A NOVEL CREDIBILITY BASED SECURITY MECHANISM FOR MIGRANTS

Ashlesha Gupta
YMCA University of Science and Technology, Faridabad
gupta_ashlesha@yahoo.co.in

Ashutosh Dixit
YMCA University of Science and Technology, Faridabad
dixit_ashutosh@rediffmail.com

Ashok Kumar Sharma
BSAITM, Faridabad
ashokkale2@rediffmail.com

ABSTRACT

To search and access desired information from WWW, users rely on the use of Search Engines which provides an interface to access Web pages. Web search engines employ web crawlers to continuously collect web pages from the web, index and store them in a database. In traditional crawling, the pages from all over the web are brought to the search engine side and then processed, that results in a lot of network traffic. Therefore, the capabilities of mobile agents have been utilized to design migrating crawler, which move to the information resources for downloading the documents, resulting in reduced load on a single machine. Migrating crawling instances filter and compress the documents at the remote host itself before transferring them to the search engine repository. Since the migrant's code from the search engine side is transferred and executed on web servers, an environment controlled by another party, gives rise to several security issues in mobile agent computing. Such issues include authentication, authorization (or access control), intrusion detection etc. Security issues are becoming more significant in the case of large heterogeneous distributed web. This paper presents a credibility based approach to maintain safety and security of the migrants as well as environment in which they execute.

Keywords: Search Engine, Migrating Crawlers, Migrants, Security, Credibility Value.

I. INTRODUCTION

Mobile agents are active objects that can autonomously migrate in a network to perform tasks on behalf of their owners. Current web crawler uses the concept of Mobile Agent to enhance their crawling speed. In mobile crawling, mobile agents called migrants are dispatched to remote web servers for local crawling and processing of web documents. After crawling a specific web server, they dispatch themselves either back at the search engine machine, or at the next web server for further crawling. Migrating crawling agents (migrants) based methods minimize network utilization and also keep update of documents. However, security is one of the major aspects that hamper the use of mobile agents. The server in a mobile agent environment can be attacked by malicious agents with illegal codes. Attacks may be the denial of service or Unauthorized Access or Masquerade. Similarly a mobile agent roaming in the distributed network can be attacked by malicious platforms.

A malicious platform may modify or kill the agent. Securing mobile agents from various security threats therefore becomes very essential. In this paper, a novel protection mechanism has been proposed to safeguard the mobile agent as well as the remote host using trustworthiness based approach. The proposed approach adjusts migrant's security by computing credibility factor. An agent who is more credible is allowed to execute freely, with more resources and permissions.

The rest of this paper is organized as follows: Section II discusses previous ways of protecting agents and platforms. Section III summarizes the credibility based security mechanism. Analysis of the proposed mechanism is done in Section IV. Section V summarizes the work.

II. LITERATURE REVIEW

Mobile agent based crawlers are becoming increasingly popular due to their mobility, flexibility and adaptability. However reliability and security of Mobile agents is still an open issue and requires a lot of concern . Security of Mobile crawlers falls mainly into two categories

- Protecting a server from malicious agents: A Malicious agent can use illegal codes to attack the server. The possible attacks include Masquerading, Unauthorized access and denial of service etc.To protect the server from malicious agents; authentication and byte code verification are the major security requirements.

- Protecting a mobile agent from malicious servers: A malicious platform can destroy or alter the agent by modifying its code, data and flow control. Protecting the mobile agent from malicious servers is a more critical issue , because the agent dispatched to remote servers is fully under the control of the remote server. The major security requirements for protecting the mobile agent environment are confidentiality and integrity, and defence against colluded attacks.

Many techniques for protecting server platforms from malicious agents have been proposed such as :

- Sandbox (Wahbe et al 1993) is the protection mechanism which provides execution environment separately for the agent. Every agent executes in a secure environment and access to any resource outside this environment is strictly controlled by a security manager. Sandboxing suffers from an "all-or-nothing" problem that allows either complete access if the signer of the Mobile Agent is trusted or very limited access for all Mobile Agents

- Joseph and Luis (1996) proposed a technique to protect the platform by agent code signing process. It is to authenticate the incoming agent by the platform. The agent code will be digitally signed by the agent owner, for the authenticity of an agent, its origin and its integrity. If the host trusts the signer of the Mobile Agent, it will allow it to carry out its execution with full access to all the resources available in the execution environment.

- Path History (Ordille et al 1995, Ordille 1996) is a system, in which the authenticable record of the platforms will be maintained by an agent visited previously. Based on those records, the newly visited platform can determine whether to process the agent or not. Path history includes the signed identity of each platform visited by the agent, and the identity of the next platform to be visited. The path history could be very lengthy for large distributed systems and thereby increase the cost of verification.

- Kumari et al (2007) introduced a System Agent (SA) for each platform. The Mobile Agent has to request the SA for every migration to a destination system. The System Agent of the local host contacts the System Agent of the remote host and verifies or correlates the policies. The SA of the local host serializes the Mobile Agent to make its control migration to a System Agent of the remote host. The SA in the remote host receives Mobile Agent and de-serializes it and also initiates the execution of the Mobile Agent from its current state. It has the drawback of having the system agent in each platform and communicates with the destination system before dispatching the agent. Time taken for Certificate verification and communication before dispatching the agent is very high.

Many techniques for protecting mobile agents have been proposed such as :

- Code obfuscation (Libes 1992) is performed to make the agent program illegible and thus making it difficult to manipulate. The mobile code is shuffled before it is moved to a remote site.

- black box (Hohl 1998) security is proposed to protect the mobile code against malicious hosts Hohl has proposed several conversion algorithms to generate a new agent (out of an original agent), which differs in code and representation but yields the same results.

- A State Appraisal (Farmer et al 1996) approach is proposed to identify the alterations of the agent's state information by malicious attacks.The author or owner of the agent has to create the appraisal functions which could be added to the agent's code

- Vigna (1997) developed a system to identify the malicious modifications on the agent code. The platform where the agents execute is required to create and retain a non-repudiation log for the operations performed by the agent and to submit a cryptographic trace. A trace consists of a sequence of statement identifiers and platform signature information. If any malicious results occur, the appropriate traces and trace summaries can be obtained and verified; then the malicious host can be identified.

- Computing with the encrypted function (Sander and Christian 1998) is to execute the agent (program) as an enciphered function without being able to discern the original function; i.e., instead of preparing an agent with function f, the agent owner can give the agent program P(E(f)) which implements E(f), an encrypted version of f. An agent's execution could be kept secret from the executing host as would any information carried by the agent.

- The Factor of Time (Grimley and Monroe 1999) is to identify the malicious host based on the period the agent is occupied by the hosts. If we provide a limited time to execute the agent, then the chance to tamper with the code is limited. If the time elapsed is more in the untrusted host, the agent must shut down or move to the next host specified on its itinerary.

- Benachenhou et al (2006) developed a system to protect the Mobile Agent with the help of the clone available in the trusted server. The Mobile Agent that visited the host has to be compared with the clone and authenticate its integrity

Lack of early detection of attacks, complex cipher computations and Overloaded mobile code information, necessitate development of new security mechanisms for Mobile Agent networks.

In this work, mechanisms have been suggested that help the migrating crawlers to become more secure and robust to possible attacks.

III. PROPOSED SECURITY MECHANISM

Owing to inherent security problems in mobile agents, following security proposals are being made for protecting the migrant as well the host.

A. Host Protection: The SecCheck Manager is used to secure the host from malicious migrant. It uses two major components namely Identity Checker(IC) and Code Checker (CC) to protect the host as shown in Fig1.

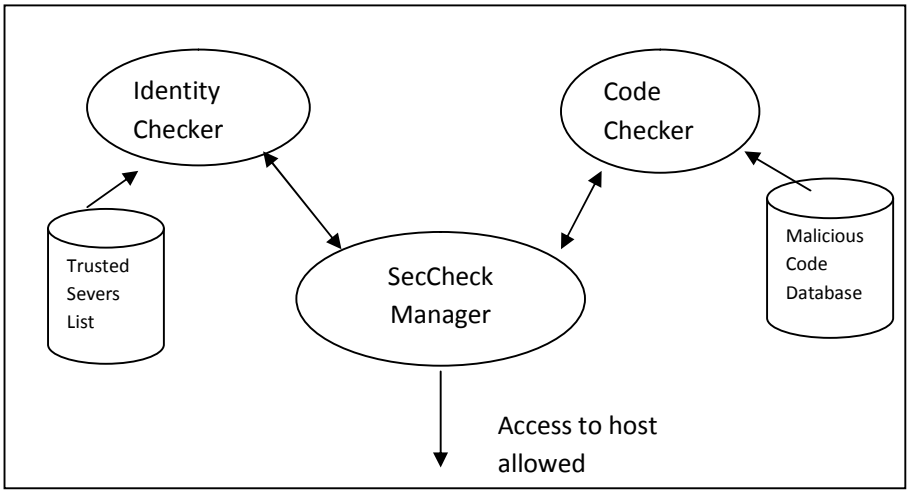

Fig1: SecCheck Module

The migrant has to pass through the SecCheck Manager module and only after successful approval from the security module, can get access to the host platform. The SecCheck manager waits for pass signals from Identity Checker and Code Checker module. After receiving the signals , a migrant is allowed to reside on the host. The algorithm for SecCheck manager is given in Fig. 2

```
Host Security Manager ()
Step 1: wait for migrant
2: call Identity checker module (id, past_visit_list );
3: wait (migrant_authenticated);
4: call code checker module ();
5: wait (safe)
6: If (safe& migrant_authenticated)
6.1: Allow migrant to enter in to the host
Else
6.2: Access to the host is denied
```

Fig. 2: Algorithm for Host Security Manager

- Identity Checker : A migrant carries encrypted key encrypted by private key of the crawler manager and a list of ids of intermediate nodes visited. As the agent reaches the remote host, Identity Checker module is invoked to verify the identity of the migrant. The encrypted information is decrypted using public key of the crawler manager. If the two keys match , the identity checker module then check the host entries in the itinerary information in the trusted server list. If all the entries match with the values in the database, authenticity of the agent is verified and Identity Checker module sends an authenticated signal to the SecChecker Manager. The algorithm for Identity Checker module is shown in Fig 3

```
Identity Checker

Step 1: At the receiving host verify the id of the crawler manager
2: If the two keys match , check listof_visited_nodes
3: If all nodes match entries in trusted server list
3.1: signal(migrant_authenticated);
3.2:access to host allowed
Else
3.3 access to host denied
```

Fig 3: Algorithm for Identity Checker

- Code Checker : The code checker module scans the byte code and data to detect malicious instructions by comparing it with the available malicious codes stored in the server. A malicious agent migrated from one machine to another machine may have illegal codes like accessing a prohibited database, killing another agent, shutting down the platform, cloning more number of agents, etc. The code checker performs linear search to scan the byte code in order to search for the string (malicious code) in the set of byte codes. If any malicious code is found within the byte code then the mobile agent is killed by the remote host and host-id is removed from trusted server list. The failure signal is sent to the crawler manager. If no such code is found, then migrant is granted permission to access web server pages.

The successful execution of Host Security module guarantees host security and protects it from security attacks.

B. Migrant Protection : A web server can actively or passively attack a migrant. In a passive attack the server extracts information of the migrant and in an active attack host intercepts and modifies the data of the migrant. A novel protection mechanism based on credibility value is proposed that secures migrant against both types of attacks. An agent who is more credible is allowed to execute freely, with more resources and permissions. A Migrant will be transferred from low level to high level of permissions if value of Credibility increases incrementally. Permissions may be revoked if credibility decreases in successive executions. The permissions with which an agent can operate are listed below :

Level	Permission
I	Agent is allocated separate address space.
II	Agent can access or crawl web server pages.
III	Agent can process information at web server side.
IV	Agent is allowed to dispatch information to its Crawler Manager.

A Monitor with its two components namely Execution Checker and Restorer, runs at penultimate host, and observes the behaviour of the executing migrant at different time intervals and based on its observations assigns credibility value to the migrant. If any suspicious activity is detected , the saved image of the migrant is restored and send to the Crawler Manger of the migrant. The crawler manager then adds the remote host to non trusted server list.

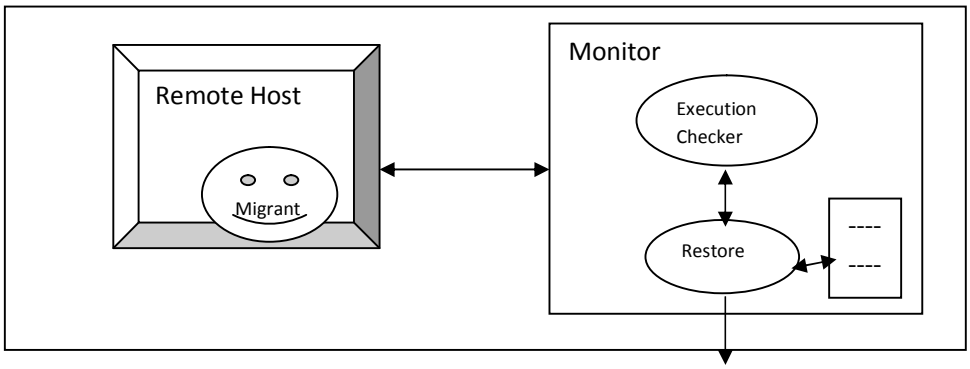

Fig 4: Modules for Migrant Protection

- Execution Checker: It observes the behaviour of the executing migrant through remote Following factors are compared to determine whether the executing migrant is safe or unsafe .
-

CAA	Change in attributes of Migrant
CSC	Change in size of Migrant
CAP	Attempt to access read-only data or change in access permissions
CRR	Change in Resource request
CET	Change in Execution time

Initially, if the migrant passes SecCheck Module , the system is considered to be safe and migrant is given a minimum credibility score (say 0.25) and is allowed to execute with Level 1 permissions. As the migrant continues its execution , the security parameters are checked after every nth interval for system security . After every nth interval if the syetm is in safe state and if the parameters are found to be intact the credibility value is increased by ΔA, where ΔA is 50% of previous credibility value and accordingly the migrant is allowed to execute with higher levels of permissions . If any change in security parameters is encountered , the credibility value is decreased by same amount. However if at any time interval the system is found to be unsafe, migrant is agent is terminated and the Restorer module is invoked to send stored image back to the migrant crawler. The algorithm for Execution Checker module is given below :

```
Step 1: If sec_check module=pass, assign cred_score=0.25
2: Migrant is allocated separate address space and both server and Migrant are
considered safe.
3: After nth Time interval ,
3.1  if (!CAA && !CSC), system is safe
3.1.1  if( (!CAP&&!CRR&&!CET) && cred_score<=0.25))
3.1.1.1          cred_score=cred_score+ΔA
3.1.1.2        if(cred_score<=0.5)
3.1.1.3        migrant executes with Level II Permission
3.1.1.4        else
3.1.1.5        if cred_score>0.5 and cred_score<=0.7
3.1.1.6        migrant executes with Level III Permissions
3.1.1.7        else
3.1.1.8        migrant executes with Level IV Permissions
3.1.2  if ((!CAP && CRR ||CET ) && (cred_score<=0.25))
3.1.2.1        cred_score=cred_score-ΔA
3.1.2.2        wait(1000 sec) and goto step 3.1.2
3.1.3  else terminate migrant and send failure signal to Restorer

3.2    if(CAA || CSC), system is unsafe
3.2.1  send failure signal to Restorer
3.2.2  terminate the migrant
```

Restorer*:* It creates an image of the migrant by storing its code and state at a temporary storage before sending the migrant to the next host. The state of a migrant is defined as the data carried by a migrant. It waits on the signal failure from the Execution checker module. As soon as it receives the failure signal it re-creates the migrant from the stored image and sends it crawler manager. The algorithm for the restore component is given in Fig 5

```
Restore ()
Step 1:create an image of migrant;
2.wait (failure;
3. send the migrant to the crawler manager
```

Fig 5: Algorithm for Restorer

The proposed Migrant protection mechanism manages migrant consistency with restriction.When migrant is not reliable the restriction is maximum and when the agent is reliable the migrant executes freely with all requested resources.

IV. ANALYSIS

The following examples show computation of credibility of a migrant and allotment of permissions by taking different datasets.

Case 1 : when the migrant arrives at web server , Identity Checker Module checks the id and code checker module tests the byte code of the migrant . If both are found OK , cred_score =0.25.

Migrant is allowed to execute with Level- I permission i.e. Migrant is allowed to reside on the Web server. The execution checker module then monitors the execution of migrant at 1st time interval and all parameters i.e Attributes, size , resource requested and access permissions are found same then

Cred_score = 0.25 +0.5(0.25)= 0.375

Migrant continues with Level-I permissions

At time interval 2 again all parameters i.e Attributes, size , resource requested and access permissions are found same then

Cred_score= 0.375+0.5(0.375)= 0.5675

Migrant now executes with Level II permissions.

Case2: At 3rd time interval , change in execution time is encountered then

cred_score=0.5675-0.5(.5675) = 0.3 , therefore Level-II permissions are reverted.

Case 3: At nth interval change in size and attributes are encountered, Migrant is terminated and stored image is sent to the crawler manager

Based on the above cases it is can be seen that when migrant is not reliable the restriction is maximum and when the agent is reliable the restriction are released and it can execute freely with requested resources. So as the credibility increases, the restriction of the environment decreases, as shown in Figure 6

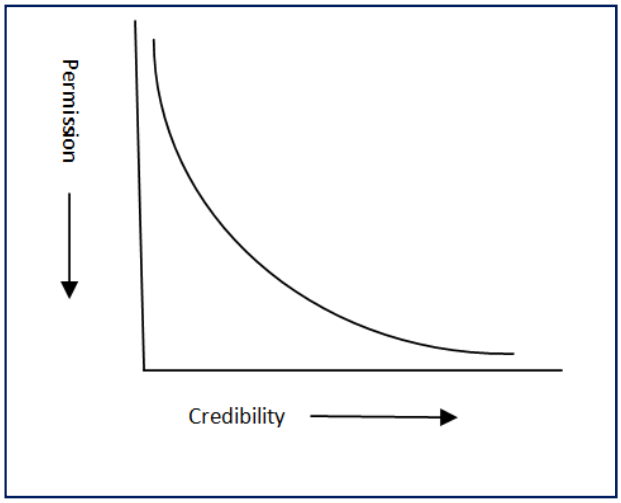

Fig 6 : Permission variation with Credibility

V. CONCLUSION

A novel credibility based security approach is proposed to protect the mobile agent as well as the host platform. The agent is validated with the help of SecCheck module .After the validation is successful, the migrant is allowed to perform its computation depending on dynamic computation of its credibility value. Migrant permissions to use host platform may be decreased (or increased) in incremental manner as reliability of the migrant increases or decreases.

REFERENCES

1. Ashutosh Dixit, Harish Kumar and A.K Sharma, "Self Adjusting Refresh Time Based Architecture For Incremental Web Crawler", International Journal of Computer Science and Network Security (IJCSNS), Vol 8, No12, Dec 2008.
2. Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile Agents: Are they a good idea? Technical report, IBM, March 1995. URL http://www.research.ibm.com/massdist/mobag.ps.
3. D. M. Chess. Security Issues in Mobile Code Systems. In G. Vigna,editor, Mobile Agents and Security, volume 1419 of LNCS, pages 1–14.
4. Springer-Verlag, June 1998. Wayne Jansen and Tom Karygiannis, "Privilege Management Mobile Agents, Twenty-third National Information Systems Security Conference, pp.362-370, October 2006.
5. Niraj Singhal, R. P. Agarwal, Ashutosh Dixit, A. K. Sharma," Information Retrieval from the Web and Application of Migrating Crawler", International Conference on Computational Intelligence and Communication Systems,2011, pp. 476-480
6. Huhns, M., Singh, M. 1999, "Readings in AGENTS, Morgan Kaufman Publishers, San Francisco, California.
7. Groot, D.R.A. De, Boonk, M.L., Brazier, F.M.T, Oskamp, A, "Issues in a Mobile Agent-based Multimedia Retrieval Scenario" , Proc. The Third European Workshop on Multi-Agent Systems (EUMAS'05), pp. 103-113.
8. Srilekha Mudumbai, Abdeliah Essiari, William Johnston, "Anchor Toolkit: A Secure Mobile Agent System," Proceedings of Mobile Agents '99 Conference, October 1999.
9. William Farmer, Joshua Gutt an, Vipin Swarup, "Security for Mobile Agents: Authentication and State Appraisal," Proceedings of the Fourth European Symposium on Research in Computer Security (ESORICS '96), September 1996, pp. 118-130.
10. F. Gasparetti, A. Micarelli, "Swarm Intelligence: Agents for Adaptive Web Search", Dept. of Information, University of ROMA TRE, Rome, Italy, 2000.
11. Fiedler J. and Hammer J., "Using the Web Efficiently: Mobile Crawling," in Proceeding of the 7th International Conference of the Association of Management (AoM/IAoM) on Computer Science, CA,pp. 324-329, 1999.
12. Fiedler J. and Hammer J., "Using Mobile Crawlers to Search the Web Efficiently," International Journal of Computer and Information Science, vol. 1, no. 1, pp. 36-58, 2000.
13. Nisha Pahal, Sunil Kumar, Ashu Bhardwaj,Naresh Chauhan, "Security on Mobile Agent Based Crawler(SMABC)", International Journal of Coputer Applications(0975-8887)Vol 1-No.15, 2010..
14. F. Hohl. Time Limited Blackbox Security. In G. Vigna, editor, Mobile Agents and Security, volume 1419 of LNCS, pages 92–113. Springer- Verlag, June 1998.
15. C. Collberg, C. Thomborson, and D. Low. A taxonomy of obfuscating transformations. Technical Report 148, University of Auckland, 1997.

16. F. Hohl. Time Limited Blackbox Security. In G. Vigna, editor, Mobile Agents and Security, volume 1419 of LNCS, pages 92–113. Springer-Verlag, June 1998.