*I.J. Education and Management Engineering* **2012, 6, 67-74**
Published Online July 2012 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2012.06.12

***Available online at http://www.mecs-press.net/ijeme***

# An Initiative Mechanism of Safe-guarding the Codes of Migrating Instance in Migrating Workflow Based on Danger Theory

[a] Li Baohui, [b] Han Fangxi, [c] Wang Xiaolin

[a, b, c] *School of Computer Science & Technology, Shandong University, Jinan, P. R .China*

**Abstract**

The mobility of Migrating Instance（MI） brings many risks to the migrating workflow system. Especially, the codes of MI face the risk of being maliciously manipulated by the hostile working places. This paper presents an initiative mechanism of safe-guarding the codes of MI based on danger theory: for MI consists of several modules in order to complete relative tasks, MI will detect whether the block MI determines to run changed before executing. If there are some changes, MI will then perceive whether the changes dangerous; if dangerous, appropriate measures will be taken to amend the damaged codes and then obfuscate them lest this module will be attacked easily again. This mechanism improves the efficiency of MI compared with other mechanism, and makes MI avoid temporary attacks, which are illustrated with the experimental results at the end of this paper.

**Index Terms:** temporary attacks; perceive changes ;perceive dangers; amend damaged codes



## 1. Introduction

Migrating workflow[1] is a technique based on mobile agent. And MI is the main body of implementing business process for its mobility; working place(WP) represents the workflow participant, which is a local area network composed of a sever for MI and some working machines, providing the runtime environment, runtime services and workflow services for MI. The mobility of MI brings flexibility for migrating workflow, but malicious WP can manipulate the codes of MI to change the way MI implementing tasks to obtain unfair interest when MI on it. For example, WP can alter the codes designed to compare prices, making MI believe that malicious WP provides lower price even if not the case, and as a result, affecting the final results obtained by MI.

To solve the problem, Xiong Yunping[2,3] et al. introduced a method for detecting the integrity of MI based on immune when MI migrates from one WP to another or have finished tasks on a WP. But it could not locate the specific location of an attack, and also have no effect on temporary attacks which are added after MI began to implement tasks and restored to the original state after attacked successfully. Most importantly, it needs extensive computation, and did not give a way what MI should do after detected attacks. Li Tao[4] made a study of artificial immune in network security, and Zhou Zhenyu[5] et al. applied the danger theory in instruction detection system by designing a instruction detecting model. Gao Lanning[6] et al. used danger theory in security assessment of WP. But all of them provided only some theoretical framework which have some distance from reality.

MI could finish different tasks by different parts of codes. This paper divides the whole codes of MI into different blocks based on their functions, denoted by $\{BL_1,\ldots,BL_n\}$. According to the running mode of MI, MI will implement one task block[7] composed of several blocks on a WP. So this paper presents an initiative mechanism: first, it finds that if there are some changes in the block that MI will implement later; if so, further determine whether such changes dangerous. If dangerous, MI will then take measures to amend the codes, and obfuscate[8] the amended codes to avoid being attacked again. This mechanism improves the efficiency of MI

Corresponding author:
E-mail address: [a] delibh@126.com

compared with the method introduced in [3], and makes MI avoid temporary attacks and being attacked again,which are illustrated with the experimental results at the end of this paper.

This paper is organized as follows. Section II gives some informations on danger theory.Section III presents the mechanism in detail. In section IV, experiments and discussions are presented, and in section V, conclusions and future works are given.

## 2. INTRODUCTION TO DANGER THEORY

With the in-depth study of immunology, there are many progresses in the theory. Certain types of dead cell contain in their cytoplasm copious amounts of a molecule called uric acid, in a crystalline form. Uric acid can be danger signals, releasing into a certain region around the dead cell, forming a hazardous zone[9]. Dendritic cells(DCs) in the hazardous zone will then be activated. DCs has three states: immature DCs(iDCs), semimature DCs(smDCs) and mature DCs(mDCs). iDCs have the ability of discovering non-selfs and accepting a variety of exogenous or endogenous signals. According to the discovered non-selfs and accepted signals, cell differentiation will happen: iDCs into smDCs or mDCs, based on certain rules[10].smDCs and mDCs can produce different co-stimulatory signals and secret different activators,which determine whether the immune response should be activated. After having eliminated risks successfully, the dangerous singnal disappeared, and the effector cells transformed into memory cells, waiting for eliminating the same risks when emerged again.

Danger theory gives good explanations about some questions that can not be explained by artificial immune system, such as,why some non-selfs could not activate the immune response.

## 3. THE INITIATIVE MECHANISM FOR GUARANTEEING THE SAFETY OF mI'S CODES

**Definition 1(Code security of MI).** The code security of MI is that the codes should not be modified for malicious purposes, and the expected codes' logic(such as control flow sequences)should not be maliciously altered, ensuring that the implementation of tasks relies on the designer's original intent.

According to danger theory and the implementing characteristics of MI in migrating workflow, this paper constructed a new MI architecture with the function of perceiving dangers based on the structure presented in [1], which was shown in Fig.1.

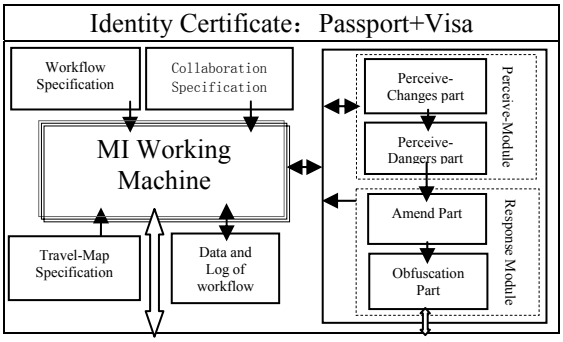


Figure 1. MI architecture with the function of perceiving dangers

The Perceive-Module contains two parts:one for perceiving changes, the other perceiving dangers. The former part is mainly used for detecting changes in a block which will be implemented by MI. The later is responsible for determing whether the changes dangerous, if so, the attacked codes will be submitted to the response-module for further processing.The reponse-module also includes two parts: one for amending the attacked codes, and the other for obfuscating codes after amended. The functions of the rest modules can be seen in [1].

The relationships between danger theoy and the mechanism are shown in table I. DCs is equivalent to the perceive-module, with functions of extracting changes' features and determining whether the changes is dangerous. The hazardous zone refers to the block that MI will implement later. Danger signal is the signal "*change*", issued when the perceive-changes part detects some changes. Exogenous or endogenous signal is the signal "*SDS*" which is used to help MI determine whether the changes dangerous, sent out by the part for perceiving dangers, helping MI determine whether the change was hazardous. Co-stimulatory signal is the decision signal, issued upon "*change*" and "*SDS*"; and the effector cells are the effectors to eliminate risks in response-module.

Supported by the National Natural Foundation of China(60573169) and Shandong Province Natural Science Foundation (ZR2009GM021)

TABLE I. THE RELATIONSHIPS BETWEEN DANGER THEOY AND THE MECHANISM

| Danger theory | This mechanism |
|---|---|
| DCs | The part for perceiving dangers |
| Dangerous Zone | the codes in a block which MI decides to implement next |
| Danger Signal | the signal "*change*" |
| Exogenous or Endogenous signals | the signal "*SDS*" |
| Costimulation signal | decision-making signal issued by the part of danger perception |
| Effector Cells | effectors in response-module |

### *3.1. The Perceive-Module*

The working process of perceive-module is as follows: first, the MI working machine extracts the code in the block which MI will implement later, denoted by "$Code_{bl\_id}$"; then, the perceive-changes part will ensure if there are some changes in "$Code_{bl\_id}$". If no changes, this mechanism will permit MI continuing implementing tasks; otherwise, the perceive-danger part will extract the features of changes and determines whether the changes are hazardous under the help of signal "*SDS*".

**Definition 2(Detecter).** A detector is a triple group(*strd*, τ, μ), stored in a set called D. *strd* is a binary string with the length of L produced by negative selection algorithm; τ represents the life value of a detector,and μ refers to the maturity of a detector, with the initial assignment of 0.

If a detector detects non-selfs, its maturity will be added by 1. When $\mu \geq \lambda$, a detector will be a memory detector,stored in the set of memory detectors, denoted by "DR". If the set reaches the maximum and a new memory detector added, the memory detector with the lowest life value will be eliminated from DR. If a detector in D does not detect changes in one test trip, its life value will minus 1. When the life value of a detector in D reaches 0, it will be deleted from D.

**Definition 3(Variation of Codes, va).** *va* is also a triple group(*bl_id*, *lcc*, *pos*). *bl_id* specifies the block $BL_{bl\_id}$ altered. *lcc* represents changes in codes' size in $BL_{bl\_id}$, and *pos* is the position where the variation starts from.

**Definition 4(Vaccine of Gene, vg).** vg is a dual group(*vg_str,* ρ). *vg_str* is the r-contiguous-bits string extracted from the memory detector when detecting non-selfs,and ρ is the usage of a vg,with the initial assignment of 0.

The role of vg is injecting some information of a memory detector into a new detector,reaching a lower missing rates. If a vg is utilized in constructing a new detector,its usage will plus 1.In order to prevent the detectors produced based on vg from converging on certain non-selfs,this paper carries a mutation of *vg_str*:switch the positions of two bits randomly selected from *vg_str*.

**Definition 5(Perceive-Module, PM).** PM is a group with six elements(S,D,DR,$f_r$, Lc,Set).S is a set of self-elements for MI's whole codes, based on self-elements-construction algorithm, denoted by$\{ s_1,s_2,\ldots,s_{NS}\}$, and each $s_i$ in S is a binary string with the length of L. D is a collection of detectors, denoted by$\{d_1,d_2,\ldots,d_{Nd}\}$, and Nd is the size of D. Each $d_i$ in D has the same length as self-elements' (This paer sets L a multiple of 8). DR is a collection of memory detectors, denoted by$\{dr_1,dr_2,\ldots,dr_{Ndr}\}$,and Ndr is size of DR. $f_r$ is a r-contiguous-bits-matching function,shown as follows:

$$f_r(X, Y) = \begin{cases} 1 & \exists i,j X[i\cdots j] = Y[i\cdots j] \\ 0 & otherwise \end{cases}$$

X and Y are two any binary strings, and $[i\ldots j]$ represents a r-consecutive bits from i to j in a binary string,that is to say, $j$-$i$+1=r. Lc is a collection of numbers, denoted by$\{lc_1, lc_2, lc_3, \ldots, lc_n\}$, and each $lc_i$ counted in bits refers to the original codes' length of $BL_{bl\_id}$. Va_set is set of *va*, denoted by $\{va_1, \ldots, va_{Nva}\}$, giving permission to some variations during MI implementing tasks among WPs, and Nva is the size of Set.

Self-elements-construction algorithm, detector generation algorithm and the cycles of detector can be seen in [2]. Algorithm 1 describes the working process of the perceive-changes part .

**Algorithm 1** Perceive if there are changes in a block of MI

**Input:** D; DR; S; the block identifier *bl_id*; the codes of $BL_{bl\_id}$ extracted by MI machine,denote by $Code_{bl\_id}$; $lc_{bl\_id}$.

**Output:** If no changes, *change*←*false*; otherwise, *change*←*true*, and output the starting elements's position of changes, denoted by $pos_{ele}$ and the amout of code' length changes, denoted by $lcc_{bl\_id}$

**Process:**

1. Initialization: PS←Φ ( null ), $DR_{mem}$←Φ; // PS: the set of elements to be detected; $DR_{mem}$:the set of memory detectors which have detected non-selfs.
2. *lc_now*←the length of $Code_{bl\_id}$ counted in bits;
3. PS←$Eod(Code_{bl\_id})$;// Eod(a): deviding a into elements to be detected based on construction algorithm for elements to be detected
4. if ($lc\_now \neq lc_{bl\_id}$) then

{/*if the length of BLbl_id has changed,it must have been altered*/

*change*←*true*; $pos_{ele}$←|PS|; $lcc_{bl\_id}$←$lc\_now$-$lc_{bl\_id}$; }

else

{for all ps∈PS do

sort DR by maturity from high to low;

{/*detect ps by DR */

for all dr∈DR do

if ($f_r$(dr,ps)=1) then

{*change*←*true;* $pos_{ele}$←*i;* $lcc_{bl\_id}$←*0*;

*vg_str*←Extract_r(dr);//Extract_r(dr): extract the r-contiguous-bits-matching substring of dr

VG←VG+{ *vg_str*, 0 };

dr.$\mu$←dr.$\mu$+1;

$DR_{mem}$ ←$DR_{mem}$ +{dr};

DR ←DR +{dr};

Go to 5;}

sort D by maturity from high to low;

for all d∈D do

{/*detect ps by D */

if ($f_r$(d,ps)=1) then

{*change*←*false*; $pos_{ele}$←*i ;* $lcc_{bl\_id}$←*0*; d.$\mu$←d.$\mu$+1;

if (d.$\mu$>λ) then

{ DR←DR+{d};

D←D+{d};}

else

{ d.$\tau$←d.$\tau$-1

if (d.$\tau$=*0*) then { D←D+{d};}}}

Update(D); // update D based on algorithm 3

}

5. Algorithm end

The difference between construction algorithm for elements to be detected and self-elements-construction algorithm is that the former does not delete the duplicated elements, others are the same.To maintain the size of D and the randomness of detectors,this paper has presented a algorithm to update D to improve the detection rate,as described in algorithm 2.

**Algorithm 2** Algorithm for updating D
**Input:** D; $DR_{mem}$; S.
**Output:** updated D
**Process:**

1. if（$DR_{mem} \neq \Phi$）then
   {for all $d \in DR_{mem}$ do
   {/*deal with each $d \in$ DRmem*/
   d_m←Mutate(d); //Mutate(d): mutate d through exchanging any two arbitrarily bits of d,
   Add (d_m, D;)}}
2. *Nd*_now←|D|;
   if (*Nd*_now >*Nd*) then //delete (ND-Nd_now) detectors from D
   {remove (*Nd*-*Nd*_now) detectors from D with the lower life value; }
   else if (*Nd*_now <*Nd*) then //add (Nd- Nd_now) new detectors to D
   {for i=(*Nd*-*Nd*_now) to 0 do
   {New(d);//generate a new detector based on detector generation algorithm
   D←D+{d};}}
3. Algorithm end.

If *change=true*, then the perceive-danges part will extract the features of the code's variation: *lcc*, *pos*, and *bl_id*, which can be obtained through algorithm 3.

**Algorithm 3** To obtain the variation of codes: *va_now*
**Input:** the number of continuous operation of algorithm 1,denoted by n;PS;D:the identifier of a block, denoted by *bl_id*; DR; S;the original block's codes in migrating workflow management machine,denoted by $Code\_MI_{bl\_id}$.
**Output:** *va_now*.
**Process:**

1. *pos_now*←|PS|;
   for *i*=1 to *n*
   {/*For D has a limited coverage to non-selfs, the probability of obtaining the right starting element of a variation is small. So it is necessary to fully update D to get the correct position */
   Call algorithm 1;
   if （$pos_{ele} < pos\_now$）then { $pos\_now = pos_{ele}$;}
   *i*←*i*+1;}
2. {$ps_{pos\_now}$←get(*pos_now*, PS);//get(n,A):get the n-th element of A
   $PS_{MI}$←*Eod(*$Code\_MI_{bl\_id}$*);*
   $ps_{MI}$←get ($pos_{pos\_now}$, $PS_{MI}$);
   *m*←Compare($ps_{MI}$, $ps_{pos\_now}$);}//compare psMI and pspos_now to get the final different bits' position
3. {/*obtain vanow*/
   *va_now.pos*←(*L**( *pos_now*-1)+*m*)/8;

*va_now.bl_id←bl_id;*

*va_now.lcc←lcc*$_{bl\_id}$;}

4. Algorithm end.

After having exactly obtained $va_{now}$, this mechanism will then produce signal "*SDS*" to help MI determine whether the changes are hazardous. And the process is described in algorithm 4.

**Algorithm 4** to produce the signal "*SDS"*
**Input:** *va_now*; Set.
**Output:** if hazardous, *SDS←true*;otherwise, *SDS←false*.
**Process:**
1. for all va∈Set do

{if (*va_now* =va) then {*SDS←true*;}}

2. *SDS←false*;

3. Algorithm end.

Till now,if MI accepted *change=false* or the two signals:*change=true* and *SDS=false* simultaneously, MI does not perceive risks and goes on implementing tasks; otherwise, MI perceived dangers and the response-module will be activated.

### *3.2. The Response-Module*

The response-module is responsible for amending and obfuscating the attacked codes, the process of which is described in algorithm 5.

**Definition 6(Effector, Eff).** Eff is a five-tuple(*bl_id*，*pos*，*lcc*，*str*，*rate*). *bl_id* represents the block in which a effector could role. *pos* is the starting position of an attack relative to the effector. *lcc* is the changes in codes' length caused by an attack. *str* is a string which can amend the attacked codes successfully, and *rate* is usage of a effector with the initial assignment 1, that is to say, Eff.r*ate* will plus 1 once Eff has been excluded a risk successfully.

**Algorithm 5** Amend the attacked codes
**Input:** the identifier of a block which has been attacked, denoted by *bl_id*; *va_now*; $Code\_MI_{bl\_id}$; $Code_{bl\_id}$.
**Output:** the correct and obfuscated $Code_{bl\_id}$.
**Process:**
1. sort Eb by *Eff.rate* from high to low; //Eb is a collections of Eff
2. for all Eff∈Eb do

{if (Matching(*va_now*, Eff)) then //if va_now, and mb are the same in { bl_id,lcc,pos}, then return true; otherwise, false.

{amend $Code_{bl\_id}$ by Eff.str;

Go to 4;}}

3. {Send(*char*, MI); // Send(A,B): send A to B; char: the original codes in Code_MIbl_id from (va_now.pos/8) to the end of this block

end $Code_{bl\_id}$ by *char*;

Eb←Eb+{( *va_now.bl_id*, *va_now. pos*, *va_now. lcc*, *char*, 1)};}

4. Algorithm end.

## 4. Experimental analysis and comparison

Experiments were carried out in the system "Commerce platform of collaborative based on mobile computing paradigm", developed by our group. MI in experiments will finish four tasks: ask-price, compare-price, order and delivery goods[1]; the codes of MI are dived into four blocks based their different functions.

**Experiment 1** The comparison between this mechanism and the method introduced in [3] under no dangers' circumstance.

In order to avoid the impact on the experimental result caused both by the time-consuming spent on the migrating way and different host performance s, this paper put all ten shops on one host. This paper recorded the

total consuming time after MI have visited each shop, when MI was implementing ask-price task.The parameter settings of this mechanism was shown in table II, and the experimental results in Fig.2.

TABLE II. LIST OF PARAMETERS

| L （bit） | r （bit） | τ/μ | Size of vg | Size of DR/n |
|---|---|---|---|---|
| 16 | 4 | 4/4 | 10 | 30/10 |

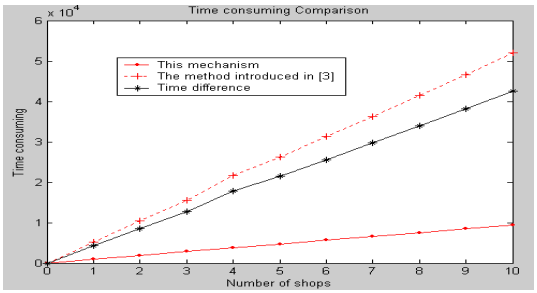


Figure 2. The time-consuming comparison between the two methods

As Fig.2 illustrated, the time difference between the two methods is a linear increase trend with the number of the shops increasing. And we also can see that there is a significant change in the slope of the time-difference line. This is because the detectors need to be updated after MI have visited four shops due to the parameter settings in the experiment. However, the more self-elements the detectors relative to, the more time will be spent, when MI updates detectors, under the same parameters.

**Experiment 2** The comparison of detection for temporary attacks between this mechanism and the method introduced in [3].

This paper designed an attack on a host decribed as follows: maliciously manipulate the codes for implementing compare-prices task to make MI take the price provided by the malicious WP as best price, regardless of the relationship between the current *best price* and the *price* provided by the malicious host; and the malicious host will restore to the original state of the codes after having attacked successfully.This experiment sets the *best-price* before arriving the malicious host was 12$,and the malicious WP will provide 21$.The parameters' settings are the same as them shown in table 2. This paper makes an analysis after MI have visited the malicious host, and the results are as follows: MI with this mechanism took 12$ as the best price, but MI with the method introduced in[3] took 21$ as the best price.

Till now,this paper can draw that this mechanism has the following advantages:

①Improve the efficiency of MI for this mechanism only deals with the needed codes,not the whole;
②This mechanism will not detect a block until MI decides to implement it, and MI implements them immediately after have amended it, so it limit the malicious WP in time. In this sense, it can detect temporary attacks;
③This mechanism provides a method of what MI should do after percepted dangers,rather than re-sending MI to a safety WP to continue to implement tasks.

## 5. conlutions and future works

To safe-guard MI's codes has great significance to the migrating workflow system.This paper presents an initiative mechanism based on danger theory,which has the some advantages illustrated by experiments.Future works will focus on designing an effective code-obfuscation method to make the codes of MI more secure.

## References

[1] Zeng Guangzhou,Dang Yan.The Study of Migrating Workflow Based on the Mobile Computing Paradigm.Chinese Journal of Computers,2003,26(10):13l3-13l9 (In Chinese).

[2] Guo Jingang,Zeng Guangzhou.Integrity Detection for Migrating Instance Based on Biology Immue.Computer Engineering,2008,34(12): 184-186 (In Chinese).
[3] Xiong Yunping,Zeng Guangzhou.Application of Artificial Immunology in Migrating Instance Integrity Detection.Journal of Computer Applications,2006,26(7):1514-1516 (In Chinese).
[4] Li Tao.Detection of Network Security Risk Based on Immune.Science in China(E Series-Information Science), 2005, 35(8) : 798−816 (In Chinese).
[5] Zhou Zhenyu, Zhang Qi, Shen Jianjing.The Danger Theory Inspired Approach to Network Security Assessment.Computer Application and Software, 2006, 23[11] : 111-113 (In Chinese).
[6] Gao Lanning, Han Fangxi.Security Assessment for Work Position Based on Danger Theory.Postgraduate Symposium in Beijing Area in 2007, 2007, 209-213 (In Chinese).
[7] Li Luyan , Zeng Guangzhou. Study of Itinerary Graph Generation Algorithm Based on Task blocks.Computer Engineering and Application, 2008, 44(32), 41-44 (In Chinese).
[8] Li Yongxiang.Rearsearch on Code Transformation.Dessertation of University of Science and Technology, 2002 (In Chinese).
[9] Bali Pulendran. Immune Activation:Death,Danger Dispatch and Dendritic Cells. Current Biology,Vol.14,R30-R32,January 6,2004.
[10] Greensmith J, Aicklin U, Cayzer S. Introducing Dendritic Cells as a Novel Immune-inspired Algorithm for Anomaly Detection.In:JacomC,PilatM,BentleyP, TimmisJ, eds. Artificial Immune Systems. Vol 3627 of Lecture Notes in Computer Science. Berlin,Heidelberg:Springer-Verlag,2005:153–167.