A Multi – Faceted Approach to Mobile Agent Security

Anthony M. Ngereki
Department of Computer Science
Chuka University, Kenya

Andrew M. Kahonge
School of Computing and Informatics
University of Nairobi, Kenya

ABSTRACT
Mobile agents are increasingly becoming popular in the development of current distributed applications. However, the mobility and autonomy factors of mobile agents present a host of security challenges in a distributed environment. In an attempt to ensure security of the mobile agent against a malicious host, a security framework is proposed. Our security mechanism uses a multi-faceted approach to protect mobile agents and must be incorporated from the design stage of agent systems. We identify major security threats against mobile agents by a malicious platform and propose algorithms to counter them. We then test a multi-agent system that incorporates these algorithms against one that doesn't and compare the results.

General Terms
Multi-agent System, Security, Encryption

Keywords
Agent, Security, Authentication, Authorization, Monitoring, Supervision, Secure Sockets Layer (SSL), Transport Layer Security (TLS), Agent Communication.

1. INTRODUCTION
Mobile agents present an evolution in computing that allows for complete mobility of cooperating applications among supporting platforms to form a large-scale, loosely-coupled distributed system. Though there a number of models that can be used to describe agent systems, a simple model consisting of two components: the agent and the agent platform is sufficient to discuss security in mobile agents. Here, an agent comprises the code and state information needed to carry out some computation. The agent platform provides the computational environment in which an agent operates. [2]

The platform from which an agent originates or is created is known as the home platform and is usually the most trusted. An agent however can move (hop) from one execution environment to another in a network. This new environment is called the host environment and takes over full control over the agent's over agent's code, data and execution state [1][3]. This control of the host over all executing programs makes it difficult to protect mobile agents from malicious hosts [12] and as such exposes them to various security threats.

Basically, the security requirements of any computer system are confidentiality, integrity, authentication, authorization, non-repudiation and availability [6][7][15]. A malicious host environment can compromise the security requirements of a mobile agent in a number of ways. This include denial of service, eavesdropping, interception, alteration, replays and masquerading [6][14][5][7][4].

While techniques such as access control, password protection and sand boxes have been developed to protect agent platforms against hostile agents [16], none of the approaches to protect mobile agents against malicious hosts adequately addresses every aspect of security [17].

2. MOBILE AGENT SECURITY THREATS
Using a simple model comprising of an agent and agent platform, security threats in mobile agent systems can be categorised into four categories namely agent-to-platform, agent-to-agent, platform-to-agent and other-to-agent platform. A simple agent model as described by [2] is shown in figure 1.

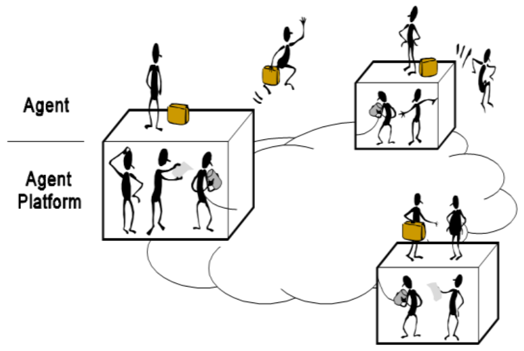


Figure 1. A simple Agent Model

2.1 Categories of Agent Security Threats
Agent-to-Platform: This category category represents the set of threats in which agents exploit security weaknesses of an agent platform or launch attacks against an agent platform. This set of threats includes masquerading, denial of service and unauthorized access.
Agent-to-Agent: The agent-to-agent category represents the set of threats in which agents exploit security weaknesses of other agents or launch attacks against other agents. This set of threats includes masquerading, unauthorized access, denial of service and repudiation. Many agent platform components are also agents themselves.

Platform-to-Agent: The platform-to-agent category represents the set of threats in which platforms compromise the security of agents. This set of threats includes masquerading, denial of service, eavesdropping, and alteration.

Other-to-Agent Platform: The other-to-agent platform category represents the set of threats in which external entities, including agents and agent platforms, threaten the security of an agent platform. This set of threats includes masquerading, denial of service, unauthorized access, and copy and replay.

2.2 Platform-to-Agent Security Threats
An agent is most secure in its home platform since it is where it is instantiated. However, mobility implies that this trusted execution environment needs to be extended to other host

platforms in the agents' itinerary. Such trust is difficult to extend beyond a single hop especially because while the home platform could trust the next host in the network, this bilateral trust is not transitive i.e. just because the home platform (x) trusts the next host (y), this doesn't mean another host in the network (z) also trusts (y). This complexity introduces a multi-hop security problem.

Some of the possible platform-to-agent security threats include the following [6][14][3]:

2.2.1 Denial of Service

An agent platform should faithfully execute an agents' requests, allocate necessary resources and abide by the agreed upon quality of services. However, a malicious agent platform, may ignore agent service requests, introduce unacceptable delays for critical tasks, refuse to execute the agent's code, or even terminate the agent without notification. Non-responsive agents on malicious platforms could either be deadlocked or livelocked.An Agent livelock occurs when an agent is continuously given tasks to perform and can never catch up or achieve its goal.

2.2.2 Masquerade

This happens when a malicious platform claims the identity of another platform that the agent should actually visit. This decieves the agent into giving the malicious host sensitive information. Once the masquerading host is able to gain the trust of the agent, it may then be able to read or modify any of agent's code, data and state. This can be prevented by use of a strong authentication protocol to authenticate a host to an agent. A masquerading platform can harm both the visiting agent and the platform whose identity it has assumed.

2.2.2 Eavesdropping

The fact that an agent must execute on a host means that the host is able to record instructions given to it by the agent. This implies that a malicious host may try to determine the code, data or flow control held by the agent. Even though the agent may not be directly exposing secret information, the platform may be able to infer meaning from the types of services requested and from the identity of the agents with which it communicates. This form of attack is difficult to prevent and detect.

2.2.1 Alteration

A malicious host can alter an agent by changing the data, code and control flow so that the agent performs other tasks than what was intended by it's creator. A mobile agent that visits several platforms on its itinerary is exposed to a new risk each time it is in transit and each time it is instantiated on a new platform. Alteration can be detected by having the original author digitally sign the agent's code. This detection however becomes difficult for agents visiting several platforms (the "mult-hop" problem).

3. PREVIOUS APPROACHES TO MOBILE AGENT SECURITY

According to Lange and Oshima in [18], There are three fundamental security issues specific to mobile agent systems. These are:

- Protecting the host (platform) from the mobile agent.
- Protecting the mobile agent from other mobile agents, and
- Protecting the mobile agent from the host.

Some of the proposals to protect a mobile agent from a malicious host are discussed below:

3.1 Shadow and Primary Agent Approach

This security approach proposed by [11] aims to identify and skip every blocking malicious host in the itinerary of a mobile agent. The sheme relies on an acknowledgement and time-out mechanism to ensure that a mobile agent has visited a host in it's itinerary and safely departed to the next one. It uses two mobile agents; a primary (PA) and a shadow (SA). Normally, SA is lagging one step in the itinerary behind PA.

The assumption is that a host is considered non-blocking should it allow the PA to continue its task and safely depart to the next host. The SA suspects a malicious action if it does not receive an acknowledgement within a proper time-out T after which it requests help from the home host to identify the malicious host and take corrective action.

When the home host identifies the the malicious host, it sends a new instance of the PA to a safe host to meet SA which carries a copy of the collected data. SA will reload the collected data into the empty PA. The newly loaded PA will continue it's itinerary skipping the malicious host.

3.2 Partial Mobility Mechanism

[7] Proposes partial mobility mechanism (PMM) to protect mobile agents integrity and privacy against malicious hosts. In PMM, the mobile agent has two types:

1. A One_Hop_Agent (OHA) that represents tasks to be executed in an untrusted host and can only visit one host.
2. The Multi-Hop-Agent (MHA) that represents tasks to be executed in trusted hosts. The MHA can visit multiple trusted hosts.

To represent one MA, PMM needs only one MHA and at least one OHA which is embedded in the MHA. In PMM, the Agent's home platform creates an agent and determines all the hosts to be visited by the agent. Hosts are classified as either trusted (serves MHAs only) or untrusted (serves OHAs only).

One backdrop of this mechanism is that an agent's itinerary is known priori which compromises the autonomy property of an agent. It is also difficult to keep track of the security status of hosts in a distributed network. The use of the One_Hop_Agent (OHA) also limits mobility of an agent and therefore abuses the undelying concept of a mobile agent.

The architecture of this mechanism is shown in figure 2 below.

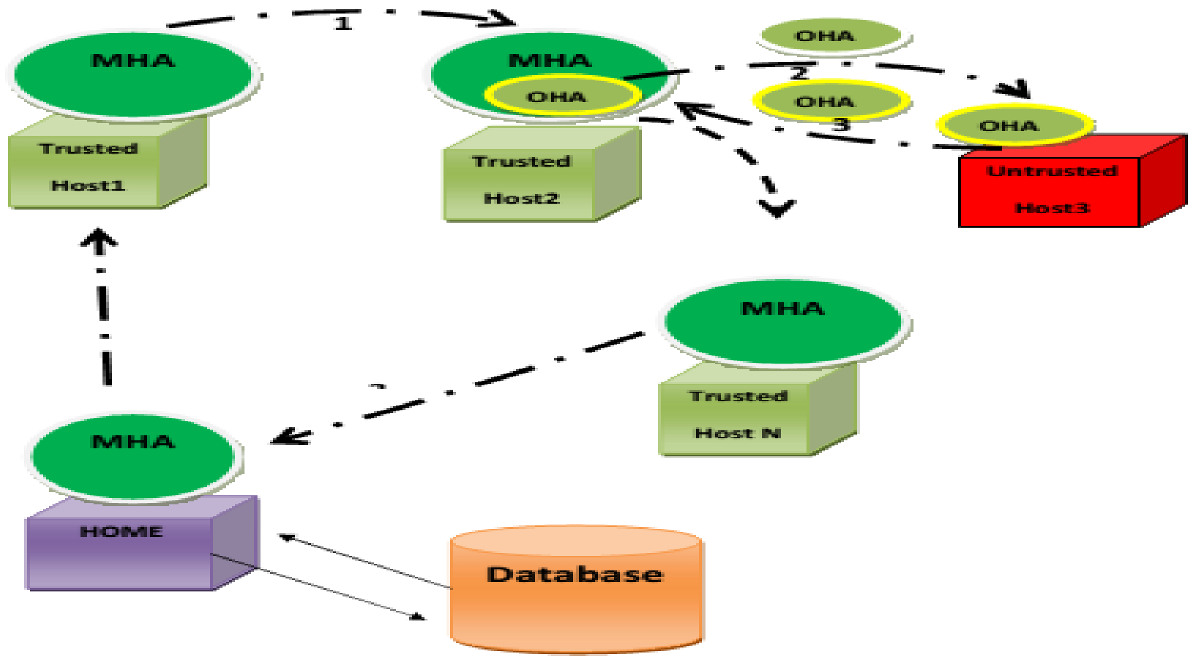


Fig. 2: PMM Architecture (Proposed by [7]

3.3 The Secure-Image Mechanism

The main objective of Secure-Image Mechanism as proposed by [4] is to protect the mobile agents against malicious hosts. SIM generates a secure image for the mobile agent before it arrives to the hosts that are classified as untrust hosts. The mechanism prevents eavesdropping and alteration attacks.

If the next host in the agent's itinerary is untrust, the agent visits the near Secure-Image Controller (SIC) which generates a secure image of the agent and sends it to the untrusted host. This protects the original agent from visiting malicious hosts. Alteration is detected by generating a digest of data of tasks implemented in untrusted hosts and comparing with a digest of the original data.

Just as is the case with PMM discussed in 3.1 above, this mechanism requires that trusted and untrusted hosts are known a priori which is difficult in a distributed system. Figure 3 presents a Mobile Agent System with SIM.

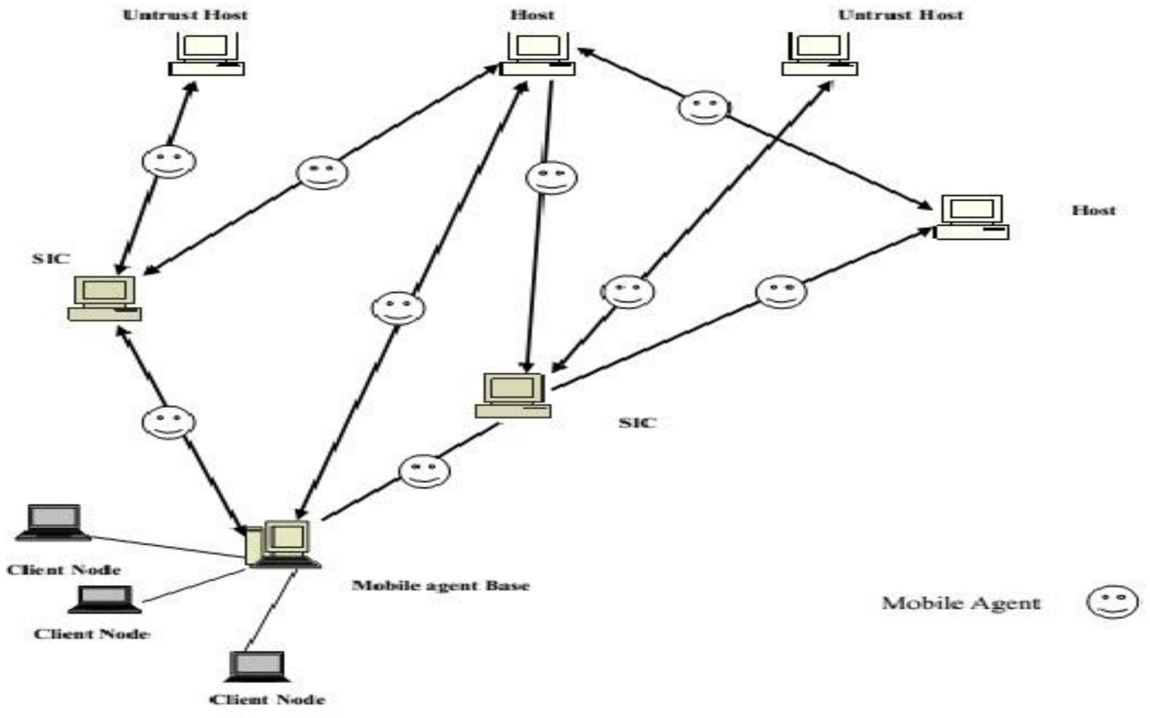


Fig. 3 Mobile Agent System with SIM (Proposed by [4])

3.4 Keylets Mechanism

This mechanism presented by [19] protects a mobile agent's code. It partitions the mobile agent code and state information into self-contained components which are then encrypted using symmetric keys and made available to platforms that will host the mobile agent in the network. A Keylet is a specific type of mobile code that determines the distribution of keys to platforms.

This mechanism however suffers various drawbacks. First, code partitioning is done by a third party code producer who supplies the mobile agent as a template to the agent owner. Secondly, a large number of transactions are related to the Keylet and a host may not be willing to support the increased computation. Thirdly, key revocation is not good in quality and requires a complicated mechanism to categorize tasks of the mobile agent. Finally, this mechanism does not protect the mobile agent code completely.

3.5 The Ajanta Mechanism

The mechanism proposed by [20]. Is used for mediating access to system-level resources. It protects hosting resources through an ad hoc security manager that uses identity-based access control lists to grant or deny agent access. For application-defined resources, Ajanta uses a proxy-based mechanism where a proxy intercepts agent requests and denies or grants based on its own security policy and on the agent's credentials. Ajanta security has a few weaknesses:

1. The proxy generator would have to be rather intelligent to create a dynamic policy that would also ensure that whatever it is bypassing is still secure.
2. Key distribution mechanism is not covered in this mechanism.
3. Sending information back the agent owning server is sometimes necessary but not always possible.
4. Ajanta provides a mechanism of spying on the agents and to replay the agent to create their own for piracy. Such requests can be logged and be used to create an agent based on its actions.

4. A MULTI-FACETED APPROACH TO MOBILE AGENT SECURITY

Most mobile agent security mechanisms proposed only detect rather than protect [9][11]. Literature reveals that a single approach cannot protect a mobile agent from all the security challenges that dog this exciting paradigm [8][10]. We therefore propose a multi-faceted approach to dealing with security threats in mobile agent systems. Such a solution must be implemented right from the design stage to implementation if a mobile agent system has to be truly secure. Sections 4.1 to 4.3 describe the algorithms that we have embedded in our mobile agent code to mitigate against threats posed by a malicious host.

4.1 Protection Against Blocking and Denial of Service

Using time to live (TTL) and heart beat mechanisms, a mobile agent can be protected against blocking or denial of the service by a malicious host. The following steps describe how the mechanism detects and protects against blocking or denial of service.

```
While not Successful;
    1.      Parent Agent
        a)  Creates a new agent (Either in the present or alternative host)
        b)  The created Agent:
                Checks resource availability (If enough resources, Acknowledge ability to execute else dies)
        c)  Transfer data to the created agent
        d)  Sets TTL and Starts timer
    2.  While child Agent Not Done (Sends heartbeat to parent)
    3.  While Timer < TTL (Waits for result from child agent)
    4.          If Timer >= TTL (Attempt to destroy child agent)
    5.  Else If Not Successful (Cut off communication from child agent)
    6.  If there is a free agent (Assign task of child agent to free agent)
    7.          Else ( Create another child agent in another host)
```

4.2 Protection Against Masquerading

We have achieved this by centralizing the security requirements of any agent to itself. This implies that when an agent desires to migrate, it creates its instance in a new host and kills itself in the current host. It also informs its parent host of this move. This completely abstracts the use of host identity in the migration process making it difficult for a host to masquerade as another. Further to this requirement, a parent agent assigns a child agent ID for each of the children it creates.

This ID is irreplicable as it uses a one-way hashing algorithm. The parent agent keeps a repository of all IDs of the children it has created and tries to match this with the ID of any agent that tries to communicate with it. If a match cannot be found, communication is denied. Communication between a parent agent and a child is that a child must identify itself when communicating with the parent or any other agent in the network while a parent agent must not necessarily identify itself with its children.
The following steps describe the process of generating an irreplicable child ID:

1. Call a hashing function (We propose a 256 bit SHA1 algorithm)
2. Call a timestamp retrival function.
    - Generate current timestamp (Computed to nanosecond precision)
    - Host timezone
3. Call an agent ID generator function (Concatenates Hash value + Timestamp + Timezone)

This process creates a unique ID for any agent created. The reasoning here is that even in a synchronized environment, it is difficult for a malicious host to precisely fake an ID with precision to the nanosecond.

4.3 Protection Against Eavesdropping and Alteration

We use encryption and decryption mechanisms to achieve this. Eavesdropping is only possible in the host environment since in our mechanism, communication between remote hosts is via SSL/ TLS. We further make eavesdropping on an agent's code by using a production ready tool (Scala) in development of our agents. This implies that the agents' code is compiled and not visible to the host environment.

The files and data created during execution are however visible by the host. To counter eavesdropping, we use an encryption algorithm to encrypt the temporary files and data before writing to memory of the host environment and a decryption algorithm whenever we need to read them.
We further protect against alteration by adding a CRC based on our encrypted data and compare it when retrieving data to detect alteration and take corrective actions. The steps below describe how we achieve this:

Writing Temporary Files to Host

1. Encrypt data before writing to host file system.
2. Generate CRC based on data in step 1 above.
3. Write out the data to the file system

Reading Data from Temporary Files

1. Retrieve data.
2. Calculate CRC.
3. Calculate CRCs to see if data is modified
4. If CRC1 is NOT equal to CRC2
    a. Inform the parent
    b. Cease processing

The parent agent then decides whether to transfer task to a free agent or to start the agent in a new host.

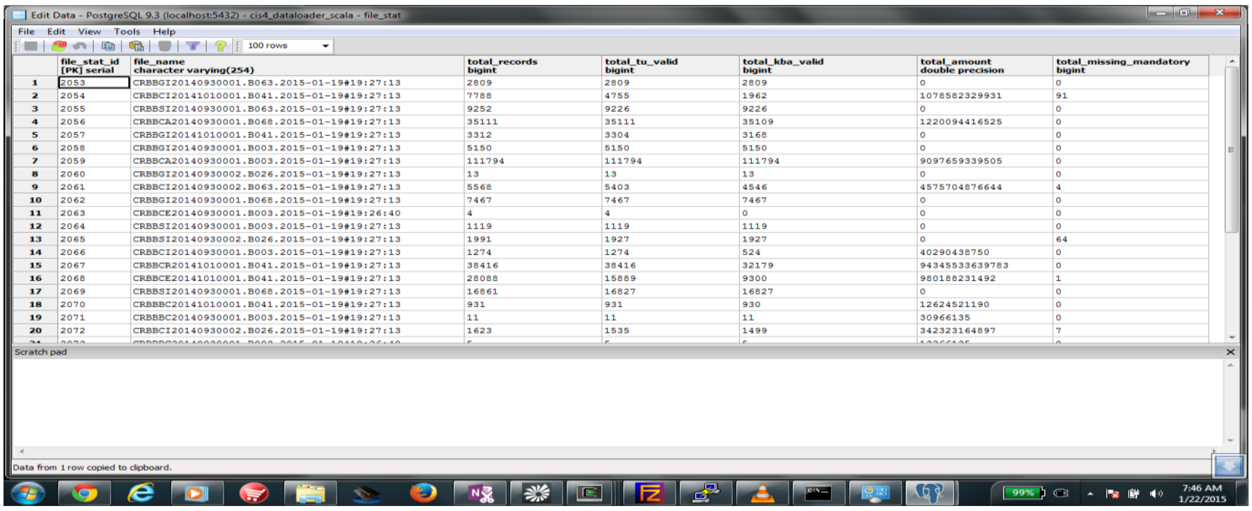
5. EVALUATION AND RESULTS

Any secure computer system must conform to several security requirements namely; confidentiality, integrity, availability and non-repudiation [15]. Using a credit bureau use case, we developed a mobile agent system implementing the approaches discussed in section 4. We then evaluated the system against one without a multi-faceted security approach.

We used a simulated environment with six hosts some of which we purposely sconstructed to corrupt other multi-agent systems' states and data by injecting random data and setting the agents' states to random values. We simulated the hosts by having them listen on different ports.

The logs generated from the tests indicated attacks ranging from interceptions on agent data to deadlocks in the unprotected system while tests with the protected system always yielded expected results.

The diagrams below show the results of the tests conducted.

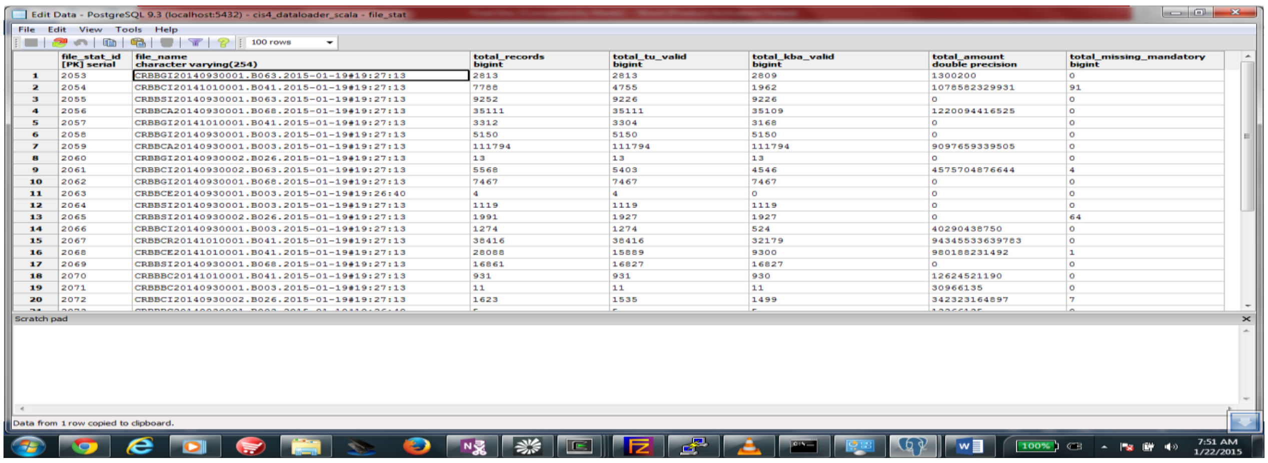

Edit Data - PostgreSQL 9.3 (localhost:5432) - cis4_dataloader_scala - file_stat

	file_stat_id [PK] serial	file_name character varying(254)	total_records bigint	total_tu_valid bigint	total_kba_valid bigint	total_amount double precision	total_missing_mandatory bigint
1	2053	CRBBGI20140930001.B063.2015-01-19#19:27:13	2809	2809	2809	0	0
2	2054	CRBBCI20141010001.B041.2015-01-19#19:27:13	7788	4755	1962	1078582329931	91
3	2055	CRBBSI20140930001.B063.2015-01-19#19:27:13	9252	9226	9226	0	0
4	2056	CRBBCA20140930001.B068.2015-01-19#19:27:13	35111	35111	35109	1220094416525	0
5	2057	CRBBGI20141010001.B041.2015-01-19#19:27:13	3312	3304	3168	0	0
6	2058	CRBBGI20140930001.B003.2015-01-19#19:27:13	5150	5150	5150	0	0
7	2059	CRBBCA20140930001.B003.2015-01-19#19:27:13	111794	111794	111794	9097659339505	0
8	2060	CRBBGI20140930002.B026.2015-01-19#19:27:13	13	13	13	0	0
9	2061	CRBBCI20140930002.B063.2015-01-19#19:27:13	5568	5403	4546	4575704876644	4
10	2062	CRBBGI20140930001.B068.2015-01-19#19:27:13	7467	7467	7467	0	0
11	2063	CRBBCE20140930001.B003.2015-01-19#19:26:40	4	4	0	0	0
12	2064	CRBBSI20140930001.B003.2015-01-19#19:27:13	1119	1119	1119	0	0
13	2065	CRBBSI20140930002.B026.2015-01-19#19:27:13	1991	1927	1927	0	64
14	2066	CRBBCI20140930001.B003.2015-01-19#19:27:13	1274	1274	524	40290438750	0
15	2067	CRBBCR20141010001.B041.2015-01-19#19:27:13	38416	38416	32179	94345533639783	0
16	2068	CRBBCE20141010001.B041.2015-01-19#19:27:13	28088	15889	9300	980188231492	1
17	2069	CRBBSI20140930001.B068.2015-01-19#19:27:13	16861	16827	16827	0	0
18	2070	CRBBBC20141010001.B041.2015-01-19#19:27:13	931	931	930	12624521190	0
19	2071	CRBBBC20140930001.B003.2015-01-19#19:27:13	11	11	11	30966135	0
20	2072	CRBBCI20140930002.B026.2015-01-19#19:27:13	1623	1535	1499	342323164897	7

Scratch pad

Data from 1 row copied to clipboard.

Fig 4. Results of computation in a secure MAS


Edit Data - PostgreSQL 9.3 (localhost:5432) - cis4_dataloader_scala - file_stat

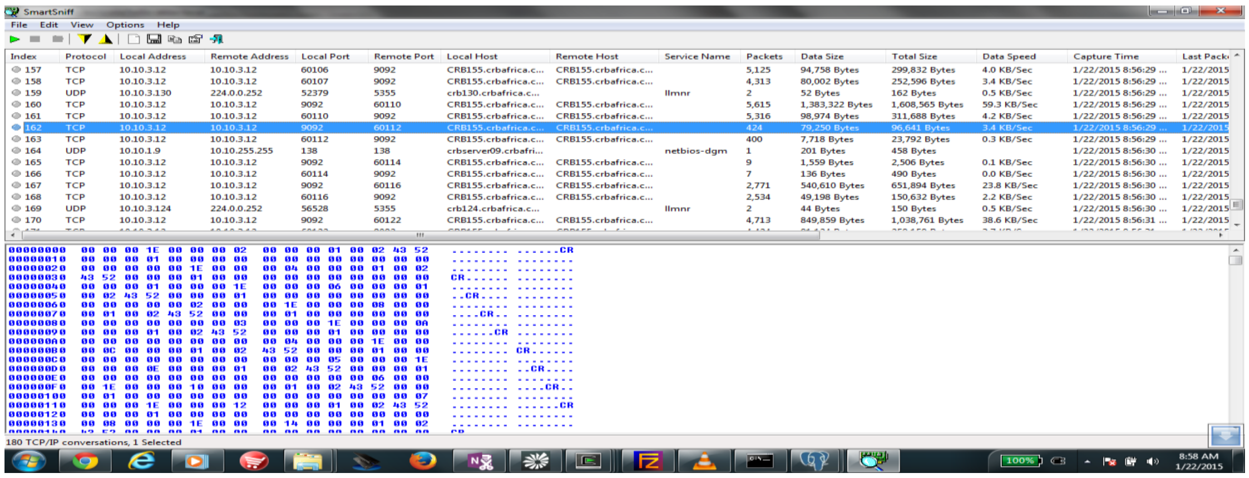
	file_stat_id [PK] serial	file_name character varying(254)	total_records bigint	total_tu_valid bigint	total_kba_valid bigint	total_amount double precision	total_missing_mandatory bigint
1	2053	CRBBGI20140930001.B063.2015-01-19#19:27:13	2813	2813	2809	1300200	0
2	2054	CRBBCI20141010001.B041.2015-01-19#19:27:13	7788	4755	1962	1078582329931	91
3	2055	CRBBSI20140930001.B063.2015-01-19#19:27:13	9252	9226	9226	0	0
4	2056	CRBBCA20140930001.B068.2015-01-19#19:27:13	35111	35111	35109	1220094416525	0
5	2057	CRBBGI20141010001.B041.2015-01-19#19:27:13	3312	3304	3168	0	0
6	2058	CRBBGI20140930001.B003.2015-01-19#19:27:13	5150	5150	5150	0	0
7	2059	CRBBCA20140930001.B003.2015-01-19#19:27:13	111794	111794	111794	9097659339505	0
8	2060	CRBBGI20140930002.B026.2015-01-19#19:27:13	13	13	13	0	0
9	2061	CRBBCI20140930002.B063.2015-01-19#19:27:13	5568	5403	4546	4575704876644	4
10	2062	CRBBGI20140930001.B068.2015-01-19#19:27:13	7467	7467	7467	0	0
11	2063	CRBBCE20140930001.B003.2015-01-19#19:26:40	4	4	0	0	0
12	2064	CRBBSI20140930001.B003.2015-01-19#19:27:13	1119	1119	1119	0	0
13	2065	CRBBSI20140930002.B026.2015-01-19#19:27:13	1991	1927	1927	0	64
14	2066	CRBBCI20140930001.B003.2015-01-19#19:27:13	1274	1274	524	40290438750	0
15	2067	CRBBCR20141010001.B041.2015-01-19#19:27:13	38416	38416	32179	94345533639783	0
16	2068	CRBBCE20141010001.B041.2015-01-19#19:27:13	28088	15889	9300	980188231492	1
17	2069	CRBBSI20140930001.B068.2015-01-19#19:27:13	16861	16827	16827	0	0
18	2070	CRBBBC20141010001.B041.2015-01-19#19:27:13	931	931	930	12624521190	0
19	2071	CRBBBC20140930001.B003.2015-01-19#19:27:13	11	11	11	30966135	0
20	2072	CRBBCI20140930002.B026.2015-01-19#19:27:13	1623	1535	1499	342323164897	7

Scratch pad

Data from 1 row copied to clipboard.

Fig 5. Results of computation in unsecured MAS

Figure 5 shows the result of computation on the same sets of files as those shown in figure 4. Unlike in figure 4, however, the multi-agent system that produced this output did not implement our proposed security mechanism. As such, another multi-agent system was able to introduce new records in the processing pipeline of the multi-agent system. The result, as illustrated by record 1 in figure 5, is an output that includes records and balance amount measures that were not contained in the original files.



SmartSniff

Index	Protocol	Local Address	Remote Address	Local Port	Remote Port	Local Host	Remote Host	Service Name	Packets	Data Size	Total Size	Data Speed	Capture Time	Last Pack
157	TCP	10.10.3.12	10.10.3.12	60106	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		5,125	94,758 Bytes	299,832 Bytes	4.0 KB/Sec	1/22/2015 8:56:29 ...	1/22/2015
158	TCP	10.10.3.12	10.10.3.12	60107	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		4,313	80,002 Bytes	252,596 Bytes	3.4 KB/Sec	1/22/2015 8:56:29 ...	1/22/2015
159	UDP	10.10.3.130	224.0.0.252	52379	5355	crb130.crbafrica.c...		llmnr	2	52 Bytes	162 Bytes	0.5 KB/Sec	1/22/2015 8:56:29 ...	1/22/2015
160	TCP	10.10.3.12	10.10.3.12	9092	60110	CRB155.crbafrica.c...	CRB155.crbafrica.c...		5,615	1,383,322 Bytes	1,608,565 Bytes	59.3 KB/Sec	1/22/2015 8:56:29 ...	1/22/2015
161	TCP	10.10.3.12	10.10.3.12	60110	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		5,316	98,974 Bytes	311,688 Bytes	4.2 KB/Sec	1/22/2015 8:56:29 ...	1/22/2015
162	TCP	10.10.3.12	10.10.3.12	9092	60112	CRB155.crbafrica.c...	CRB155.crbafrica.c...		424	79,250 Bytes	96,641 Bytes	3.4 KB/Sec	1/22/2015 8:56:29 ...	1/22/2015
163	TCP	10.10.3.12	10.10.3.12	60112	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		400	7,718 Bytes	23,792 Bytes	0.3 KB/Sec	1/22/2015 8:56:29 ...	1/22/2015
164	UDP	10.10.1.9	10.10.255.255	138	138	crbserver09.crbafri...		netbios-dgm	1	201 Bytes	458 Bytes		1/22/2015 8:56:30 ...	1/22/2015
165	TCP	10.10.3.12	10.10.3.12	9092	60114	CRB155.crbafrica.c...	CRB155.crbafrica.c...		9	1,559 Bytes	2,506 Bytes	0.1 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
166	TCP	10.10.3.12	10.10.3.12	60114	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		7	136 Bytes	490 Bytes	0.0 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
167	TCP	10.10.3.12	10.10.3.12	9092	60116	CRB155.crbafrica.c...	CRB155.crbafrica.c...		2,771	540,610 Bytes	651,894 Bytes	23.8 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
168	TCP	10.10.3.12	10.10.3.12	60116	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		2,534	49,198 Bytes	150,632 Bytes	2.2 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
169	UDP	10.10.3.124	224.0.0.252	56528	5355	crb124.crbafrica.c...		llmnr	2	44 Bytes	150 Bytes	0.5 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
170	TCP	10.10.3.12	10.10.3.12	9092	60122	CRB155.crbafrica.c...	CRB155.crbafrica.c...		4,713	849,859 Bytes	1,038,761 Bytes	38.6 KB/Sec	1/22/2015 8:56:31 ...	1/22/2015

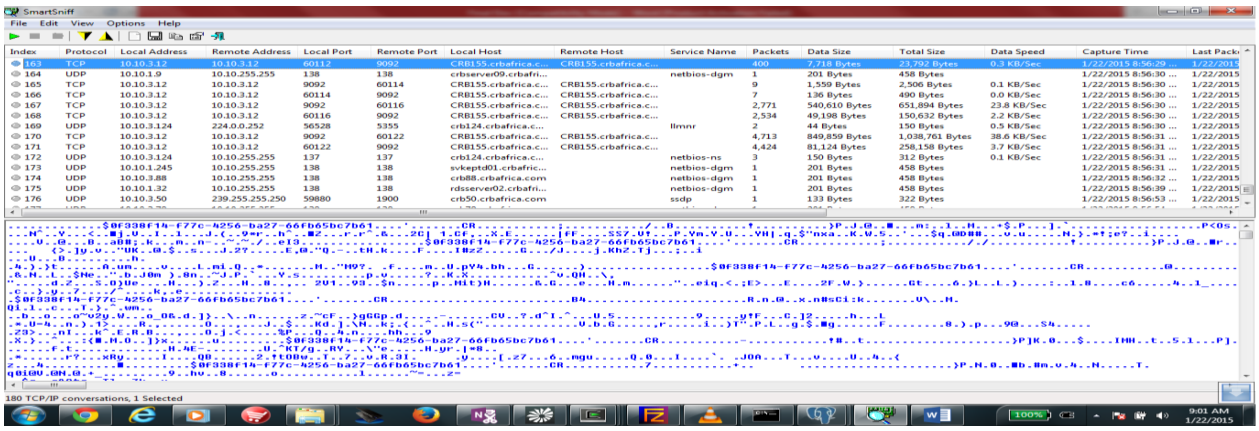
```
00000000  00 00 00 1E 00 00 00 02  00 00 00 01 00 02 43 52   ........ ......CR
00000010  00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000020  00 00 00 00 00 1E 00 00  00 04 00 00 00 01 00 02   ........ ........
00000030  43 52 00 00 00 01 00 00  00 00 00 00 00 00 00 00   CR...... ........
00000040  00 00 00 01 00 00 00 1E  00 00 00 06 00 00 00 01   ........ ........
00000050  00 02 43 52 00 00 00 01  00 00 00 00 00 00 00 00   ..CR.... ........
00000060  00 00 00 00 00 02 00 00  00 1E 00 00 00 08 00 00   ........ ........
00000070  00 01 00 02 43 52 00 00  00 01 00 00 00 00 00 00   ....CR.. ........
00000080  00 00 00 00 00 00 00 03  00 00 00 1E 00 00 00 0A   ........ ........
00000090  00 00 00 01 00 02 43 52  00 00 00 01 00 00 00 00   ......CR ........
000000A0  00 00 00 00 00 00 00 00  00 04 00 00 00 1E 00 00   ........ ........
000000B0  00 0C 00 00 00 01 00 02  43 52 00 00 00 01 00 00   ........ CR......
000000C0  00 00 00 00 00 00 00 00  00 00 00 05 00 00 00 1E   ........ ........
000000D0  00 00 00 0E 00 00 00 01  00 02 43 52 00 00 00 01   ........ ..CR....
000000E0  00 00 00 00 00 00 00 00  00 00 00 00 00 06 00 00   ........ ........
000000F0  00 1E 00 00 00 10 00 00  00 01 00 02 43 52 00 00   ........ ....CR..
00000100  00 01 00 00 00 00 00 00  00 00 00 00 00 00 00 07   ........ ........
00000110  00 00 00 1E 00 00 00 12  00 00 00 01 00 02 43 52   ........ ......CR
00000120  00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000130  00 08 00 00 00 1E 00 00  00 14 00 00 00 01 00 02   ........ ........
```

180 TCP/IP conversations, 1 Selected

8:58 AM
1/22/2015

Fig 6. Communication traffic in unsecured MAS

Figure 6 shows intercepted communication traffic in a multi-agent system which has not implemented our security mechanism. As shown, the data can be easily converted into readable format with appropriate conversion tools.



SmartSniff

Index	Protocol	Local Address	Remote Address	Local Port	Remote Port	Local Host	Remote Host	Service Name	Packets	Data Size	Total Size	Data Speed	Capture Time	Last Pack
163	TCP	10.10.3.12	10.10.3.12	60112	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		400	7,718 Bytes	23,792 Bytes	0.3 KB/Sec	1/22/2015 8:56:29 ...	1/22/2015
164	UDP	10.10.1.9	10.10.255.255	138	138	crbserver09.crbafri...		netbios-dgm	1	201 Bytes	458 Bytes		1/22/2015 8:56:30 ...	1/22/2015
165	TCP	10.10.3.12	10.10.3.12	9092	60114	CRB155.crbafrica.c...	CRB155.crbafrica.c...		9	1,559 Bytes	2,506 Bytes	0.1 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
166	TCP	10.10.3.12	10.10.3.12	60114	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		7	136 Bytes	490 Bytes	0.0 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
167	TCP	10.10.3.12	10.10.3.12	9092	60116	CRB155.crbafrica.c...	CRB155.crbafrica.c...		2,771	540,610 Bytes	651,894 Bytes	23.8 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
168	TCP	10.10.3.12	10.10.3.12	60116	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		2,534	49,198 Bytes	150,632 Bytes	2.2 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
169	UDP	10.10.3.124	224.0.0.252	56528	5355	crb124.crbafrica.c...		llmnr	2	44 Bytes	150 Bytes	0.5 KB/Sec	1/22/2015 8:56:30 ...	1/22/2015
170	TCP	10.10.3.12	10.10.3.12	9092	60122	CRB155.crbafrica.c...	CRB155.crbafrica.c...		4,713	849,859 Bytes	1,038,761 Bytes	38.6 KB/Sec	1/22/2015 8:56:31 ...	1/22/2015
171	TCP	10.10.3.12	10.10.3.12	60122	9092	CRB155.crbafrica.c...	CRB155.crbafrica.c...		4,424	81,124 Bytes	258,158 Bytes	3.7 KB/Sec	1/22/2015 8:56:31 ...	1/22/2015
172	UDP	10.10.3.124	10.10.255.255	137	137	crb124.crbafrica.c...		netbios-ns	3	150 Bytes	312 Bytes	0.1 KB/Sec	1/22/2015 8:56:31 ...	1/22/2015
173	UDP	10.10.1.245	10.10.255.255	138	138	svkeptd01.crbafric...		netbios-dgm	1	201 Bytes	458 Bytes		1/22/2015 8:56:31 ...	1/22/2015
174	UDP	10.10.3.88	10.10.255.255	138	138	crb88.crbafrica.com		netbios-dgm	1	201 Bytes	458 Bytes		1/22/2015 8:56:32 ...	1/22/2015
175	UDP	10.10.1.32	10.10.255.255	138	138	rdsserver02.crbafri...		netbios-dgm	1	201 Bytes	458 Bytes		1/22/2015 8:56:39 ...	1/22/2015
176	UDP	10.10.3.50	239.255.255.250	59880	1900	crb50.crbafrica.com		ssdp	1	133 Bytes	322 Bytes		1/22/2015 8:56:53 ...	1/22/2015

```
[illegible]
```

180 TCP/IP conversations, 1 Selected

9:01 AM
1/22/2015

Fig 7. Communication traffic in unsecured MAS

Figure 7 shows the same traffic but in an environment that has implemented our security mechanism. As shown, traffic data is encrypted and thus cannot be easily interpreted.

6. CONCLUSION

This work aimed at proposing a security approach that addresses most of security threats that dog multi-agent systems especially from malicious hosts.. Security threats in a multi-agent system come in a number of ways; a hosting environment may try to exploit agents resident on it, a malicious person may launch a man-in-the-middle attack on a multi-agent system, a hosting environment may attempt, deliberately or otherwise, to corrupt an agent's data or state, and a malicious agent may deliberately feed an agent the wrong data on which to act on.

A secure multi-agent system is the one that can protect both its internal state, data its working on, and the data it produces. Internal state of an agent refers to the agent's attribute values. If these values can be changed by an external program in unpredictable ways, then such an agent is not secure and, therefore, is not reliable. Agents act on data and produce data that correspond to results of their computation. Once an agent gets data, it is the agent's responsibility to guard that data for the duration of its computation. Any results from computation should also be protected from accidental or deliberate manipulation from any other agent. If an agent is incapable of guaranteeing the security of the data it's acting on, then such an agent cannot be relied upon to produce correct computation results.

To achieve security in multi-agent systems, a multi-faceted approach to security must be adopted. This multi-faceted approach should start from requirements engineering phase all the way to maintenance of the developed system This approach makes agent execution faster by localizing security controls. It also makes the system robust because an exception to the top level agent is escalated up the agent hierarchy until a definite resolution is taken.

The future scope of this idea would be integration of the mechanism in well known mobile agent development tools and platforms such as java.

6.1 Further Work

This research project concentrated on securing agent systems to mitigate against security threats imposed by malicious executing environments. The approach used mitigates against majority of such threats. Further study can be done to improve this work in the following.

1. Include a mechanism in the approach to flag malicious hosts when detected so as to be avoided by successive agents. This would reduce latency.
2. A reverse mechanism for the point noted in 1 above when the threts in malicious host are resolved.
3. The use of sophisticated cryptographic algorithms and emerging security patches in the proposed approach.

7. ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their detailed and valuable comments.

8. REFERENCES

[1] Shzrivastava, S. & Nandi, G.C., (2014) Fragmentation based encryption approach for self protected mobile agent. *Journal of King Saud University - Computer and Information Sciences*, 26, pp.131- 142.

[2] Jansen W. A., (1998) Mobile Agents And Security. *National Institute of Standards and Technology*, USA.

[3] Razouki, H. & Hair, A., (2014). Towards A New Security Architecture of Mobile Agents. *International Journal of Soft Computing and Engineering (IJSCE)*, Vol 3 Issue 6, 55-60.

[4] Ahmed, T.M. (2009) 'Using Secure-Image Mechanism to Protect Mobile Agent against malicious Hosts ', *International Scholarly and Scientific Research & Innovation,* 3(11), pp. 364-369.

[5] Mahmoodi, M., Varnamkhasti, M. M. (2013) 'A Secure Communication in Mobile Agent System', *International Journal of Engineering Trends and Technology (IJETT),* 6(2), pp. 186-188.

[6] Dadhich, P., Dutta, K., Govil, M. C. (2010) 'Security Issues in Mobile Agents', *International Journal of Computer Applications,* 11(4), pp. 1-7.

[7] Ahmed, T.M. (2013) 'Protect Mobile Agent Against Malicious Host Using Partial-Mobility Mechanism', *International Journal in Foundations of Computer Science & Technology (IJFCST),* 3(6), pp. 41-52.

[8] D'Anna, L., Matt, B., Reisse, A., Vleck, T.V., Schwab, S. and LeBlanc, P (2003) *Self-Protecting Mobile Agents Obfuscation Report* , DARPA: Network Associates Laboratories.

[9] Gupta, S. (2013) 'A Secure Architecture for Mobile Agent Based Communication System ', *International Journal of Latest Trends in Engineering and Technology (IJLTET),* 2(2), pp. 160-164.

[10] Ismail, L. (2008) 'A Secure Mobile Agents Platform', *Journal of Communications,* 3(2), pp. 1-12.

[11] Madkour, M.A., Eassa, F.E., Ali, A.M., & Qayyum, N.U (2014) 'Securing Mobile-Agent-Based Systems Against Malicious Hosts',*World Applied Sciences Journal,* 29(2), pp. 287-297.

[12] Dey, S & Sinha, D., (2014) A Survey on Protection Techniques of Mobile Agents from Malicious Hosts. *International Journal of Innovative Research in Computer and Communication Engineering,* Vol 2, Issue 8.

[13] Mishra, P.K., Singh, R. (2014) 'A Survey on Reliability Estimation Techniques for Mobile Agent based Systems ', *International Journal of Advanced Computer Research,* 4(14), pp. 123-133.

[14] Pai, P., Shinde, S.K., Khachane, A.R. (2012) 'Security in Mobile Agent Communication', *International Journal of AdvancedEngineering Research and Studies,* 1(4), pp. 74-80.

[15] Singh, D., Thakur, A., Gupta, D. (2015) 'A Review of Mobile Agent Security', *International Journal of Advanced Research in Computer Science and Software Engineering,* 5(2), pp. 188-190.

[16] Lee, H., Alves-Foss, J., and Harrison, S. (2004)'The Use Of Encrypted Functions For Mobile Agent Security'. *Hawaii International Conference On System Sciences*. Hawaii: DARPA. 1-10.

[17] Shrivastava, R., Mehta, P. (2012) 'Securing Mobile Agent And Reducing Overhead Using Dummy And Monitoring Mobile Agents', *International Journal of Management, IT and Engineering,* 2(4), pp. 296-303.

[18] Ebietomere, E.P. & Ekuobase, G.O., (2014) Issues on Mobile Agent Technology Adoption. *African Journal of Computing & ICT*, 7.

[19] Hock Kim T. &. Moreau L, (2001), Mobile Code For Key Propagation, Paper, Notes in theoretical Computer Science 63, UK,.

[20] Anand T.& Neeran K. (2000) A Security Architecture for Mobile Agents in Ajanta, Proceedings of the International Conference on Distributed Computing Systems.