基于 Multi-Agent 的电子商务中的恶意 Agent 研究*)

唐年庆
(内江师范学院计算机与信息科学系　内江 641112)

摘　要　描述并讨论了以 Agent 为中介的电子商务中的恶意 Agent 问题。恶意 Agent 是一种带有不良意图，能够攻击标准代理，扰乱交易，欺骗用户，盗窃机密信息，浪费资源和破坏市场等的软件实体，在某些方面它们比 PC 病毒更有危害。为此提出了 SCAAM，并且用一些策略来具体描述这种模型的工作过程。

关键词　Agent，恶意代理，电子商务，SCAAM

Malicious Agents Search in Agent-mediated Electronic Markets

TANG Nian-qing
(Department of Computer and Information Science, Neijiang Normal College, Neijiang 641112, China)

Abstract　This paper describes and discusses the malicious agent issues in the agent-mediated electronic markets. We define the malicious agents as software agents that have malicious purpose. Malicious agents can do harm in various ways, such as attacking normal agents, disordering deals, cheating users, stealing intimate information, wasting resources, destroying the markets and so on. In our opinion, malicious agents are more harmful than the common PC viruses in some measure. We propose the SCAAM, and use some strotegies to show the SCAAM how to works.

Keywords　Agent, Malicious agents, Electronic markets, SCAAM

1　基于 Multi-Agent 的电子模型概述

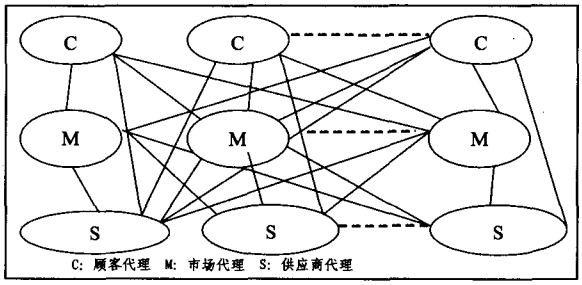


图 1　基于 Multi-Agent 的电子商务模型

当前以 Agent 为中介的电子商务(Electronic Market)大多由三个基本元素组成：顾客代理(Customer Agents)、供应商代理(Supplier Agents)和市场代理(Broker Agents)，它们在电子商务中相遇、谈判和交易[1,2]，如图 1 所示。

电子商务中存在的一些安全问题(如欺骗、攻击和破坏等)已经被发现并确认，同时很多专家学者都对此进行了研究并提出了一些解决方案，但是都不太完善。因此制定更详细的安全交易规则是非常必要的。本文通过分析人们所关心的恶意代理问题，给出了一个解决该问题的方案，它比以往的解决方案具有更高的安全性。

2　电子商务中恶意 Agent 的定义

Agent 是驻留在环境中的实体，它可以解释从环境中获得的反应环境中所发生事件的数据，并执行对环境产生影响的行为[3]。恶意 Agent 就是带有不良意图的软件实体。只要一个 Agent 是为了损害环境(其它 Agents 或者电子商务)而建，不管它的确切的行为和影响怎样，我们都认为它是恶意 Agent。例如：恶意用户为了破坏电子商务发送了一个 Agent 到电子商务中，由于程序编写不好或是电子商务系统防御很好等原因，最终 Agent 没有对电子商务造成任何伤害，我们也说这个 Agent 是恶意 Agent，因为它是为了一个恶意的目的而建立的。

3　电子商务中恶意 Agent 的行为

恶意 Agent 的恶意代码段能够在 Internet 上传播、复制和发作，攻击标准 Agent、扰乱交易、欺骗用户、盗窃机密信息、浪费资源和破坏市场。

攻击标准用户代理是恶意用户用得最多的破坏手段。为了非法获取更多的利益，恶意用户发送恶意 Agent 去攻击标准 Agent，盗取其密码，滥发信息和操纵其行为；扰乱交易是恶意代理可能采取的另一个手段。标准代理在电子商务中必须遵守相遇规则、谈判规则、定货规则等等，而恶意 Agent 想扰乱这些规则是非常容易的；欺骗用户是一些供应商将一些虚假的信息提供给用户，使用户在购物时做出错误的决定。

表 1　恶意 Agent 和 PC 病毒的比较

比较点	恶意 Agent	PC 病毒
建立者	顾客，供应商，电子商务市场建立者，黑客	软件技术人员，系统管理者，黑客
行为	攻击，扰乱，欺骗，盗窃，破坏	浪费资源，删除文件，修改数据，破坏系统
入侵方式	移动，相遇，谈判	复制文件，运行入侵代码，下载软件，接收邮件
入侵速度	快	中

*)内江师范学院院级科研项目 06NJZ-11。**唐年庆**　讲师，硕士，主要研究方向为计算机应用研究。

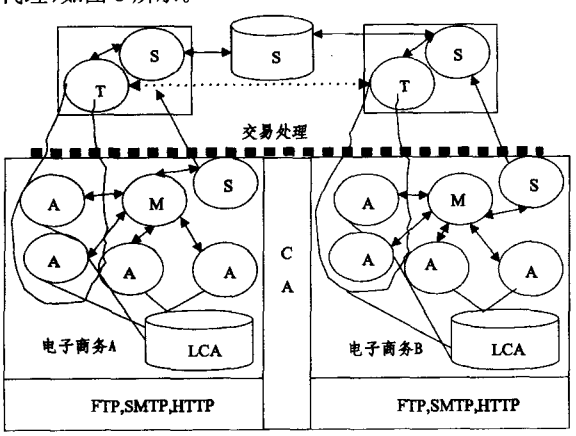
破坏电子商务市场是一个恶意 Agent 所能造成的最坏影响，它能像 PC 病毒一样，浪费资源，删除文件，破坏软件系统和毁损硬盘等。在有些方面，恶意 Agent 造成的危害比 PC 病毒造成的危害更严重。表 1 对两者进行了比较。

4 处理恶意 Agent 的安全认证代理模型 SCAAM

为了解决恶意 Agent 问题，提高以 Agent 为中介的电子商务的效率和可靠性，本文提出了 SCAAM(Security Certificate Authority Agents Model)来保证交易安全可靠，这种电子商务模型(如图 2)，包括交易处理层和基础结构层两层。

假设存在一个用户 U_1 代理 A_{U1} 和一个市场或供应商 U_2 代理 A_{U2}，我们用一些符号来表示一些信息。U 表示用户，P_N 表示公众，S_N 表示个体 N 的私钥，CERT(N)表示个体 N 的数字认证书，$K_N[M]$表示用密匙 K_N 的密码，$SIGN_N(X)$表示个体 N 在事件 X 上的签名，$H()$表示一个解决冲突的哈希函数[4]。

基础结构层是电子商务交易的基础，CA 优于电子商务中其它所有元素，它由其自己的 LCA 来管理每个电子商务市场中的 Agents 行为。在某一时间段，LCA 赋予每个代理最基本的任务，并且其任务只被 LCA 和代理本身所知道。对电子商务市场建立者来说，如果他们想建立一个面向公众的电子商务市场，就必须先向 CA 提供充足的材料来证明他们有建立电子商务市场的能力，然后再提出申请，最后由 CA 对这些材料进行检查审核并决定是否赋予他们有建立电子商务市场的权力。这种方法能够防止非法电子商务市场的建立，并且合法的电子商务市场也在 CA 的监督之下；对一个标准用户(顾客代理、市场代理和供应商代理)来说，在他被允许建立代理之前必须先在一家电子商务市场中注册，然后 CA 分配给他一个公钥，私钥和认证书。LCA 检查一个用户的资料合格后，该用户才能被允许进入电子商务市场并在 LCA 的监督之下进行交易。如果认证通过，两个代理就可以进行交易谈判，为了保证交易结果的公平公正性和完整性，两个代理必须分别发送它们的认证书、密钥和交易结果给当地认证机构，接受当地认证机构的审查监督。当地认证机构检查两个代理密钥的有效日期，如果其中一个代理的密钥是无效的或者已经被终止，交易将被宣布为非法的并被停止，否则当地认证机构将用两个代理各自的密钥加密交易认可签名并分别发送给两个代理，如图 3 所示。



A：标准代理 M：中介代理 S：安全代理
T：交易处理代理 LCA：当地认证机构
CA：根认证机构 SS：系统安全代理

图 2 处理恶意 Agent 的安全认证代理模型 SCAAM

各个代理间通过 M 进行商务活动，同时我们还提供了 S 对访问 M 的所有代理进行安全检测。SS 负责从总体上监视流入和流出电子商务网站的数据流，但它并不是将所有通过的数据流都机械记录下来，而是通过监视电子商务网站上的 S，间接地对电子商务网站实施监控。电子商务网站上的各个 S 负责对该电子商务网站进行监视，完成各自的安全及报警工作，对一些简单的任务，各个 S 可自行处理，只有当发现它处理不了的复杂任务时，才向 SS 发出报警信息，由 SS 作进一步的判断分析。这种有选择地向 SS 发出报警信息的体制，减少了 SS 处理的信息流量，减轻了它的负荷，有利于加快网络上信息传输的速度，进而提高整个模型的效率。

在搜索到的最终满意交易伙伴后，交易处理层通过 T 进行交易处理，同时接受 S 的安全检测。在代理交易处理层中，标准用户可用两种方法来进行交易，它们分别是：一对一代理交易和一对多代理交易。

这种分层模型将 A 移动、相遇、谈判和最终交易处理分离开来，并接受 S 的安全检测，从而减少了被恶意代理攻击的可能性；即使在被攻击成功的情况下，也可以将损失减少到最低程度。

5 策略实现

在模型上我们用 SCAAM 来减少 A 被恶意代理攻击的可能性。另外，在 A 实现策略上，我们可采用意图扩大和意图分割方法来减少 A 被恶意代理攻击的可能性。

代理 A 的意图用 P 来表示，其意图熵为 $S(P)$[5]，我们在它里面加入一干扰码，得到新的代理 NA，它的意图用 NP 表示，设 NP 有 N 个子意图：$NP=\{NP_1, NP_2, \cdots, NP_n\}$，我们用 $g(pi)$来表示 $NP_1, NP_2, \cdots, NP_n$，则有求代理 A 的意图熵公式如下(公式证明略)：

$$S(P)=\sum_{1\leqslant i\leqslant n} g(p_i)\log_2\left(\frac{1}{g(p_i)}\right)$$

由公式我们可以知道，NP 的意图熵在 0 到 $\log_2 n$ 之间，这样 A 的意图得以扩大化，恶意代理就不易得到 A 的真实意图。

U→ LCA:P_{LCA}[T_t,CERT(U),ID(A_U),
TransactionType$_{AU}$,Capability$_{AU}$,SIGN$_U$(H(CERT(U),
ID(A_U),TransactionType$_{AU}$,Capability$_{AU}$),T_t)]
LCA→U:P_U[T_t,SIGN$_{LCA}$(T_t,SIGN$_U$(H(CERT(U),
ID(A_U),TransactionType$_{AU}$,Capability$_{AU}$)))]
U→A_U:CERT(U),ID(A_U),
TransactionType$_{AU}$,Capability$_{AU}$,T_t,
SIGNU(H(CERT(U),ID(A_U),TransactionType$_{AU}$,
Capability$_{AU}$)))
PLCA[key$_{AU}$, ExpiryDate$_{AU}$, T_t], SING$_U$ (P_{LCA} [key$_{AU}$, ExpiryDate$_{AU}$, T_t])

图 3 用户交易认证需求

```
bestprice=150;
besttranscationman="transcationmanA";
if (transcationman[index]. askprice(cosmetic)<bestprice then
{bestprice=transcationman[index]. askprice(cosmetic);
besttranscationman=transcationman[index];}
if (index>=transcationman. length-1 then
{buy(besttranscationman,cosmetic,bestprice);go(host);
```

图 4 未加干扰码买化妆品代码

```
Bestprice1=150; Bestprice2=200; Bestprice3=250;
Besttranscationman1="transcationmanA";
Besttranscationman2="transcationmanB";
Besttranscationman3="transcationmanB"
;price=transcationman[index]. askprice(cosmetic);
effect=transcationman[index]. askeffect(cosmetic);
effecttime=transcationman[index]. effecttime(cosmetic)
;r1=price;r2=effect;r3=effecttime;
```

```
if r1> =bestprice1 then
{transcationman1= transcationman[index];bestprice1=r1;}
if r2> =bestprice2 then
```

图5 加干扰码后买化妆品代码

比如我们想买一化妆品(cosmetic),想问其价格,部分程序如图4。

显而易见,这个代理易被恶意代理攻击得到它收集的最好价格150和提供商是transcationmanA。我们用意图扩大方法加入问效果(effect)和问见效时间(effecttime)等干扰码,防止恶意代理知道我们的真实意图,部分程序如图5。

采用意图分割方法同样能达到隐藏代理真实意图不被恶意代理攻击的目的,本文略。

结束语 以Agent为中介的电子商务是一种全新的在线商店模式,将Agent技术用于电子商务领域是当前的研究热点。恶意Agent带来的灾害比PC病毒更严重,因此能够检测并防御恶意Agent的电子商务体系是非常安全而吸引顾客的。

本文提出了一种适用于当前电子商务市场的SCAAM和实现策略,它能够解决以Agent为中介的电子商务中的恶意Agent危害问题,使电子商务交易公平和完整可靠。

参考文献

[1] 阿瑟.斯加利,威廉,等.B2B交易场:电子商务第三次浪潮.现代出版社,2001

[2] 梅绍祖,吕殿平.电子商务基础.清华大学出版社,2000

[3] Hindriks K V, de Boer FS, van der Hoek W, et al. Agent Programming with Declarative Goals. Intelligent Agents VII: Agent Theories Architectures and Languages, July 2000

[4] 祁明,卓光辉.多智能代理网络购物系统的设计与分析.计算机工程与设计,2001(6)

[5] Caglayan A, Harrison C. Agent Sourcebook. Wiley Computer Publishing, John Wiley & Sons, Inc., 1997

(上接第281页)

访问Web应用的客户端。

结束语 Web应用在全球范围内得到快速的发展,许多组织将Web应用集成到业务关键型的系统中,如电子市场、电子银行、电子贸易以及公共管理服务。由于Web应用的这种广泛而迅猛的发展,使得对其严格的质量需求变得非常紧迫,Web应用的测试为满足这种需求扮演着越来越重要的角色。然而,测试是一个非常耗时、耗财以及耗可计算资源的过程,其有用性可能受到用户交互的限制。显然,实用的测试是使其过程自动化。我们对Web应用建模并采用形式化的测试方法对其进行确认与验证,整个过程很少人工干预,基本实现了自动化。

Web应用测试的主要目的是发现需求中的错误,验证和已定义需求中规定的应用行为的一致性。本文所做的工作是独特的,其主要贡献如下:

(1)使用形式化的建模工具FSM精确地捕捉和定义Web应用的结构和行为;

(2)提出了将ORD转化为FSM模型的算法;

(3)根据给出的转换规则和求解规则,基于FSM模型,提出了一种有效的测试路径自动生成方法,这些测试路径可以转化为XML语法的测试规格说明;

(4)从Web应用的需求本身出发,对其进行分析和建模,提出了一种可行的Web测试模型,根据该模型,最终可以产生测试报告。

对于一个大的Web应用,可以把其分解(比如根据功能、结构或导航行为)成多个子Web应用,然后针对每个子Web应用利用本文提出的方法分别进行建模和测试。当然,各个子Web应用之间的关系也须加以考虑,对此我们将另文给出。

下一步的工作是对本文的建模与测试方法加以改进,进一步提升其自动化程度,并构建一个原型工具实现提出的Web测试模型。

参考文献

[1] Stout G A. Testing a Website: Best Practices. A Whitepaper. http://www.reveregroup.com.

[2] Hieatt E, Mee R. Going Faster: Testing the Web Application. IEEE Software, 2002, 19 (2): 60-65

[3] Jia X, Liu H, Qin L. Formal Structured Specification for Web Applications Testing. 2003 Midwest Software Engineering Conference, Chicago, USA, June 2003

[4] Andrews A, Offutt J, Alexander R. Testing Web Applications by Modeling with FSMs. Software and Systems Modeling, 2004

[5] Kung D C, Liu C H, Hsia P. An Object-Oriented Web Test Model for Testing Web Applications // Proceedings of the 1st Asia-Pacific Conference on Web Applications (APAQS 2000). IEEE Press, New York, USA, 2000: 111-120

[6] Benedikt M, Freire J, Godefroid P. VeriWeb: Automatically Testing Dynamic Web Sites // Proceedings of 11th International World Wide Web Conference. Honolulu, HI, May 2002

[7] Lucca G A D, Fasolino A R. Testing Web-based Applications: The State of the Art and Future Trends. Information and Software Technology, 2006 (48): 1172-1186

[8] Ackroyd M. Object-oriented Design of a Finite State Machine. Journal of Object-Oriented Programming, June 1995: 50-59

[9] van Gurp J, Bosch J. On the Implementation of Finite State Machines // Proceedings of the 3rd Annual IASTED International Conference on Software Engineering and Applications. Scottsdale, Arizona, USA, Oct. 1999: 172-178

[10] Elbaum S, Karre S, Rothermel G. Improving Web Application Testing with User Session Data // Proceedings of the 25th International Conference on Software Engineering. Portland, Oregon, May 2003: 49-59

[11] Offutt J, Wu Y, Du X, et al. Bypass Testing of Web Applications // Proceedings of the 15th IEEE International Symposium on Software Reliability Engineering. Saint-Malo, Bretagne, France, Nov. 2004