International Journal of Computers, Communications & Control
Vol. III (2008), No. 1, pp. 60-68

Evaluation of the Recorded State Mechanism for Protecting Agent Integrity Against Malicious Hosts

Kamalrulnizam Abu Bakar, B. S. Doherty

Abstract: As agent technology is expected to become a possible base platform for an electronic services framework, especially in the area of Electronic Commerce, reliable security protection is a crucial aspect, since some transactions in this area might involve confidential information, such as credit card number, bank account information or some form of digital cash, that has value and might therefore be attacked. In addition, without proper and reliable security protection, the wide spread use of agent technology in real world applications could be impeded. In this paper, evaluation of the Recorded State Mechanism (RSM) previously proposed by the authors is presented. The evaluation examines the RSM security protection and implementation overhead, in order to analyse the RSM security strength and implementation feasibility in real world application.

Keywords: Agent security, Malicious host, Recorded State Mechanism.

1 Introduction

Problem in agent technology arise when agents are used in an open and unsecured environment. For example, a customised agent application is sent out to visit several airline servers (in an open and unsecured environment) to find a suitable flight. In this example, the agent application is allowed to completely migrate to (the agent originating host transfer the agent's code, data and state to the remote server) and execute in (the remote server executed the receiving agent application) the remote server environment, to take advantage of exploiting resource near the data source and thus reducing network traffic [20]. This opens a greater opportunity for the agent application to be abused by the executing host, because the agent application is fully under control of the executing host [10, 19].

An example of attack by the executing host (the malicious host) is to tamper with the agent's data or state, so that the agent will forget all the previous visits and offers held by the agent, and thus force the agent (application) to accept an offer from the malicious host even though the malicious host's offer is not the best offer [10, 17, 19]. This kind of attack is known as a *manipulation attack* [11, 10]. In this attack, the owner of the agent may not know the attack has happened. This is because the malicious host may make subtle changes in the agent's code, data and state, which are difficult to detect, thus enabling the malicious host to achieve its objective. In addition, the agent (application) that returns from the malicious host does not show any different behaviour from an untampered agent, which makes the attack difficult to detect and prevent.

The problem of manipulation attack has been addressed by the authors using the Recorded State Mechanism (RSM) [1, 2]. The RSM uses the state of an agent, which is recorded during the agent execution process inside an execution host environment to detect the malicious host manipulation attack. In this paper the evaluation of the Recorded State Mechanism is presented. The evaluation analyse the RSM's security and overhead its feasibility in real world applications.

The paper is organized as follows: section 2 presents the evaluation of the Recorded State Mechanism, which includes the analysis on the security and implementation overhead of the RSM. Section 3 presents a discussion and the conclusion is presented in section 4.

2 The Evaluation of the Recorded State Mechanism

The Recorded State Mechanism is an integrity protection mechanism that is able to detect manipulation attacks from a malicious host. The mechanism consists of three different types of container, the RecordedReadOnly, RecordedExecuteOnly and RecordedCollectOnly that are used to record the agent state information. This recorded agent state information, which consists of the data of the agent (located in its variables) and the execution information (such as the program counter, the call stack and a few more items) is used for detecting any modification attacks from the malicious host in order to protect the integrity of the agent during agent execution inside the malicious host environment.

The evaluation of the Recorded State Mechanism is presented by examines the RSM security protection and implementation overhead that will be discussed in the next section.

2.1 The Security Analysis of the Recorded State Mechanism

To assess the strength of the Recorded State Mechanism, its ability to handle well-known attacks is discussed in Table 1.

Summary of evaluation of Recorded State Mechanism

The Recorded State Mechanism is able to detect most of the malicious host attacks that try to tamper with the agent's data and state integrity. This mechanism when combined with distributed migration pattern, can prevent collaboration attacks by two or more hosts and extraction of information by the malicious host. However attacks such as an execution host lying about input data cannot be detected or prevented by this mechanism, because the attack does not alter any state information, and so leaves no trace.

2.2 The Overhead of Implementing the Recorded State Mechanism

The experiments to measure the overhead of implementing the Recorded State Mechanism are conducted using six 400 MHz Sun Ultra Sparc 5 workstations with 128 MB of main memory. Each of the workstations is running the Solaris 8 operating system and is connected to the others using 100 Mbit/s UTP[1] cable. All of the workstations involved in this experiment were situated in the same room.

In this configuration, one workstation will be chosen among the six workstations to be the home host for the agent, and only this host has the permission to manage and dispatch the agent. The rest of the workstations are assumed to be the remote host, having only the capability to receive and dispatch the agent back to its home host.

To evaluate the security overhead for implementing the Recorded State Mechanism in an agent-based application, times are measured starting from sending of the agents to the remote hosts and ending by receiving the agents back from the remote hosts. The times, are measured using the "System.currentTimeMillis()" method in the Java language. This method produces a specific instant in time with millisecond precision [14].

The experiments are done using four different remote host starting with one remote host, two remote hosts, three remote hosts and five remote hosts on three different types of agent: plain agent[2], agent with cryptographic security mechanism (Crypto) and agent with these security mechanisms and the recorded state mechanism (Crypto+RSM). There are four different experiments used in examining the overhead for implementing the Recorded State Mechanism: one input and one cycle, one hundred inputs and one cycle, one input and one thousand cycles, and one hundred inputs and one thousand cycles. The input is

[1] Unshielded Twisted Pair Category 5e

[2] agents without security mechanisms

Attacks	Solutions
The malicious host could make subtle changes on read-only data inside the RecordedReadOnly container to enable it to achieve its objective, in such a way that the owner's digital signature that was signed in the RecordedReadOnly container still remains valid.	This attack can be ruled out, because the digital signature (using SHA1), which is used in the Recorded State mechanism is secured against brute-force collision and inversion attacks, where by using the SHA1 as the digital signature function could make the attacker computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.
The malicious host could also make subtle changes on both the read-only data and the digital signature for the RecordedReadOnly container, in order to make both of them appear to be valid.	This possible attack can also be ruled out because in order to create a new digital signature that will be valid for other host, the malicious host needs to have a private key of the agent owner. However, only the key owner has this key and no other entity can produce this key from a modified hash value.
The malicious host could also tamper with the agent state recorded in the RecordedExecuteOnly and RecordedCollectOnly container by modifying the agent state before the state is recorded into both containers. This due to the fact that the recorded process of the Recorded State Mechanism is under the malicious host's control and therefore, the malicious host can do anything to it.	This attack can also be ruled out because the malicious host has to use its own private key that contains its identity in order to compute and re-compute the digital signature on the tampered state, thus revealing itself during the verification process.
The malicious host could also attack the agent by launching collaboration attacks in cooperation with two or more consecutive hosts in order to deny the checking process for detecting any malicious host attack from the previous visit or to remove any agent state that records the changes made by the previous host on the agent during its execution session. In addition, the malicious host could also use the collaboration attack to extract information from the agent to win some competition over other hosts.	The attack can be ruled out since the used of master-slave agent architecture in implementing the Recorded State Mechanism only allows different agents to be sent and served by different remote hosts. An agent visits only one host, thus precluding the collaborating attack. In addition, the information extracted from a single agent does not give enough information to the malicious host to allow it to win any competition over other hosts because the extracted information is not sufficient by itself.
The malicious host could also lie about the input data, which is recorded in the RecordedExecuteOnly and the RecordedCollectOnly containers in order to deceive the owner of the agent.	This attack is unable to be ruled out by the Recorded State Mechanism because the input data that was supplied by the malicious host is assumed to be a correct data by the agent, since only the malicious host knows whether the input data is correct or incorrect. However, the owner of the agent knows the identity of the host, which supplies the input data to the agent because all the data and state are digitally signed by the execution host before the data and state leaved the execution host. Thus, the owner of the agent knows which execution host is responsible for supplying the false input data.
The malicious host could alter the Random Sequence 3-level obfuscation algorithm to execute in many different ways (incorrect execution attack)	This attack can be ruled out since the Recorded State Mechanism will check the results gathered by the returning slave agent by executing the same execution process that is assumed, has been done by the slave agent inside the remote host execution environment.

Table 1: Possible attacks and solutions of the Recorded State Mechanism

represents a character. This character is use as a data that need to be protected by the agent. The cycle, on the other hand represents a loop that is used to simulate an agent tasks.

Note that all of the experiments on the Recorded State Mechanism are using master-slave agent architecture and operates on the distributed migration pattern [1, 2].

The experiment is performed for 20 runs and the result for each run is gathered in milliseconds. From

the author's observation, all the 20 runs in this experiment give very similar results and for this reason, 20 runs of the experiment are considered sufficient. The average result of all the 20 runs is taken and converted into seconds. The result is then rounded up and presented in two decimal places as given and illustrated in Tables 2 to 5 and Figure 1 to 4 respectively.

Number of Remote Hosts	Mean			Standard Error			Standard Deviation		
	Plain	Cyp	Cyp+RSM	Plain	Cyp	Cyp+RSM	Plain	Cyp	Cyp+RSM
1	1.54	29.15	28.91	0.003	2.18	1.23	0.01	9.73	5.52
2	2.49	30.89	31.11	0.003	2	1.38	0.02	8.93	6.19
3	3.37	31	32	0.004	0.93	0.98	0.02	4.15	4.38
5	5.35	36.36	36.03	0.011	1.5	1.87	0.05	6.7	8.37

Plain = Without Cryptographic Mechanism
Cyp (Crypto) = Cryptographic Mechanism
RSM = Recorded State Mechanism

Table 2: Summary Statistics of The Recorded State Mechanism Overhead (1 Input and 1 Cycle Experiment)

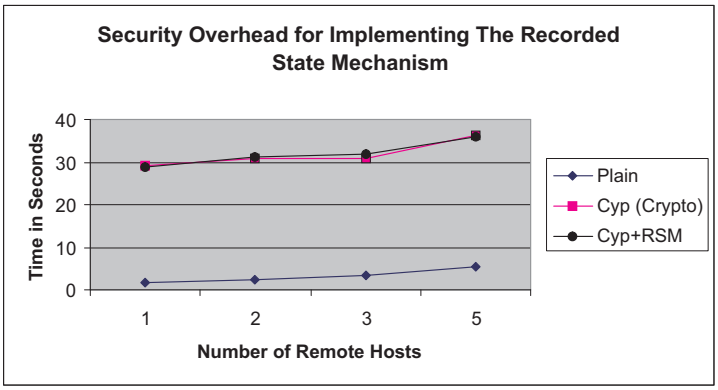


Figure 1: Security Overhead of The Recorded State Mechanism (1 Input and 1 Cycle Experiment)

Based on the observation on the results gained through the experiments done, it can be seen that the standard error and the standard deviation of the security overhead are similar regarding the number of remote host but different between agents. Agents with security mechanism give larger standard error since the agents have to execute many tasks such as generate the cryptography key, generate digital signature, verify digital signature and execute encryption and decryption.

From the results given in Table 2 and illustrated in Figure 1, it can be seen that the mean of the security overhead is almost the same for agents with security mechanism and agents with security mechanism plus RSM, where the security overhead for the agent with security mechanism plus RSM is just 7.82 % higher than the overhead for the agent with security mechanism. However, both agent's security overheads are higher by up to 1792.86 % than the overhead for the plain agent.

From Table 3 and Figure 2, the security overhead for the plain agent is almost the same as with one input given in Table 2 and Figure 1, but the security overhead for the agents with security mechanism is increased by up to 60.52 % along the security overhead with one input.

Results in Table 4, Table 5, Figure 3 and Figure 4 show that the security overhead for all the agents

Number of Remote Hosts	Mean			Standard Error			Standard Deviation		
	Plain	Cyp	Cyp+RSM	Plain	Cyp	Cyp+RSM	Plain	Cyp	Cyp+RSM
1	1.55	45.43	45.32	0.003	1.21	1.19	0.01	5.42	5.34
2	2.53	45.99	48.21	0.005	1.35	1.83	0.02	6.03	8.18
3	3.37	49.76	49.42	0.002	1.13	1.26	0.01	5.03	5.65
5	5.43	53.09	53.14	0.005	1.09	1.76	0.02	4.86	7.85

Table 3: Summary Statistics of The Recorded State Mechanism Overhead (100 Input and 1 Cycle Experiment)

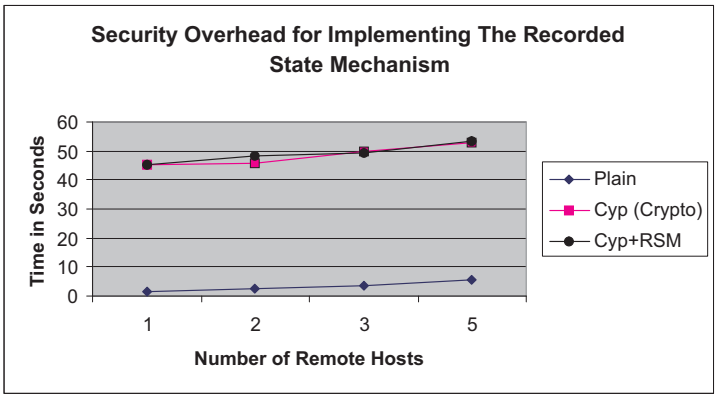


Figure 2: Security Overhead of The Recorded State Mechanism (100 Input and 1 Cycle Experiment)

is similar to the security overhead of the agents with the same number of input but different number of cycle given in Table 2, Table 3, Figure 1 and Figure 2 respectively. Therefore, it is worth noting that number of cycles does not affect the security overhead of the agents.

Number of Remote Hosts	Mean			Standard Error			Standard Deviation		
	Plain	Cyp	Cyp+RSM	Plain	Cyp	Cyp+RSM	Plain	Cyp	Cyp+RSM
1	2.41	27.65	28.94	0.008	0.77	1.55	0.04	3.45	6.94
2	3.53	30.91	31.31	0.008	1.38	1.68	0.04	6.18	7.53
3	5.17	30.94	33.36	0.005	1	1.04	0.02	4.46	4.65
5	7.34	38.46	37.57	0.01	1.29	1.69	0.05	5.76	7.55

Table 4: Summary Statistics of The Recorded State Mechanism Overhead (1 Input and 10000 Cycle Experiment)

Summary of experimental results

It can be seen from the results shown in Tables 2 to 5 and illustrated in Figures 1 to 4 that the implementation of the Recorded State Mechanism does increase the overhead by only up to an acceptable 7.82 % when compared to the agent with security mechanism but 2830.96 % when compared to the plain agent. However, the low overhead of the plain agent is not important since the plain agent does not have any security protection.

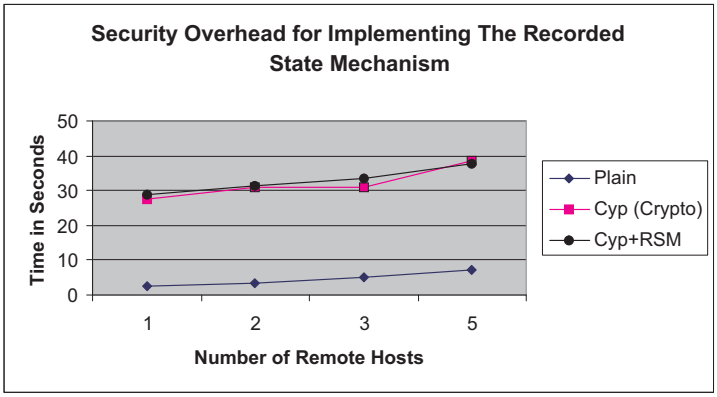


Figure 3: Security Overhead of The Recorded State Mechanism (1 Input and 10000 Cycle Experiment)

Number of Remote Hosts	Mean			Standard Error			Standard Deviation		
	Plain	Cyp	Cyp+RSM	Plain	Cyp	Cyp+RSM	Plain	Cyp	Cyp+RSM
1	2.39	45.93	47.26	0.005	1.06	1.11	0.02	4.76	4.97
2	3.56	48.63	48.85	0.005	0.73	1.42	0.02	3.28	6.36
3	5.17	50.86	51.54	0.008	1.52	1.57	0.03	6.82	7.01
5	7.4	54.36	54.48	0.015	1.8	1.94	0.07	8.05	8.66

Table 5: Summary Statistics of The Recorded State Mechanism Overhead (100 Input and 10000 Cycle Experiment)

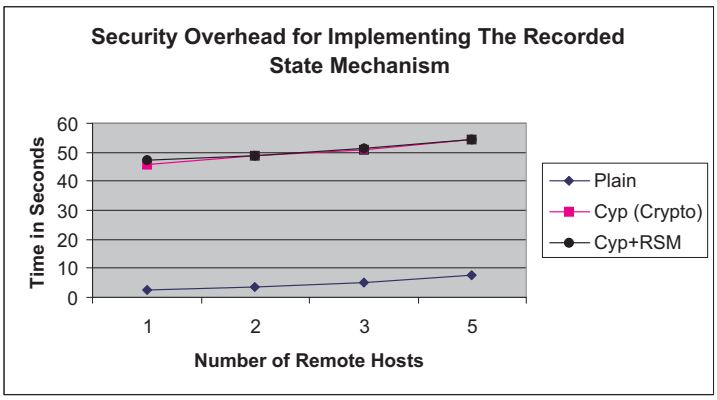


Figure 4: Security Overhead of The Recorded State Mechanism (100 Input and 10000 Cycle Experiment)

3 Discussion

Integrity protection is one of the main requirements for protecting agents against a malicious host attacks. The requirement was successfully fulfilled[3] by the Recorded State Mechanism, which is able to detect most of the malicious host attacks that try to tamper with the agent's data and state integrity.

The analysis on security strength and implementation feasibility of the Recorded State Mechanism in real world applications has been conducted. The security strength of the Recorded State Mechanism has

[3]detect or prevent some attacks and made others more difficult

been analysed by evaluating the mechanism against well-known attack scenarios, and from the results, it can be seen that the mechanism is capable to prevent or detect some of the attacks and made other attacks more difficult. The implementation feasibility is measured by examining the overhead imposed by the mechanism in protecting agents integrity against malicious host attacks. The result shows that the RSM imposed an acceptable overhead.

4 Conclusion

This paper presented the evaluation of the Recorded State Mechanism for protecting the integrity of the agents against the malicious host attacks. The evaluation produced significant results on the strength of the Recorded State Mechanism, where it is able to prevent or detect some attacks and made other attacks more difficult with an acceptable overhead. In conclusion, the mechanism offered significant advances in protection of agents against malicious host attacks and is therefore suitable for use in real world applications.

Bibliography

[1] Abu Bakar, K. and Doherty, B. S. A New Model for Protecting Mobile Agents against Malicious Host. Proceedings of the IADIS International Conference WWW/Internet. IADIS Press, Portugal (2002) 780-784

[2] Abu Bakar, K. and Doherty, B. S. Protecting Mobile Agents Against A Malicious Host Attacks Using Recorded State Mechanism. Proceedings of the 2003 International Conference on Informatics, Cybernetics and Systems. I-Shou University(2003) 396 – 401

[3] Chess, D.M. Security Issues in Mobile Code Systems. G. Vigna(Ed.): Mobile Agents and Security, Vol. 1419. Springer Verlag (1998) 1 – 14

[4] Chess, D.M. and Harrison, C.G. and Kershenbaum, A. Mobile Agents: Are They a Good Idea?. IBM Research Report. IBM Research Division (1995). http://www.research.ibm.com/iagents/publications.html

[5] Corradi, A. and Cremonini, M. and Montanari, R. and Stefanelli, C. Mobile Agents Integrity for Electronic Commerce Application. Information System. Elsevier Science (1999) 519 – 533

[6] Diaz, J. and Gutierrez, D. and Lovelle, J. An Implementation of A Secure Java2-Based Mobile Agent System. Proceedings of The Second International Conference on The Practical Application of Java. Practical Application Company (2000) 125 – 142

[7] Farmer, W.M. and Guttman, J.D. and Swarup, V.: Security for Mobile Agents: Issues and Requirements. Proceedings of the 19th National Information System Security Conference. Baltimore (1996) 591-597

[8] Ford, W. and Baum, M. Secure Electronic Commerce, Ed. 2nd. Prentice Hall (2001)

[9] Guan, X. and Yang, Y. and You, J. POM - A Mobile Agent Security Model against Malicious Hosts. Proceedings of IS & N'99 Spring Verlag (1999) 155 – 167

[10] Hohl, F. A Framework to Protect Mobile Agents by Using Reference States. In: Proceedings of the 20th international conference on distributed computing systems (ICDCS 2000). IEEE Computer Society (2000) 410 – 417

[11] Hohl, F.: Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts. In: G. Vigna (Ed.). Mobile Agent and Security. Lecture Notes in Computer Science, Vol. 1419. Springer-Verlag, Berlin(1998) 92–113

[12] Hohl, F. A Model of Attacks of Malicious Hosts Against Mobile Agents. In 4th ECOOP Workshop on Mobile Object Systems (MOS'98): Secure Internet Mobile Computations. (1998) http://mole.informatik.uni-stuttgart.de/papers.html

[13] Hohl, F. An Approach to Solve the Problem of Malicious Hosts. Institute of Parallel and Distributed High-Performance Systems (IPVR), University of Stuttgart, Germany (1997)

[14] Sun Microsystems, Inc. Java 2 Platform Std. Ed. V1.3.1 http://java.sun.com/j2se/ 1.3/docs/api/index.html (2004)

[15] Kun, Y. and Xin, G. and Dayou, L. Security in Mobile Agent System: Problems and Approaches. Operating System Review, Vol. 34, No. 1. ACM (2000) 21 – 28

[16] Reisner, J. and Donkor, E. Protecting Software Agents from Malicious Hosts using Quantum Computing. Proceedings of SPIE - The International Society for Optical Engineering. IEE (2000) 50 – 57

[17] Sander, T. and Tschudin, C.: Protecting Mobile Agent Against Malicious Hosts. In: G. Vigna (Ed.). Mobile Agent and Security. Lecture Notes in Computer Science, Vol. 1419. Springer-Verlag, Berlin(1998) 44-60

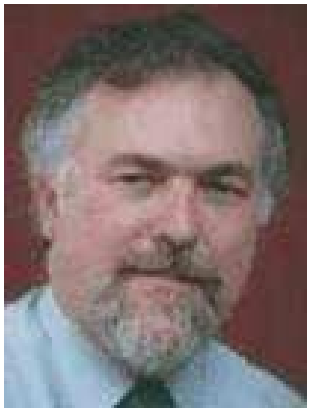
[18] Schneier, S. Applied Cryptography, Ed. 2nd. Wiley & Son (1996)

[19] Vigna, G. Cryptographic Traces for Mobile Agents. In: G. Vigna (Ed.). Mobile Agent and Security. Lecture note in Computer Science, Vol. 1419. Springer Verlag (1998) 137 – 153

[20] Wang, T., Guan, S. and Chan, T.: Integrity Protection for Code-on-Demand Mobile Agents in E-Commerce. The Journal of Systems and Software. Elsevier (2002) 211-221

Kamalrulnizam Abu Bakar
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81310 UTM Skudai
Johor D. T.
Malaysia
E-mail: kamarul@fsksm.utm.my
B. S. Doherty
School of Engineering and Applied Science
Aston University
Aston Triangle, Birmingham B4 7ET
United Kingdom
E-mail: b.s.doherty@aston.ac.uk
Received: July 22, 2006

Kamalrulnizam Abu Bakar is a lecturer at Universiti Teknologi Malaysia, Malaysia. He received the diploma and degree of Computer Science in 1994 and 1996 respectively from Universiti Teknologi Malaysia, Malaysia. He then received Masters in Computer Communication and Networks degree from Leeds Metropolitan University, United Kingdom in 1998 and PhD in Network Security from Aston University, United Kingdom in 2004. His current research interests include computer and network security, distributed systems and parallel processing, grid computing, wireless and cellular network.



Bernard S. Doherty (born October 2nd, 1945) obtained the degrees of Bachelor of Engineering (Electrical), Bachelor of Arts and Master of Engineering Science from the University of Melbourne in 1967, 1971 and 1981 respectively. He has held positions with the State Electricicity Commission of Victoria, LM Ericsson Pty Ltd, Swinburne College of Technology (all in Melbourne) and, since 1980, at Aston University (Birmingham, UK), where is presently Lecturer in Computer Science. His main fields of teaching and research are Distributed and Networked applications and Information Security. In addition to supervising a number of Doctoral students, he has developed computer-based administration and teaching software, written a number of papers and presented papers at international conferences.