Proceedings of the Federated Conference on Computer Science and Information Systems pp. 707–714
DOI: 10.15439/2018F262
ISSN 2300-5963 ACSIS, Vol. 15

Applying Formal Methods to Specify Security Requirements in Multi–Agent Systems

Vinitha Hannah Subburaj
School of Engineering, Computer Science and Mathematics
WTAMU Box 60767
West Texas A&M University Canyon, TX 79016 USA
Email: vsubburaj@wtamu.edu

Joseph E. Urban
Arizona State University
Tempe, AZ 85281 USA
Email: urban@asu.edu

***Abstract*—Security has become an important concern with the development of large scale distributed and heterogeneous multi-agent systems (MAS). One of the main problems in addressing security during the development of MAS is that security is often an afterthought. The cost involved to patch existing systems against vulnerabilities and attacks after deployment is high. If developers and designers can spend some quality time investigating security aspects before beginning to code then this cost can be reduced significantly. Also, using formal methods to specify the complex behavior of large scale software systems has resulted in reliable software systems. This research effort was focused on using formal methods early in the development lifecycle to specify security requirements for MAS. New solutions are emerging to fix security related issues, but how much thought gets in during the early phases of development in terms of security needs to be answered. In this paper, analysis of security requirements for MAS, existing solutions to secure MAS, and the use of formal methods to specify security requirements has been studied. Descartes – Agent, a formal specification language for specifying agent systems has been taken into study to model the security requirements of MAS early on in the development process. Functional specifications of MAS are modelled along with the non-functional security requirements using the Descartes – Agent specification language. A case study example is used to illustrate the specification of security requirements in MAS using the Descartes – Agent.**

***Index Terms*—multi-agent systems, security requirements, formal methods, Descartes - Agent**

I. Introduction

MAS are a set of software agents that work together to solve problems that are beyond the individual capacity of a single software agent. MAS are a comparably new software paradigm, which has been accepted widely in several application sectors that involve large and complex tasks. The autonomous, pro-active and dynamic problem solving characteristics of MAS have recently caught the attention of several application areas, such as: banking, transportation, e-business, and healthcare. In all these mentioned services, it is imperative that security must be assured. These services will face serious deployment issues if the security requirements are not being enforced. This approach is possible by considering the agent properties and the security aspects that relate with those specific properties.

The use of MAS in open, distributed, and heterogeneous applications, however may cause problems with security issues which in turn may affect the success of the various applications. Security in MAS is an upcoming field in a well-established field of study, such as security in networks, P2P, and web services communication. Hence, this paper analyzes the basic security concepts required to be applied to security of MAS.

This paper includes a review of the past and present work related to the security issues of MAS. Also, the research effort has studied the existing security technologies used as solutions to address the security issues of MAS. Mobile agents, host security, agent communication, and delegation are some of the current security technologies that are used to address security issues [1].

The need for systematic and secure system development has increased the use of formal methods. The following are some of the specific characteristics of using formal methods to specify secure software systems [5]:

- enable reasoning from logical/mathematical specifications of the behaviors of computing devices
- offer accurate proofs, so that all system behaviors meet desirable properties
- crucial for security goals
- rule out a range of attacks
- provide guidance for gapless construction and
- always use models.

Implementing formal methods in various areas such as verification of hardware system, embedded systems, analysis and testing of software has improved the quality of computer systems. There is a forecast that formal methods can bring similar improvement in the security of software systems. Formal methods have been associated with security

applications for a while [15], thereby offering new techniques for security goals across a wider range of components. Without the implementation of formal methods, security will always remain weak. In this paper, one such formal method has been used to specify the security requirements of MAS.

Wing, in her paper, has stated that security always had played a vital role in the development of formal methods in the 70s and early 80s [7]. There are a few questions that might arise regarding the formal methods. Has the scenario changed? Are the formal methods now ready to have a significant role in the production of more secure systems? The answer is yes, formal methods now play an important role in security systems. In this paper, limitations of formal methods, summary of the results on how model checking and theorem proving tools were discussed. Also, the challenges and opportunities for formal methods in analyzing the security of systems, beyond the protocol level are also elaborated. Formal methods need integration with 1) other methods that address issues on formalization (analysis must include several factors such as risk, hazard, fault, and intrusion detection) and 2) into the entire software development lifecycle (such as during requirements analysis, testing, and simulation). Finally, there is a necessity to introduce the human factor (cannot be ignored), which in principle is part of the system's environment. Research conducted on modeling of human behavior, human-computer interaction, and management of processes and organizations can all aggregate the formal nature of research on formal methods.

The remainder of this paper is structured as follows:
Section 2 discusses the existing work related to security issues in MAS, security solutions to MAS, and the use of formal specification to specify security requirements in MAS. Section 3 discusses the earlier extensions done to the Descartes specification language to specify agent systems. Section 4 discusses the security framework developed in this research effort to specify the security requirement of MAS using Descartes – Agent. Section 5 provides a case study example that illustrates the application of the developed security framework with an e-commerce application. Section 6 discusses the lessons learnt and Section 7 summarizes the paper with a brief discussion of future work.

II. Related Work

Jung, et. al [2] surveyed existing research efforts that exist related to security in MAS, with a special focus on access control and trust/reputation. The paper concluded that security of agent based environments is critical. In spite of several efforts, many problems still remain and appear to be challenging with the continuous development of new technologies that are developed.

The research described in the research paper [3] identified the various security issues encountered by MAS. In order to assure MAS security, the paper examined the following: 1) basic concepts of security in computing, 2) characteristics of agents and MAS that introduce new threats, and 3) different strategies to prevent attacks. However, despite the similarities, security in MAS has specific requirements which need the autonomy, mobility, and other agent features that are not usually found in most conventional systems.

A model (based on the concepts and models regarding agent's role and communications) is presented [4] for securing MAS. The model provides an adequate way to ensure the security requirements and design are combined with system functionalities during the development process. The proposed model also incorporates the general security requirements at the agent and system levels. The paper has considered and addressed several system level threats, such as 1) corrupted mobile agents attack the main system host, 2) fake agent, 3) insecure communication among the platforms, and 4) agent level threats. The research work has attempted to extend the Gaia methodology with the security model. Further research work is needed in order to provide developers with security solutions for MAS based on the Gaia methodology.

A secure-critical system is difficult to develop and there are several known research issues regarding the security weaknesses in many sectors. Hence, a good methodology to support secure systems development is immediately needed. The research paper [6] presents the aim to assist the difficult task of developing security-critical systems using an approach of the Unified Modeling Language. The extension UMLsec of UML [6] (that allows expressing security relevant information within the diagrams) in a system specification is described in this paper. The UMLsec is defined in the form of a UML profile using the standard UML extension mechanisms. In particular, the related constraints provide criteria to classify the security aspects of a system design, by attributing to the formal semantics of a simplified fragment of UML. Formal evaluation is possible since the behavioral parts of UMLsec are considered with formal semantics. Hence, even the security experts who undertake a formal evaluation for certification purposes also may benefit from the possibility of using a specification language that may be more adaptable than some conventional formal methods.

Even though security has a major role in the development of MAS, security requirements are usually considered after the design of a system. The main reason is because of the fact that agent oriented software engineering methodologies have not unified security concerns throughout their developing stages. Mouratidis and Giorgini [12, 20] in their paper, introduce extensions to the Tropos methodology to enable them to model security concerns throughout the entire development process. This paper also describes the new concepts and modeling activities getting integrated to the current stages of Tropos. Tropos is characterized by the following three key aspects.

- deals with all the phases of a system development, adopting a uniform and homogeneous way,
- attends to the early requirements (emphasizing the need to understand organizational goals), and
- builds a model of the system that is refined and extended from a conceptual level to executable level, by a sequence of transformational steps.

The Tropos methodology includes five main software development stages, such as early and late requirements analysis, architectural design, detailed design, and implementation. In order to extend Tropos with security related concepts, factors such as security concepts and security modeling activities are detailed in the paper. A real life case study from the health and social care sector is used to illustrate the approach using Single Assessment Process (eSAP) system.

MAS have become a promising architectural approach for constructing Internet-based applications. Recent research work in software architecture have resulted in the necessity to truly define languages for designing and formalizing agent architectures and more specifically secure ones. This paper describes the basic fundamentals for an architectural description language (ADL) to specify secure MAS. Mouratidis, et al. [13] in their paper introduce a set of system design primitives that is conceptualized with the Z specification language to build secure MAS architectures. The main concepts of SKwyRL-ADL, including the security aspects, are described in this paper. The Z specification language is used to describe SKwyRLADL concepts. Z is widely used as a formal specification language as it is clear, concise and easy to learn. The three sub-models of SKwyRLADL: agent model, security model, and architectural model are detailed in this paper. The concept is applied on an e-commerce example to illustrate the research effort. The illustration involves the description of formally specified architectural aspects, such as interfaces, knowledge bases, security objectives, security mechanisms, and plans of the e-Media system.

III. Background

The Descartes specification language, developed by Urban [10] was designed to be used throughout the software life cycle. The relationship between the input and the output of a system is functionally specified when using this specification language. Descartes defines the input data and output data and then relates them in such a way that output data becomes a function of input data. The data structuring methods used with this language are known as Hoare trees. These Hoare trees use three structuring methods namely direct product, discriminated union, and sequence.

Direct product provides for the concatenation of sets of elements. Discriminated union provides for the selection of one element out of a set of elements. A plus sign (+) is used to denote discriminated union. Sequence represents zero or more repetitions of a set of elements. Sequence is indicated by an asterisk (*) suffixed to the node name.

By definition of Hoare trees, a sequence node is followed by a sub node. A single node can accommodate a sequence of direct product or a sequence of discriminated union. In the Descartes specification language, a literal is any string that is enclosed within single quotes. Consider the following example,

agent

'autonomous_agent' wherein autonomous_agent is a literal.

The Descartes specification language was extended in 2013 by Subburaj [11] for specifying complex agent systems. The extensions made to the Descartes specification language follows a top-down modular development allowing for the decomposition and incremental development of large agent systems. Six new concepts were added to Descartes for specifying and validating agent software systems. The added concepts were: (1) agent construct; (2) agent goal; (3) agent attributes; (4) agent roles; (5) agent plans; and (6) communication protocol.

Agent systems consist of multiple autonomous agents. Each of the agents has a specific goal to achieve and a set of actions to perform in order to achieve a goal. The agent construct in an agent system is used to define the behavior of an agent, including the goal, different roles, type of events, the plans, and the knowledge base. Each agent in an agent system has a structure. The notion of declaring an agent can be compared to the identification of objects in an object oriented methodology. The declaration of an agent module is pre-pended with a unary "agent" reserved word. Consider the following example,

agent AGENT_MODLUE_NAME_(INPUT)

Every agent has a goal of achieving a certain state or task. For example, imagine an agent that would start running with a goal of cleaning a house. The initial goal of such an agent is to clean the house and perform actions accordingly to achieve the goal statement. In Descartes - Agent, the agent goal is specified by using a new primitive, "goal", added to the Descartes syntax. An agent goal is an important attribute to be specified in an agent system. The plans that are executed by an agent solely depend upon the goal defined for a specific agent.

The agent roles are used to identify the key roles in an agent system. The notion and description of role models has been adopted from the Gaia methodology [8].

One of the most important aspects of agents is that they act autonomously to achieve their goals. This characteristic of agents to act autonomously in an environment is realized through the plans part in an agent system. The plans consist of a sequence of actions that an agent will take when a corresponding event occurs. The first part of the plan specified the list of events that trigger the execution of a specific plan by the agent. The second part context describes the contexts when the plan is applicable. The context part is

used to specify the current beliefs of the agent system. This part consists of a set of rules that can be specified with respect to specific agents. The context part also communicates with the knowledge/belief component in the agent framework to update and reads agent specific rules. The next extension is the reserved keyword "plans" used to specify the agent plans. The keyword triggered_events is used to list the triggered events. The keyword context is used to specify the agent specific rules and belief. The keyword method is used to specify the list of actions to be taken. In order to specify the context of the plan, new logical primitives were added to Descartes - Agent.

The knowledge/belief base in an agent system contains the knowledge that the agent has about itself and its environment. An agent's plan reads and modifies the knowledge/beliefs base. The knowledge/belief base consists of logical rules that are known initially before the agent starts to execute the plans. Also, based upon the execution of plans by the agents in the agent systems, the knowledge/belief base gets updated according to a current belief. In the Descartes - Agent processor, the knowledge/belief base was implemented as a separate component. The processor before executing the agent plans and also after executing the agent plans will access the knowledge/belief component to take appropriate decisions.

The last extension to Descartes - Agent for specifying agent systems is the communication protocol. Agents interact with other agents in the agent system and also with the environment to realize agent goals. The communication protocol in the extended Descartes is set up by the name tag (in upper case letters) of the calling agent module within parentheses followed by a period and the name of the relevant message within parentheses followed by the "^" symbol and then the name tag (in upper case letters) of the called agent module within parentheses.

IV. Specifying Security Requirements Using The Descartes – Agent

A. MAS properties

Wooldridge and Jennings [15] software agents come with the following properties:

Autonomy: An agent has its own goal and the ability to operate without any human intervention; more importantly, agent has control over its own state and can regulate its own functioning without outside assistance.

Sociability: An agent is capable of interacting with other agents and humans using an agent communication language. This approach allows an agent to seek and provide services.

Reactivity: An agent is capable of perceiving and acting on its close environment. The agent can respond to changes that occur in its surroundings.

Pro-activeness: Agents are not only capable of responding to the stimulus from their surroundings, but are also capable of exhibiting a goal-oriented behavior by taking initiatives.

In addition, there are some other characteristics, such as situadeness, mobility, rationality, veracity, and benevolence. Situadeness means agents are capable of sensing a special condition based on the inputs received from the environment.

Table I.
Agent properties and associated security concerns

Agent property	Description	Security concerns
Situatedness	If the agent gets to sense the input from its local host, then problems are less. But, instead if the information is coming from the Internet then there comes the problem of trust.	Trust, authentication, and integrity
Autonomy	Malicious agents can intrude without any request from humans or other agents.	Authorization
Social Ability	Enabling secure communications among agents and between humans and agents.	Confidentiality, integrity, availability, accountability, and non-repudiation
Mobility	By being able to self-migrate from one platform to other platforms, agents are prone to a number of security attacks.	Authentication, confidentiality, integrity, privacy, and faulty tolerance Damage, DoS, breach of privacy or theft, harassment, social engineering, event-triggered attack, compound attacks, masquerading, unauthorized access, copy-and-reply, and repudiation
Cooperation	Many agents cooperatively working together to access resources and internal status of other agents. This leads to security concerns.	Authentication and authorization

The term software agents covers a wide range of more specific agent types. Etzioni and Weld [16] and Franklin and

Graesser [17] provide a list of attributes that each agent must possess to a lesser or greater degree. The software agent attributes are as follows:

"Reactivity: the ability to selectively sense and act

Autonomy: goal-directedness, proactive and self-starting behavior

Collaborative behavior: can work in concert with other agents to achieve a common goal

Communication ability: the ability to communicate with persons and other agents with language more resembling humanlike "speech acts" than typical symbol-level program-to-program protocols

Inferential capability: can act on abstract task specification using prior knowledge of general goals and preferred methods to achieve flexibility;

Temporal continuity: persistence of identity and state over long periods of time

Personality: the capability of manifesting the attributes of a "believable" character such as emotion

Adaptivity: being able to learn and improve with experience

Mobility: being able to migrate in a self-directed way from one host platform to another."

B. Security requirements in MAS

The autonomous, pro-active, and dynamic nature of software agents thought proven to solve challenging problems, also comes with security concerns. Often, these security aspects get unnoticed until the deployment of the end-deliverables. Patching the security flaws after deployment has always resulted in high costs.

From the above properties, it is evident that software agents operate in an open environment and are free to interact with their surroundings to achieve their goal. This openness gives rise to a number of security and trust issues. Some of the commonly occurring security problems with agent based systems [2] are: confidentiality, integrity, availability, accountability, and non-repudiation.

Based on agent characteristics, there [2, 12] have been presented a list of security requirements of the MAS. Table I associates agent characteristics with their associated security problems.

C. Descartes – Agent Security Specifications

Among the list of security concerns listed above, in this paper we focus on two concerns namely: access control and confidentiality.

To provide access control there are two steps involved: first is to provide authentication to a group of agents enabling them to establish their true identity and then authorization that allows us to define the type of access privileges each agent obtains.

Every agent has a goal of achieving a certain state or task. In the secure agent framework specified by using the primitive, "goal" being prepended by a "!" symbol. With the specification of the secure agents, the goal is enclosed within a * symbol denoting the goal of secure agents.

Figures 1 and 2 illustrate the Descartes – Agent framework for specifying MAS and the Descartes – Agent secure framework for specifying MAS.

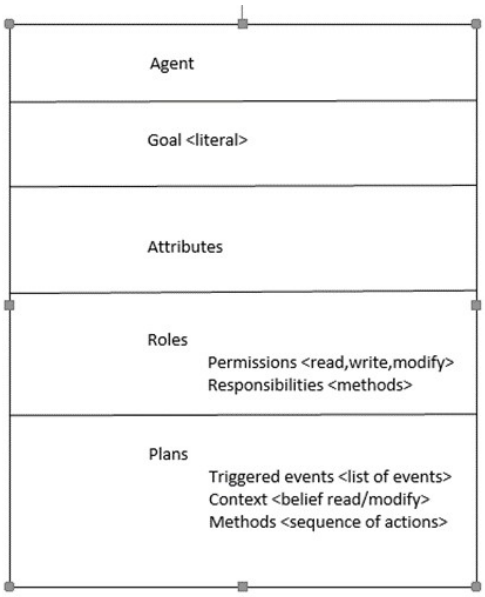

Fig 1. The Descartes – Agent framework for MAS

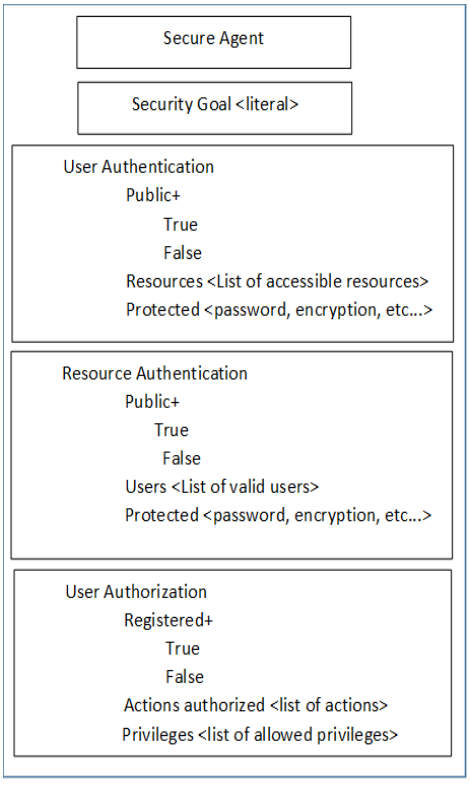

Fig 2. The Descartes – Agent security framework for MAS

1) Formally specifying authentication requirements for software agents

The authentication block in the security framework specified in the above Figure, is decomposed as user authentication and resource authentication. Within the user authentication, the node named public is a discriminated union meaning the public attribute can either be true or false. Resources in a secure agent system define the different types of resources that the secure agent can access. The protected attribute defines the method of authentication used by that user. There can be unique credentials such as passwords, encrypted passwords, and public-key-infrastructure schemes. The resource block is the same as the user except that there is a list of users that are given permission to access particular resources. Figure 3 illustrates the specification of the authentication requirement using Descartes – Agent.

```
agent TRANSACTION_AGENT
goal
   *! To successfully complete a secure transaction *
user1
   public+
        true
        false
      users
        files
             'list of files that can accessed'
          servers
             'list of servers that user1 can talk to'
      protected
          password*
             alphanumericstring

resource1
   public+
        true
        false
      resources
        users
             'list of users that can this resource'
      protected
         password*
            alphanumericstring
```

Fig 3. Authentication example using secure Descartes - Agent

2) Formally specifying authorization requirements for software agents

The user authorization block specified in the secure agent framework consists of three parts: registered, actions authorized, and privileges. The first part of the user authorization block specifies whether the user is a registered user or not. The second part of the authorization block allows one to specify all the actions that an authorized user can perform. The third part allows for the specification of all the privileges or access rights to specific resources. In order to specify the access privileges of users, new logical primitives were added to Descartes - Agent. Figure 5 illustrates the specification of an authorization requirement using Descartes – Agent. Figure 4, lists the newly added authorization primitives.

(<userx>)_HAS_READ_PERMISSIONS_TO (<resourcex>) – user X has read permission to access a specific resource

(<userx>)_HAS_WRITE_PERMISSIONS_TO (<resourcex>) – user X has write permission with a specific resource

(<userx>)_HAS_APPEND_PERMISSIONS_TO (<resourcex>) – user X has append permission with a specific resource

(<userx>)_HAS_EXECUTE_PERMISSIONS_TO (<resourcex>) – user X has execute permission with a specific

Fig 4. Authorization primitives

```
user2
      registered+
          true
          false
       actions_authorized
          update_database
            'update_the_database'
          receive_payment_info_from_agenty
            'receive_payment_info'
          process_payment
            'process_payment_info'
          confirm_transaction
            'confirm_the_completion_of_a_transaction'
        privileges
                (USER2)_HAS_WRITE_PERMISSIONS_TO
                (TRASACTION_RECORD)
                (USER2)_HAS_EXECUTE_PERMISSIONS_
                TO_(COMMIT_QUERY)
```

Fig 5. Authorization example using secure Descartes – Agent

3) Design and Implementation

The secure Descartes – Agent specifications discussed in Section IV can be transferred into UML design and then into implementation code. AGENT UML [22][23], an extension of Unified Modeling Language (UML) was proposed to facilitate developers with a smooth agent development process. The extended Descartes has already been specified using AUML in [21]. The extended Descartes – Agent security requirements can be specified using use case, sequence, and communication protocol diagrams. For instance, a sequence diagram in AUML is a diagram that describe sequence of messages between agents that exchange messages through protocols. These diagrams define the different agent roles, constraints, and the messages that are ordered according to a time axis. Sequence diagrams use the following basic components along with other components to describe a communication pattern between agents: agents and agent roles, agent lifelines and thread of interaction, connectors, messages, and conditions on messages. Authentication and

authorization requirements can be enforced via protocols every time a message transfer occurs between the agents. Since the Descartes specifications are in an executable form, with formal specification constructs close to that of programming, implementation of these Descartes – agent specifications can be done in any high-level programming language. With Descartes – Agent specifications, the transition from specification to design and then to implementation happens seamlessly.

V. Case Study Example

MAS are used to provide efficient e-commerce solutions, but different security related issues are associated with the agent solutions of e-commerce applications. A case study example of a real time MAS for e-commerce applications [19] is described for illustrating the security framework introduced in this research effort. The real time multi-agent architecture for an e-commerce application consists of four different types of agents namely: UserAgent, QuotingAgent, TrendWatchingAgent, and BuySellAgent.

The main goal of the USER_AGENT is to determine the user requirements such as the risk level, amount of money to spend, and the market sector preferences. The USER_AGENT specifies the quality threshold to ensure if the actual stock price lies within the threshold value [9]. Security requirements associated with this user agent requires authentication, authorization, and confidentiality. The following Descartes – Agent specification adds the security requirements discussed in Section IV.B to the USER_AGENT. Figure 6 illustrates the specification of USER_AGENT that includes authentication and confidentially security requirements.

VI. Lessons Learnt

Following were the lessons learnt out of this research effort. Security is a major issue when it comes to addressing requirements for MAS. The existing Descartes - Agent specification language constructs along with few newly added ones were successfully used to specify the security specifications of MAS. Case study examples similar to but not limited to the one described in the paper can be used to illustrate the extensions made to the Descartes - Agent specification language. The formal executable specification demonstrated the possibility of converting the security specification into design and then into implementation.

Some of the drawbacks identified out of this research effort are as follows: general framework for understanding the security requirements of MAS is not available; automated design and code generation techniques from the formal specification languages used to specify secure MAS is also scarce in the literature; efficient ways to rank and specify security requirements according to importance is not adequately discussed.

```
agent
    USER_AGENT_(RISK_LEVEL)_AND_(AM
    NCE)
goal
   *!to_determine_user_requirements_securily*
attributes
   RISK_LEVEL
          INTEGER
   AMT
          FLOATING_POINT
   PREFERENCE
          STRING
   stock_price
          'value_read_from_knowledge_base'
   quality_threshold
          'value_read_from_knowledge_base'
   get_price
          'message_sent'
user_authentication
   public+
          true
          false
       resources
          files
              'list_of_files_that_can_be_accessed_
          servers
               'list_of_servers_that_can_be_acces
          protected
                password*
                     alphanumericstring
user_confidentiality
    registered+
           true
           false
    actions_authorized
         check_stock_price
             'check_stock_price'
         check_risk_level
             'check_risk_level'
         decide_on_stock
              'make_decision_based_on_price_val
         confirm_transaction
             'confirm_the_completion_of_a_trans
     privileges
         (USER_AGENT)_HAS_READ_PERMI
         ROFILE)
         (USER_AGENT)_HAS_WRITE_PERM
         ECORD)
         (USER_AGENT)_HAS_EXECUTE_PE
         ESHOLD_QUERY)
```

Fig 6. Case study example using secure Descartes - Agent

VII. Summary and Future Work

A security framework that allows developers to formally specify the security requirements of MAS has been discussed in this paper. The security framework has been built as a part of the Descartes – Agent formal specification language. The key point on the developed security framework is that it can be applied early on in the development process of MAS. The identification of these security requirements early during the development of agent systems reduces the security patching cost involved with MAS development. One of the main benefits of using Descartes – Agent is that it allows partial specifications to be developed and executed. This feature allows one to specify security requirements with a high-level of abstraction.

Three important security issues with MAS, namely authentication, authorization, and confidentiality, were taken into study. The security framework built in this research effort allows for the specifications of security requirements that would implement these security solutions in MAS. The challenging aspect of incorporating a formal executable specification language to specify security requirements for MAS has been accomplished in this research effort. A case study example has also been discussed to illustrate the use of the security framework built in this research effort. The case study discussed in this paper serves as a basis for formally specifying security requirements for MAS and can be applied to different applications in similar fields.

As future work, the security framework developed will be extended to provide solutions to other security issues, such as trust, integrity, availability, accountability, and non-repudiation. Extending the security framework to enforce security with distributed MAS will be a future challenging effort.

References

[1] N. Borselius. "Security in multi-agent systems," Proceedings of the International Conference on Security and Management (SAM'02). 2002.

[2] Y. Jung, M. Kim, A. Masoumzadeh, and J. B. D. Joshi, "A survey of security issue in multi-agent systems," Artificial Intelligence Review, vol. 37, no. 3, pp. 239–260, Apr. 2011.

[3] R. C. Cavalcante, I. I. Bittencourt, A. P. D. Silva, M. Silva, E. Costa, and R. Santos, "A survey of security in multi-agent systems," Expert Systems with Applications, vol. 39, no. 5, pp. 4835–4846, 2012.

[4] Y. Hedin and E. Moradian, "Security in Multi-Agent Systems," Procedia Computer Science, vol. 60, pp. 1604–1612, 2015.

[5] S. Chong, J. Guttman, A. Datta, A. C. Myers, B. Pierce, P. Schaumont, T. Sherwood, N. Zeldovich, "Report on the NSF workshop on formal methods for security," CoRR, vol. abs/1608.00678, 2016.

[6] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," ≪UML≫ 2002 — The Unified Modeling Language Lecture Notes in Computer Science, pp. 412–425, 2002.

[7] J. Wing, "A symbiotic relationship between formal methods and security," Proceedings Computer Security, Dependability, and Assurance: From Needs to Solutions (Cat. No.98EX358).

[8] Cernuooi, L., et al., "The gaia methodology: basic concepts and extensions," in Multiagent systems, Artificial Societies and Simulated Organizations. 2004. 11(2). P. 69-88.

[9] V. H. Subburaj and J. E. Urban, "Formal Specification Language and Agent Applications," Studies in Big Data Intelligent Agents in Data-intensive Computing, pp. 99–122, 2015.

[10] Urban, J. E., "A Specification Language and its Processor," Computer Science Department. University of Southwestern Louisiana. 1977.

[11] V. H. Subburaj and J. E. Urban, "A formal specification language for modeling agent systems," 2013 Second International Conference on Informatics & Applications (ICIA), 2013.

[12] H. Mouratidis and P. Giorgini, "Secure Tropos: A Security-Oriented Extension Of The Tropos Methodology," International Journal of Software Engineering and Knowledge Engineering, vol. 17, no. 02, pp. 285–309, 2007.

[13] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. Neural Networks*, vol. 4, pp. 570–578, July 1993.

[14] Hussain, Shafiq, Peter Dunne, and Ghulam Rasool. "Formal Specification of Security Properties using Z Notation," Research Journal of Applied Sciences, Engineering and Technology 5.19 (2013): 4664-4670

[15] Wooldridge, M., Jennings, N.R.: Intelligent agents: Theories, Architectures and Languages, January 1995. Lecture Notes in Artificial Intelligence, vol. 890, ISBN 3-540-58855-8

[16] O. Etzioni and D. Weld, "Intelligent agents on the Internet: Fact, fiction, and forecast," IEEE Expert, vol. 10, no. 4, pp. 44–49, 1995.

[17] S. Franklin and A. Graesser, "Is It an agent, or just a program?: A taxonomy for autonomous agents," Intelligent Agents III Agent Theories, Architectures, and Languages Lecture Notes in Computer Science, pp. 21–35, 1997.W. D. Doyle, "Magnetization reversal in films with biaxial anisotropy," in *1987 Proc. INTERMAG Conf.*, pp. 2.2-1–2.2-6.

[18] N. Borselius, "Mobile agent security," Electronics & Communication Engineering Journal, vol. 14, no. 5, pp. 211–218, Jan. 2002

[19] L. C. Dipippo, V. Fay-Wolfe, L. Nair, E. Hodys, and O. Uvarov, "A Real-Time Multi-Agent System Architecture for E-Commerce Applications," Jan. 2000.

[20] H. Mouratidis, P. Giorgini, and G. Manson, "Modelling secure multiagent systems," Proceedings of the second international joint conference on Autonomous agents and multiagent systems - AAMAS 03, 2003.

[21] V. H. Subburaj, J. E. Urban, "Intelligent Agent Software Development Using AUML and the Descartes Specification Language," Proceedings of the 2nd IEEE International Workshop on Object / component/service-oriented Real-time Networked Ultra-dependable Systems (WORNUS 2011), pp. 297-305, March 28, 2011.

[22] B. Bauer, J. Muller, and J. Odell, "An extension of UML by protocols for multi-agent interaction," Proceedings Fourth International Conference on Multi Agent Systems, pp. 207-214, 2000.

[23] M. P. Huget and J. Odell, "Representing Agent Interaction Protocols with Agent UML," Agent-Oriented Software Engineering V Lecture Notes in Computer Science, pp. 16–30, 2005.