

# Mobile Software Agenten und Sicherheit – eine Haßliebe?

Volker Roth

Fraunhofer Institut für Graphische Datenverarbeitung  
Rundeturmstraße 6, 64283 Darmstadt, Deutschland  
vroth@igd.fhg.de

**Zusammenfassung** In diesem Artikel beschreiben wir einen Einblick in den Stand der Technik der Sicherheit Mobiler Agenten und greifen einzelne Details für eine nähere Diskussion auf. Die gebotene Kürze des Artikel zwingt allerdings zu einer starken Einschränkung; daher geben wir für interessierte Leser weiterführende Literatur an.

**Keywords:** Mobile Agenten, Security

## 1 Introduction

Gegenstand dieses Artikels ist die Sicherheit mobile Software-Agenten Systemen. Der Begriff “Software”-Agent legt nahe, daß solche Agenten in irgendeiner Weise programmiert werden. Über die Beziehung zwischen Software-Agenten und Programmen schreiben beispielsweise Franklin et al. in [3]: “All software agents are programs, but not all programs are agents”, und untersuchen eine Reihe von Definition aus der Literatur, was ein Agent denn nun eigentlich sei. Welche Definition auch immer man bevorzugt, eine ganze Reihe von Agentensystemen existieren gegenwärtig im universitären und kommerziellen Bereich [5].

In den folgenden Abschnitten sind wir an Agenten interessiert, die eine besondere Eigenschaft aufweisen – die Eigenschaft *mobil* zu sein. Mobile Agenten haben die Fähigkeit, ihr Programm zusammen mit Zustandsinformationen und weiteren Daten von einem Rechner über ein Netzwerk zu einem anderen Rechner zu transportieren. Im allgemeinen wird dazu eine entsprechende Infrastruktur benötigt, bestehend aus Agentenservern (kurz Server), die Agenten empfangen, starten, anhalten und wieder versenden können. Man kann dabei zwischen *single-hop* und *multiple-hop* Agenten unterscheiden. Single-hop Agenten springen von ihrem Ursprungsrechner lediglich einmal zu ihrem Zielrechner wohingegen multi-hop Agenten vom ersten Zielrechner zu weiteren Rechnern springen. Die multi-hop Agenten lassen sich weiter in solche mit fester vorgegebener Route und solchen mit wahlfreier Route unterteilen. Diese drei Typen

von mobilen Agenten stellen jeweils andere Anforderungen an die Sicherheit des zugrunde liegenden Agentensystems.

Ein wesentlicher Vorteil von mobilen Agenten ist die einfache Softwareverteilung. Mobile Agenten sind in der Lage sich ein Ausführungsumfeld für ihre Berechnungen zu suchen, das ihren Bedürfnissen optimal entspricht. So können sie beispielsweise Daten direkt an der Datenquelle verarbeiten indem sie zu dieser Quelle “springen”, ohne daß die dazu erforderliche Software erst umständlich durch Administratoren installiert werden muß. Nachdem der Agent die Datenquelle verlassen hat, wird automatisch auch dessen Software wieder deinstalliert. Als Beispiel mag die Suche nach Bildern in verteilten Bildbeständen dienen, wobei Bilder gesucht werden, die einem gegebenen Bild “ähnlich” sein sollen. Eine Reihe möglicher Algorithmen für den Vergleich von Bildern existieren, aber nicht alle können gleichzeitig auf allen Bildservern verfügbar gehalten werden. Mobile Agenten erlauben die einfache Verteilung des im konkreten Fall zu verwendenden Algorithmus. Eine herkömmliche Client/Server Anwendung müßte die gesamten Bildbestände über das Netzwerk laden, um die notwendigen Vergleiche auf den Daten lokal vorzunehmen.

Ein weiterer Vorteil ergibt sich, wenn das von dem mobilen Agenten zu lösende Problem von diesem hinreichend autonom behandelt werden kann. In diesem Fall ist es nicht erforderlich, eine permanente Verbindung zum Agenten aufrecht zu halten. Auf das obige Beispiel bezogen bedeutet dies, daß der Agent von einem Endrechner in das Netzwerk eingespeist wird und die Netzverbindung zum Endgerät anschließend abgebaut werden kann. Sobald der Agent fertig ist und der Endrechner wieder online geht, kommt der Agent zurück und präsentiert die Ergebnisse. Dies ist besonders vorteilhaft im Falle mobiler Endgeräte und teurer Netzzugänge. Eine ganze Reihe weiterer Informationen zum *warum* und *wofür* mobiler Agenten finden sich beispielsweise in einem Artikel von Lange [6] und auch in dem Archiv der *mobility* Mailingliste, das unter der URL <http://mobility.lboro.ac.uk/cgi-bin/wilma/mobility> zugänglich ist.

Im Prinzip braucht es nicht viel für ein mobile Agenten-System – Unterstützung für dynamisches Laden, Ausführen und Entladen von Code läßt sich auch auf einem PDA realisieren, wie man Beispiel der *K Virtual Machine*<sup>1</sup> von Sun sehen kann. Wie auf jedem speicherbegrenzten System kommt es auch hier darauf an, wie weitreichend die Unterstützung der Agenten beispielsweise durch Standards und Sicherheitsdienste ausfallen soll.

---

<sup>1</sup> <http://java.sun.com/products/kvm/>

## 2 Elemente der Agentensicherheit

Boshafterweise könnte man sagen, daß der Internet Worm [15, 14, 13], den Robert T. Morris im November 1988 in Umlauf brachte und der innerhalb weniger Stunden Tausende Rechner lahmlegte, der erste öffentlich bekannt gewordene mobile Agent gewesen ist. Der Wurm kam auch ohne spezielle Server aus und nutzte unter anderem Lücken in *sendmail* [9] und dem *fingerd* [19] aus, um sich zu verbreiten.

Trotz guter Gründe für den Einsatz von mobilen Agenten [2, 6] sind noch nicht viele solcher Systeme im Einsatz; Sicherheitsbedenken sind einer der größten Gründe dafür. Wo also steckt dort noch “der Wurm drin?”

Der große Unterschied zwischen mobile Agenten-Systemen und einem Client/Server System besteht darin, daß ein mobiler Agent die Sicherheitsdomäne seines angestammten Rechners verläßt und samt seinem Programmcode – vergleichbar mit einem Virus oder einem Wurm – in die Sicherheitsdomäne seines Zielrechners eindringt, wohingegen Client und Server geschützt aus ihren Sicherheitsdomänen heraus kommunizieren und deren Programme weitgehend unbeeinflußt ablaufen können. Angriffe auf Client/Server Systeme erfolgen im allgemeinen erst einmal auf deren Kommunikationskanal und die Grenzen von deren Sicherheitsdomänen. Die Sicherungsmechanismen, die hier zum Einsatz kommen, sind im allgemeinen die gleichen, die auch heute bereits zur Sicherung der Kommunikation im World Wide Web eingesetzt werden, bestehend aus Authentisierung, Schlüsselaustausch und Verschlüsselung auf Transportebene wie auch allgemeine Maßnahmen zur Rechner- und Netzwerksicherheit.

Mobile Agenten ändern diese Spielregeln. Zum einen kann bei einem mobilen Agenten, der mehrere Sprünge hinter sich gebracht hat, nicht unbedingt angenommen werden, daß dessen Handlungen mit denen von seinem Besitzer gewünschten Handlungen übereinstimmen; der Agent könnte auf einem der Server modifiziert worden sein. Die Identitätsannahme, eine Grundlage, auf der viele traditionelle Sicherheitssysteme beruhen, ist in diesem Fall nicht ohne weiteres haltbar [1]. Zum anderen führen Agentenserver Programmcode aus, der potentiell nicht vertrauenswürdig ist.

Zu den traditionellen Sicherheitsanforderungen eines Client/Server Systems gesellen sich noch weiteren Anforderungen hinzu: (1) Server müssen vor bösarigen Agenten geschützt werden<sup>2</sup>; (2) Agenten müssen vor anderen Agenten geschützt werden, die sich auf dem gleichen Server befinden wie sie selber; (3) Agenten müssen gegen Angriffe durch deren Server geschützt werden<sup>3</sup>. Punkt (2) baut auf Punkt (1) auf; die saubere Trennung von Agenten ist Sache des Ser-

---

<sup>2</sup> Dies ist auch bekannt als das *malicious agent* Problem.

<sup>3</sup> Dies ist auch als *malicious host* Problem bekannt.

vers und Voraussetzung, um diese Aufgabe zu erfüllen, ist, daß sich der Server selber schützen kann.

Zu den Aufgaben des Server zählt insbesondere das Laden von Agenten, deren Prüfung, Installation und Ausführung, wie auch die Terminierung von Agenten und deren Weitertransport. Bei der Erfüllung dieser Aufgabe hat der Server Rechnung zu tragen, daß die Sicherheitsinvarianten eingehalten werden. Insbesondere muß der Server eine Form von *reference monitor* implementieren, der die Agenten überwacht und kontrolliert. Welche Rechte ein Agent ausüben darf, richtet sich nach der Autorisierung des Agenten, üblicherweise im Anschluß an dessen Authentisierung nach dem Laden. Die Authentisierung eines Agenten ist deshalb schwierig, weil sich der Zustand des Agenten regelmäßig ändert. Die Anwendung digitaler Signaturen als Authentikator des Besitzers eines Agenten macht daher nur Sinn, wenn die Signatur auf den unveränderlichen Daten eines Agenten geleistet wird. Ansonsten wird die Signatur beim ersten Zustandswechsel des Agenten ungültig. Die veränderlichen Daten eines Agenten können im wesentlichen nur durch den Server bestätigt werden, auf dem die Änderungen erfolgt sind.

Die Ausführung von Agenten birgt eine Reihe potentieller Gefahren. Agenten könnten versuchen, sich mehr als die ihnen zugestandenen Privilegien zu verschaffen. Welche Rechte man Agenten zuweist will gut überlegt sein. Hat beispielsweise ein Agent die Möglichkeit, das Programm des Servers zu überschreiben, so ist dies gleichbedeutend mit der Ausübung aller Rechte des Servers. Auch könnten Agenten durch Ausschöpfung des Systemressourcen, wie Speicherplatz oder Rechenzeit, anderen Agenten die Nutzung des Servers verwehren. Nicht zuletzt könnten mobile Agenten dazu mißbraucht werden, um den Ursprung von Angriffen auf dritte Ziele zu verschleiern, oder gar einen verteilten koordinierten Angriff auf ein einzelnes Ziel durchzuführen.

Der Agentenserver muß entsprechend sorgfältig entworfen sein, um mit diesen Gefahren fertig zu werden. Abgesehen von der Durchsetzung der Zugriffsbeschränkungen ist die Inspektion und Prüfung von Agentencode ein wichtiger Mechanismus der Serversicherheit. So kann beispielsweise geprüft werden, ob der Code eines Agenten gemäß eines eingeschränkten Regelsatzes "wohlgeformt" ist. Als Beispiel mag hier der *byte code verifier* der Java Virtual Machine dienen. Darüberhinaus kann der Server weitere Prüfungen veranlassen, wie beispielsweise die Prüfung auf die Implementierung "gefährlicher" Methoden. Im Forschungsbereich gibt es darüberhinaus Vorschläge zur Nutzung deduktiver Methoden zur Führung von Beweisen, daß ein Code eine gegebene Sicherheitspolitik (des Servers) einhält [8]. Der Server prüft dann, ob der Code einerseits dem Beweis genügt und der mit dem Code gelieferte Beweis tatsächlich der Sicherheitspolitik entspricht. Dieses Verfahren ist jedoch noch nicht anwendungs-

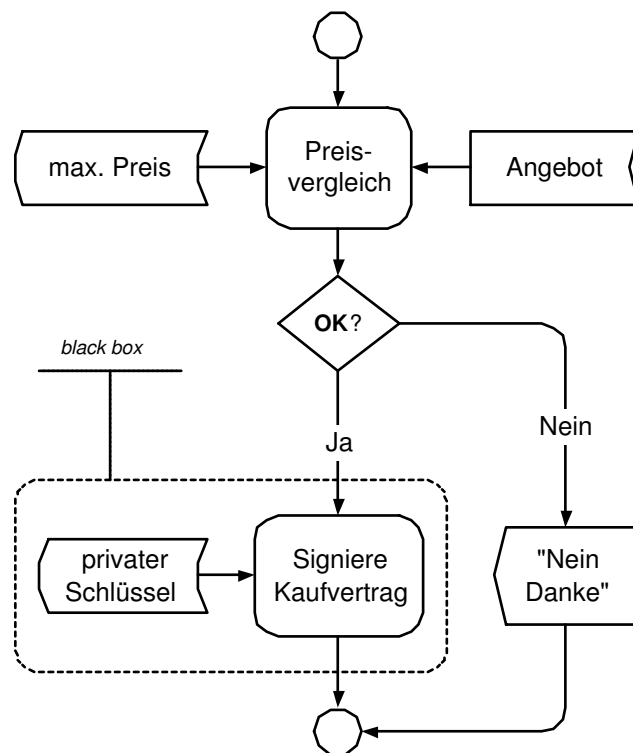
reif. Die Führung des Beweises kann im allgemeinen nur semi-automatisch erfolgen und verlangt ein hohes Maß an Expertise. Reicht dies noch nicht, dann kann der Server in den Code des Agenten auch auf geeignete Weise modifizieren, beispielsweise um eine Ressourcenkontrolle zu ermöglichen (vgl. [4]).

Allen praktischen Schwierigkeiten zum Trotz besteht wenigstens die theoretische Gewißheit, daß ein mit dem Ziel der Sicherheit entworfener Server angemessen gegen böswillige Agenten geschützt werden kann. Beim Schutz von Agenten gegen böswillige Server ist dies nicht der Fall. Zwar sind eine Reihe von technischen Mechanismen mit diesem Ziel publiziert worden, jedoch adressieren all diese Mechanismen im Regelfall einzelne Aspekte des Schutzes mobiler Agenten. Man unterscheidet hier: (1) die Integrität der Daten; (2) die Integrität der Berechnungen; (3) die Vertraulichkeit der Daten und (4) die Vertraulichkeit der Berechnungen eines Agenten. Die grundsätzliche Schwierigkeit ergibt sich daraus, daß der Server den Code des Agenten auf dessen Daten Schritt für Schritt interpretiert und daher über jede Operation Kenntnis erlangt und diese beeinflussen kann.

Ein Beispiel dazu zeigt Abbildung 1. Der dort abgebildete Algorithmus entspricht einer üblichen Abfolge von Operationen, mit denen ein Agent auf einem Server einen Kauf tätigen könnte. Im Zuge der Ausführung des Agenten erhält der Server Kenntnis des zur Bestätigung des verbindlichen Kaufvertrages notwendigen privaten Schlüssels und könnte diesen dazu nutzen, um beliebige Verträge im Namen des Besitzers des Agenten zu signieren. Sander und Tschudin haben unter Nutzung der Homomorphieeigenschaft geeigneter Kryptosysteme einen Vorschlag gemacht, wie die Signaturoperation auf eine Weise erfolgen kann, die den privaten Schlüssel vor Komprimittierung schützt [12]. Ein geübter Programmierer mit einem Debugger sollte jedoch keine Schwierigkeiten haben, den als *black box* dargestellten Code und die zugehörigen Daten zu isolieren und auch ohne Kenntnis des privaten Schlüssels damit beliebige Eingaben signieren zu lassen. Dies zu verhindern ist noch nicht angemessen gelöst worden.

Eine weitere Alternative ist die Verteilung der Anwendung und kritischer Daten auf mehrere Agenten, die aus unterschiedlichen Sicherheitsdomänen heraus operieren und miteinander kommunizieren. Voraussetzung ist hierbei allerdings, daß die Server der kooperierenden Agenten diese nicht gemeinsam angreifen [10].

Verlegt man sich auf die Erkennung von Angriffen anstatt der Prävention, dann bietet das Protokoll von Vigna [16] einen Ansatz auf dessen Basis eine falsche Ausführung von Agenten erkannt werden kann. Mit jedem Transport eines Agenten bestätigt der Server auf eine nicht abstreitbare Weise einen *execution trace*. Besteht der Verdacht auf einen Angriff, dann kann der Besitzer des Agenten die traces seines Agenten anfordern, die Ausführung des Agenten



**Abbildung1.** Der Kaufalgorithmus, wie er von einem beliebigen Agenten durchgeführt werden könnte.

auf diesem Trace emulieren und einen Nachweis auf Korrektheit führen. Allerdings ist das Verfahren auf Agenten mit einem einzelnen thread beschränkt und erfordert einen sehr hohen Aufwand für die Speicherung der traces und deren Nachweis. Hier wird noch an effizienteren Verfahren der Beweisführung geforscht.

Eine Reihe weiterer Verfahren und Ansätze zur Sicherung verschiedener Aspekte von mobile Agenten-Systemen sind beispielsweise in [17, 18, 11] beschrieben.

### 3 Abschluß

Bevor man jedoch die Frage nach der Sicherheit eines konkreten mobile Agenten-Systems stellt, ist als erstes zu klären, welche Sicherheitsbedürfnisse überhaupt existieren und welches Bedrohungspotential sich für die angestrebte Anwendung ergibt. Eine Anwendung, die nur einer eingeschränkten vertrauenswürdigen Nutzergruppe zugänglich ist, und die innerhalb eines vertrauenswürdigen Intranet oder über ein *Virtual Private Network* [7] läuft, kann durchaus nützlich sein und hat weit geringere Sicherheitsanforderungen als eine weit verteilte offene Plattform für mobile Agenten im Internet. Ist die eigene Anwendung eher letzterer Kategorie zuzuordnen, dann kommt man um eine genaue Analyse der Sicherheitsanforderungen und des technischen Schutzes des gewählten Systems ohnehin nicht herum.

Generell sollte man sehr Mißtrauisch werden, wenn der Anbieter eines Systems die Sicherheit von mobilen Agenten gegenüber böartigen Hosts anpreist, ohne genaue Angaben zu machen, welche Aspekte durch welche Maßnahmen geschützt sind. Hier hat das Marketing die rosarote Brille auf.

### Literatur

1. David M. Chess. Security issues in mobile code systems. In Vigna [17], pages 1–14.
2. Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile agents: Are they a good idea? Technical report, IBM T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, March 1995.
3. Stan Franklin and Art Graesser. Is it an agent, or just a program? In *Intelligent Agents III*, volume 1193 of *Lecture Notes in Artificial Intelligence*, pages 21–36, Berlin, 1997. Springer Verlag.
4. Grzegorz Gzajkowski and Thorsten von Eicken. JRes: A resource accounting interface for Java. In *Proc. ACM OOPSLA Conference*, Vancouver, BC, October 1998.
5. Fritz Hohl. The mobile agent list. Internet document available at URL <http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/mal/mal.html>. Version current 11 October 1999.
6. Danny B. Lange and Mitsuru Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, March 1999.

7. Martin Murhammer, Tim Bourne, Tamas Gaidosch, Charles Kunzinger, Laura Rademacher, and Andreas Weinfurter. *A Guide to Virtual Private Networks*. Prentice Hall, 1999.
8. George C. Necula and Pete Lee. Safe, untrusted agents using proof-carrying code. In Vigna [17], pages 61–91.
9. Jonathan B. Postel. Simple Mail Transfer Protocol. Request for Comments 821, Internet Engineering Task Force, aug 1982.
10. V. Roth. Mutual protection of co-operating agents. In *Secure Internet Programming* [18].
11. K. Rothermel and F. Hohl, editors. *Proceedings of the Second International Workshop on Mobile Agents (MA '98)*, volume 1477 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin Heidelberg, September 1998.
12. Tomas Sander and Christian F. Tschudin. Protecting mobile agents against malicious hosts. In Vigna [17], pages 44–60.
13. D. Seeley. A tour of the Worm. In *Proceedings of 1989 Winter USENIX Conference*, San Diego, CA, February 1989. Usenix Association.
14. E. Spafford. The Internet Worm: Crisis and aftermath. *Communications of the ACM*, 32(6):689–698, June 1989.
15. E. Spafford. The Internet Worm program: An analysis. *Computer Communication Review*, 19(1), January 1989. Also issued as Purdue CS Technical Report CSD-TR-823, 28 November 1988.
16. Giovanni Vigna. Cryptographic traces for mobile agents. In *Mobile Agents and Security* [17], pages 137–153.
17. Giovanni Vigna, editor. *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin Heidelberg, 1998.
18. Jan Vitek and Christian Jensen. *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, volume 1603 of *Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1999.
19. D. Zimmerman. The Finger User Information Protocol. Request for Comments 1288, Internet Engineering Task Force, December 1991. Obsoletes RFCs 1196, 1194, 742.