩余的 Agent 平台看是否有负载轻的，若有则发出使 Agent 移动的消息，若发现有的平台隔半小时还没有更新元组，则可认为该平台已崩溃，可在别的平台上再启动一个 Agent，不仅使 Agent 的设计轻松简单，还增强了对 Agent 平台的监控能力。如何把可编程反应元组空间技术融合进我们现有的 Agent 系统中是进一步研究的问题。

参 考 文 献

1 Ricci A, Denti E, Omicini A. Agent Coordation Infrastructure for Virtual Enterprises and Workflow Management. Cooperative Information Agents Ⅴ, 5th International Workshop(CIA 2001), Modena (Italy), September 2001, Proceedings, LNCS 2182, Springer-Verlag, 2001

2 Ricci A, Omicini A, Denti E. Enlightened Agents in TuCSoN. Daqli oqqetti aqli agenti: tendenze evolutive dei sistemi software, AIA * IA/TABOO Joint Workshop(WOA 2001), Modena (I), Sep. 2001

3 Lehman T J, Cozzi A, Xiòng Yuhong. Hitting the distributed computing sweet spot with Tspaces. ELSEVIER Computer Networks, 2001, 35: 457～472

4 Boloni L, Jun K. The Bond Agent System and Applications. March 2, 2000. http: // bond. cs. purdue. edu/publication/ASAMA. PS