British Journal of Applied Science & Technology
19(5): 1-7, 2017; Article no.BJAST.31759
ISSN: 2231-0843, NLM ID: 101664541

SCIENCEDOMAIN *international*
www.sciencedomain.org

Modeling State Dependency in Agent Based Systems

Azween Abdullah[1*] and Ramachandran Ponnan[2]

[1]*School of Computing and IT, Taylors University, Subang Jaya, Selangor, Malaysia.*
[2]*School of Communications, Taylors University, Subang Jaya, Selangor, Malaysia.*

Authors' contributions

This work was carried out in collaboration between both authors. Author AA designed the study, performed the statistical analysis, wrote the protocol and wrote the first draft of the manuscript. Author RP managed the analyses of the study. Both authors managed the literature searches and read and approved the final manuscript.

Article Information

DOI: 10.9734/BJAST/2017/31759
Editor(s):
(1) Hui Li, School of Economics and Management, Zhejiang Normal University, China.
Reviewers:
(1) Alfonso González Briones, Salamanca University, Spain.
(2) Madhumathy, Visvesvaraya Technological University, India.
Complete Peer review History: http://www.sciencedomain.org/review-history/18315

Short Research Article

Received 24th January 2017
Accepted 19th March 2017
Published 23rd March 2017

ABSTRACT

The present paper reviews the problem of investigating incidents in systems with dependent states. Actions of an agent in such systems may lead to changes in the system, which could be made by an agent not directly, but indirectly through other agents. A method for modelling such systems is presented in this paper. The method allows defining a "manipulated system", in which other agents can be used to obtain the changes. A method for analysis of such systems by deriving them to the non-manipulated type is also presented.

Keywords: Informational no influence; mathematical models; security models; manipulation.

1. INTRODUCTION

Mathematical models of information non-influence today are probably the most commonly used in designing information processes and information systems, as well as in searching for new vulnerabilities. Joseph Amadee Goguen and Jose Meseguer were the first to express the

**Corresponding author: E-mail: Azween.Abdullah@taylors.edu.my;*

model of informational non-influence in paper [1] and to later develop it.

The most interesting result of their research was that they made it possible to separate the agents and the commands, which did not influence any other parts of the system. When some processes do not reflect on the other processes in any way, then it may be concluded that there are no information leaks in the system. Consequently, system security can be defined as the requirements of information non-influence of certain processes on the some other processes.

Then the research was extended to the area of technical processes, which provided the ability to define requirements to all operations in the system, so that observance of those requirements at every step also provided the ability to maintain system security [2]. Currently the model is widely used in practical applications in system design and in searching for vulnerabilities, as in [3,4,5,6,7].

This paper does not focus on modelling of system security and conditions to establish it. There is another interesting fact. The problem of finding the initiator of actions (changes in the system) may appear to be nontrivial, and it may have no solution in some cases within a group of information processes, where agents and actions have informational influence on one another. That is, a system's agent may remain "invisible", not because of the informational non-influence on the other processes, but due to the ability to influence them. The present paper describes the mathematical model of such processes, which allow an agent to remain indistinguishable in a system, where its actions influence the other elements of the system. The results obtained and presented by Goguen and Meseguer [1,8] and [2,9,10,11,12,13,14,15] have been used in this paper.

2. PROBLEM STATEMENT

The system shown in Fig. 1 has three agents and three objects. Each of the agents can rewrite only one object.

Agent S_1 can rewrite object O_1, but it cannot rewrite objects O_2 and O_3. Consequently, agent S_2 can rewrite only object O_2, and agent S_3 can rewrite only object O_3. While agent S_2 can also read O_1, and agent S_3 can read O_2. At first, it seems evident that when $value_{O_1}$ of object O_1 is modified, then the modification is the result of actions performed by agent S_1. However, there may be several nuances. What if actions of the agents are the result of information they receive? That is, agent S_2 writes data into O_2 based on the information received from O_1?

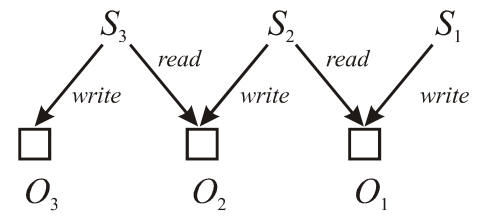

Fig. 1. The system with manipulation

In that case, agent S_1 can analyze the system dynamics and obtain a function.

$$value_{O_2} = f_{S_2}(value_{O_1}).$$

Thus, when agent S_1 wants to get the required $value_{O_3}$, it needs to find the suitable $value_{O_1}$, which is calculated as follows:

$$value_{O_1} = f_{S_2}^{-1}(f_{S_3}^{-1}(value_{O_3})).$$

It appears that agent S_1 can modify the states of all objects in the system, though its permissions do not include that. Such system will be referred to as "manipulated".

Consider another example. Assume that there are three network nodes: nod_1, nod_2, nod_3, as shown in Fig. 2.

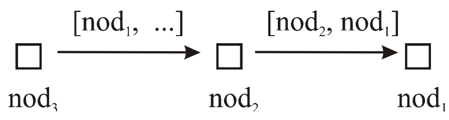

Fig. 2. Mediated information transfer channel

Nod_1 and nod_3 cannot send packets to each other and nod_2 is a proxy node. However, this does not mean that they cannot set up an information transmission channel. Assume that nod_3 sends a packet to nod_2 to set up a connection, but nod_3 indicates nod_1 as sender of the package instead of itself. Thus, nod_2 responds with a packet to nod_1. Nod_1 knows that it did not send any packets, and considers it as information sent by nod_3. Similar to the above case, it looks like packet [nod_2, nod_1] was sent by nod_3, although it did not do it. Below is the method for modelling the dynamics of such processes.

3. THE FORMAL MODEL OF THE SYSTEM

As the modern model of informational non-influence is aimed to check the formal requirements to processes and variables in memory, thus they had the notions of agents, which existed in the initial model, eliminated by Goguen and Meseguer, [1,8], but were later replaced with actions divided by information security domains. As the main objective of the paper is to define invisibility of actions made by specific agents, thus the set of agents needs to be returned to the model and additional formulations and rules need to be introduced.

The system's model will be defined by sets of agents **S**, a set of actions **A**, and a set of the system's states **V**.

To review manipulation mechanisms in our model, we will convert the notion of "influence" into a specific definition, which may not be all-embracing with respect to the definition of the classical model, but it suffices as shown in the examples above. Influence, according to that definition, is a property of states when one state in the system $v \in \mathbf{V}$ induces another state in the system $v' \in \mathbf{V}$. Thus, influence ($\rightsquigarrow$) is a relation established in a set of states:

$$\rightsquigarrow : \wp(\mathbf{V} \times \mathbf{V}).$$

When $v \rightsquigarrow v'$ occurs, it means that the system in state v will inevitably go to state v'. Such states will be referred to as *dependent states.*

The function shall also be defined:

$$step: V \times S \times A \to V.$$

The expression $step(v, s, a) = v'$ means that agent S while performing action A in state *v* transfers the system into **state** v'.

Thus, when $v \rightsquigarrow v'$, then it follows that there are such ***s*-and *a*-values,** for which $step(v, s, a) = v'$, and agent s always performs action a in that state.

That is, the influence relation ($\rightsquigarrow$) means that there is always an agent, which will perform the transition of the system in that state.

An auxiliary element of the action sequence is introduced. The action sequence will be denoted with Greek letters, for example, $\alpha = (a_1, \ldots, a_n)$. Function *run* is defined as a total of the transitions made by one agent. The function's output will be a new system state:

$$run: V \times S \times A^* \to V.$$

Function *run* can be defined recursively:

$$\mathbf{def}\ run(v, s, [\,]) = v,$$
$$\mathbf{def}\ run(v, s, a \circ \alpha) = run(step(v, s, a), s, \alpha).$$

Here [ ] denotes an empty sequence. Statement ∘ establishes a sequence of actions, that is, action *a* is done first, and set of actions α. As defined above, the system has states, which are initiating states for actions to be performed by other agents. Hence, having performed some actions, an agent can make a transition in the system, which it could not make on its own, although it was the initiator. Hence, function run', which considers the required actions of the other agents in the system.

$$run': V \times S \times A^* \to V.$$

To define function run', auxiliary function *fin* shall be introduced

$$fin: V \to V.$$

The function finds the system's final state in the influence graph, available from the current one. That is, if we consider relation ($\rightsquigarrow$) as a directed graph, then for each point $v \in \mathbf{V}$ in the graph we can find a way to point $v' \in \mathbf{V}$, which is most remote from it. In that case $fin(v) = v'$. When point *v* does not have outbound edges, then $fin(v) = v$. Implementation of function *fin* we can define $run^{'}$ recursively:

$$\mathbf{def}\ run'(v, s, [\,]) = v,$$
$$\mathbf{def}\ run'(v, s, a \circ \alpha) = run'(fin(step(v, s, a)), s, \alpha).$$

Functions that return the set of system states accessible for the agent:

$$rich: V \times S \to \wp(V).$$

The function returns the set of all states accessible for the agent in the current state v_0

$$rich(v_0, s) = \{v \mid \exists \alpha : v = run(v_0, s, \alpha)\}.$$

That formal model will be sufficient for us to define the further definitions of manipulability and system security.

4. A SYSTEM WITH MANIPULATION

The total of states, agents, actions and transition functions (*step*) is referred to as *the system (SYS).* The system (SYS) is considered as non-manipulatable when the below condition is fulfilled for all *v*'s and *s*'s.

SYS is non-manipulated $\Leftrightarrow$ $\forall v \in V, s \in Srich(v,s) \setminus rich'(v,s) = \emptyset$ is executed.

Otherwise, the system is manipulated. In a manipulated system, agents can perform actions, which will not be directly associated to them. A manipulated system will be denoted as M, and non-manipulated, as NM.

The set of system states may contain such states, in which the system can be shifted by manipulation, and those in which the system cannot be shifted by manipulation.

$$V_M \subseteq V, V_{NM} \subseteq V,$$
$$V_M \cup V_{NM} = V,$$
$$V_M \cap V_{NM} = \emptyset.$$

A set of manipulated states can be obtained as follows:

$$v \in V_M \Leftrightarrow \exists v_0, s: (v \in rich'(v_0,s) \text{and } (v \notin rich(v_0,s)).$$

and

$$v \in V_{NM} \Leftrightarrow \nexists v_0, s: (v \in rich'(v_0,s) \text{and } (v \notin rich(v_0,s)).$$

From the practical point of view, we are not interested in the system manipulation itself. Our concern is focused on the situations when an agent can succeed in causing some unwanted consequences and later escape responsibility for the incident, when it is investigated. A set of system states is divided into secure states(S) and non-secure states (NS):

$$V_S \subseteq V, V_{NS} \subseteq V,$$
$$V_S \cup V_{NS} = V,$$
$$V_S \cap V_{NS} = \emptyset.$$

With the knowledge of all the system's dynamics, we can define non-secure states, into which the system may transit from secure states:

$$V_{NS}^{alarm} \subseteq V_{NS},$$
$$v \in V_{NS}^{alarm} \Leftrightarrow v \in V_{NS} \text{ and } \exists v_0, s: \left(v_0 \in V_s \text{and} \left(v \in rich(v_0,s) \text{or } v \in rich'(v_0,s)\right)\right).$$

Here states in which the system can shift particularly by manipulation can be singled out:

$$V_{NS}^{alarm-M} \subseteq V_{NS}^{alarm},$$
$$v \in V_{NS}^{alarm-M} \Leftrightarrow v \in V_{NS} \text{ and } \exists v_0, s: \left(v_0 \in V_s \text{and } \left(v \notin rich(v_0,s) \text{and } v \in rich'(v_0,s)\right)\right)$$

Similarly, dangerous states can be found in set $\mathbf{V_S}$.

$$V_S^{alarm} \subseteq V_S, V_S^{alarm-M} \subseteq V_S^{alarm},$$
$$v \in V_S^{alarm} \Leftrightarrow v \in V_S \text{and } \exists v', s: \left(v' \in V_{NS} \text{and} \left(v' \in rich(v,s) \text{or } v' \in rich'(v,s)\right)\right).$$
$$v \in V_S^{alarm} \Leftrightarrow v \in V_S \text{and } \exists v', s: \left(v' \in V_{NS} \text{and} \left(v' \notin rich(v,s) \text{and } v' \in rich'(v,s)\right)\right).$$

States $\mathbf{V_S^{alarm-M}}$ from which the system can be shifted into non-secure states $\mathbf{V_{NS}^{alarm-M}}$ considering manipulation can be found by performing the above analysis. The highest hazard of those states, is that with the system dynamics it is not always possible to find out which agent "violated" the security requirements when the system is in one of the states $\mathbf{V_{NS}^{alarm-M}}$.

Below is a directed graph of system states, where states are the vertices and the edges are relations of some states influence the other states. An example of such graph is shown in Fig. 3.

The graph's special feature is that it contains a cycle. Thus, function *fin* cannot be computed. The system in states v_1, v_2, v_3, v_5 enters a cycle and it does not have the "freedom of choice" to shift into some other state. The systems with those conditions will be defined as *inert.* Agents in inert systems can be removed only by specifying its operating algorithm.

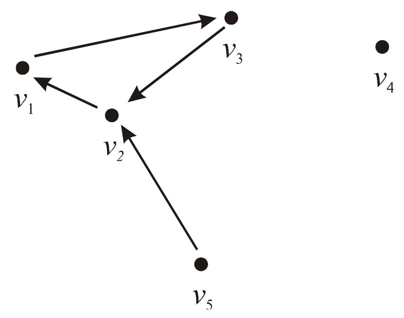

Fig. 3. Graph of system states

Thus, function *fin* imposes restrictions on the researched system. It can be found provided that the following conditions are fulfilled:

- The state graph does not include cycles;
- Every vertex in the graph has not more than one outbound edge.

When those conditions are not fulfilled, then the graph may be shifted into any required state by performing two operations:

- Vertices that have more than one outbound edge are duplicated by the number of outbound edges;
- The cycle in the state graph is replaced with one vertex.

5. SYSTEM SECURITY ANALYSIS

Analysis of the methods for system transition into non-secure states remains the most important issue. A system is *secure* when being in a secure state it cannot transit into a non-secure state. The requirement can be demonstrated with step transition.

SYS is secure $\Leftrightarrow \forall v \in \mathbf{V_S}$ and $\forall a \in \mathbf{A}, \forall s \in \mathbf{S}, step(v,a,s) = v', v' \in \mathbf{V_S}$.

This requirement cannot be analyzed at all times, because many states are initiators of transitions for the next ones. That is, dependent transitions also need to be analyzed in a manipulated system.

For simplification of the analysis of manipulated systems, the technique for derivation of transition functions (step) can be implemented, so that initial states are excluded in the influence graph.

Thus, when $step(v,a,s) = v'$ and $v' \rightsquigarrow v'$, then $\widetilde{step}(v,a,s) = v'$. That is, $\widetilde{step}(v,a,s) = fin(step(v,a,s))$. The system with an initial function *step* is denoted as SYS, as before. The system with derived function $\widetilde{step}$ is demoted as $\widetilde{SYS}$. Hence, we can arrive at the theorem below:

Theorem 1. $\widetilde{SYS}$ is secure $\Leftrightarrow$ SYS is secure.

For proving the theorem, we consider a case when *SYS* is secure, but at the same time $\widetilde{SYS}$ is not secure. Then there is such $step(v,a,s) = v'$ and $v' \rightsquigarrow v''$, with which $v' \in \mathbf{V_S}$ and $v'' \notin \mathbf{V_S}$. It appears that initial system *SYS* has some transition $step(v', a', s') = v''$ with which $v' \in \mathbf{V_S}$ and $v'' \notin \mathbf{V_S}$, and it contradicts the security definition.

Thus, we arrive at the following condition: $\widetilde{SYS}$ is secure $\Leftarrow$ SYS is secure.

Similar proof is made for this condition: $\widetilde{SYS}$ is secure $\Rightarrow$ SYS is secure.

Here we review the example of the derivation presented in the beginning of the paper. In that case when node nod_3 sends packet [nod_1, nod_2] then for the derived function $\widetilde{step}$, it sends packet [nod_2, nod_1].

Similarly, the example with three agents and objects that have read and write permission, description of the system is provided in Fig. 1. The system state will be defined by the states of each of the three objects. Each of the objects can be in state {1, 2, 3}. The action will express the change of state for a single object. Thus, agent s_1 can perform actions (…, …, 1), (…, …, 2) and (…, …, 3).

For example, $step((1,1,1),(\dots,\dots,3),s_1) = (1,1,3)$. Previously it was established that subject s_2 reads information from o_1 and writes information in o_2 based on it. Assume that system SYS the rule makes agent $s_2$write ino_2 a value that is by 1 greater than the value in o_1. That is, in the influence state graph will have edge (…, …, 1) $\rightarrow$ (…, 2, 1) added, due to performing the mandatory transition $step((\dots,\dots,1),(\dots,2,\dots),s_2) = (\dots,2,1)$. Similarly, edges shown below are added:

(…, …, 2) $\rightarrow$ (…, 3, 2),
(…, …, 3) $\rightarrow$ (…, 1, 3).

Also for agent s_3:

$(\ldots, 1, \ldots) \to (2, 1, \ldots)$,
$(\ldots, 2, \ldots) \to (3, 2, \ldots)$,
$(\ldots, 3, \ldots) \to (1, 3, \ldots)$,

As a result, we described a manipulated system SYS. For deriving it into an equivalent non-manipulated system $\widetilde{SYS}$ it is required to change transition functions by assigning values of *fin*(*v*) to them.

The following is obtained:

$$step((\ldots,\ldots,\ldots),(\ldots,\ldots,1),s_1) = (1,2,3),$$
$$step((\ldots,\ldots,\ldots),(\ldots,\ldots,2),s_1) = (1,3,2),$$
$$step((\ldots,\ldots,\ldots),(\ldots,\ldots,3),s_1) = (2,1,3).$$

The influence graph of system $\widetilde{SYS}$ does not have edges. It is a manipulated system, but all the accessible states in system SYS are also accessible in system $\widetilde{SYS}$. Thus we can find out that there are only three steady states and all of them are accessible by agent s_1. When, for example, some state is a violation of the security policy then there is no need to investigate the behavior of the other system agents.

6. CONCLUSIONS

The problem of manipulated systems reviewed in this paper is related not only to technical and informational systems but also to social systems. The complexity of analyzing similar systems consists in finding dependent system states. Dependent states for social systems are rather of probabilistic nature. That is, each single agent has freedom to choose their behavior, however a large number of people as a whole do not have such "freedom of choice". Consequently, major retailers learned quite well how to find dependent states with data mining methods and thus turn a social system into a manipulated one. It is not always easier to find dependent states in technical systems than in social systems. However, it is possible when we focus only on one "plane" of states. For example, IDS system is set up, and all data packages go through it. The statistics of selected packages is used for analyzing the dependence; that is, packet transmission patterns that indicate dependent states in the system are detected.

This paper reviewed a general method for modelling manipulated systems. A method for transforming a manipulated into a non-manipulated system was presented to simplify its analysis and finding real initiators of changes in the system. The present study can be a basis for creating incident investigation mechanisms in systems with non-isolated agents, which can have some influence on each other's behavior and for improvement of user behavior tracking.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Goguen J, Meseguer J. Security policies and security models. Security and Privacy, In Proceedings of the IEEE Security and Privacy Symposium on 11–11 10.1109/SP. Oakland, CA, USA. 1982;11-20.
2. Oheimb VD. Information flow control revisited: Noninfluence = noninterference + nonleakage. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2004;3193:225-243.
3. Verbeek F, et al. Formal API specification of the PikeOS separation kernel. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2015;9058:375-389.
4. Murray T, Klein G, Gammie P, Sewel T. seL4: From general purpose to a proof of information flow enforcement. In Proceedings of the IEEE Symposium on Security and Privacy. Oakland, CA, USA. 2013;415-429.
DOI: 10.1109/SP.2013.35
5. Murray T. On high-assurance information-flow-secure programming languages. Proceedings of the 10th ACM Workshop on Programming Languages and Analysis for Security, PLAS 2015. 4 July. 2015;43-48.
6. Betarte G, Campo J, Luna C. Formally verifying isolation and availability in an idealized model of virtualization. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2011;6664:231-245.
7. Betarte G, Campo J, Luna C. Cache-leakage resilient OS isolation in an idealized model of virtualization. Proceedings of the Computer Security

Foundations Workshop, IEEE 25th Computer Security Foundations Symposium, CSF 2012; Cambridge, MA, US. 2012;186-197.

8. Goguen J, Meseguer J. Unwinding and inference control. In Proceedings of the IEEE Symposium on Security and Privacy. Oakland, CA, USA. 1984;75-86.
9. Baran ME, El-Markabi IM. A multiagent-based dispatching scheme for distributed generators for voltage support on distribution feeders. IEEE Trans. Power Syst. 2007;22(1):52–59.
10. Wan H, Li KK, Wong KP. An adaptive multiagent approach to protection relay coordination with distributed generators in industrial power distribution system. IEEE Trans Ind. Appl. 2010;46(5):2118–2124.
11. Amini MH, Nabi B, Haghifam MR. Load management using multi-agent systems in smart distribution network. IEEE PES General Meeting, Vancouver, BC, Canada; 2013.
12. Pipattanasomporn M, Feroze H, Rahman S. Multi-agent systems in a distributed smart grid: Design and implementation. IEEE/PES Power Systems Conf. and Exposition, PSCE'09. 2009;1–8.
13. Hadi Amini M, Jaddivada R, Mishra S, et al. Distributed security constrained economic dispatch. IEEE Innovative Smart Grid Technologies-Asia (ISGT ASIA), Bangkok. 2015;1–6.
14. Kamyab F, Amini M, Sheykhaa S. Demand response program in smart grid using supply function bidding mechanism. IEEE Trans. Smart Grid. 2016;7(3):1277–1284.
15. Müller SC, Häger U, Rehtanz C. A multi-agent system for adaptive power flow control in electrical transmission systems. IEEE Trans. Ind. Inf. 2014;10(4):1551–3203.