Computing and Informatics, Vol. 24, 2005, 351–370

ETHICALLY-SOCIAL APPROACH TO COMPUTER SECURITY PROBLEM

Krzysztof Cetnarowicz

Institute of Computer Science
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
e-mail: cetnar@agh.edu.pl

Gabriel Rojek

Department of Computer Science in Industry
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
e-mail: rojek@agh.edu.pl

Manuscript received 2 September 2004; revised 22 September 2005
Communicated by Patrick Brézillon

Abstract. This article came to existence on the ground of interest in what makes possible inter-human contacts and co-operation without excessive risk for any person. An individual is protected in society from dishonesty by ethical system as well as social mechanisms, which are not infallible, although so effective in working that transferring them onto the ground of computer systems is worth testing. Equipping computer resources with mechanisms observed in individuals within human societies will enable these resources to control the safety. Our goal is to obtain mechanisms which will enable security without definition what is desirable or undesirable in a computer system. This should enable to detect viruses or attack techniques that are not already known. Mechanisms are presented which may enable to recognize and dismiss resources undesirable or harmful in the computer system on the basis of behavior observation. The proposed mechanisms were tested and tests results are presented and discussed.

Keywords: Computer security, behavioral detection, ethically–social mechanisms, multi–agent systems, open systems

1 INTRODUCTION

From the moment of origin of the first computer virus or of the first illegal use of computer resources continuous works on successive protection has been going on. Despite rising expenditures on traditional safety concept of computer systems, the number of incidents doubles each year. Attacks are more and more sophisticated and those protecting themselves from them stay behind the possibilities of illegal use of computer systems. Traditional protection techniques may become insufficient in quite a short time [1].

Apart from the noticed trends in attacks on computer systems [1], there are noticeable trends of integration in computer systems themselves. The process of integration leads to creating one extensive computer system in the place of many smaller and easier to manage ones. The computer system that is the consequence of integration is, in the scope of correct ensuring of operation and development, difficult to command by man. The solution is to provide and develop tools that are a part of the system mentioned or of its resources. These tools enable automatic solving or assistance in the tasks given to them, which leads to solution of the above problems. The use of multi-agent systems is an example, in which man delegates his tasks to an agent autonomous in its work. Universal use of agent technique also enforces reviewing and presumable supplementing of the systems ensuring computer safety.

Our goal is to obtain a security system which will enable to detect undesirable resources in a computer system that is open. The security system should self-adjust to the computer system and to undesirable activities (viruses, attack techniques, etc.). The detection of undesirable resources should not be based on any *a priori* knowledge because of the changing activity of legitimate users and intruders in a computer system. One of known solutions how to detect *a priori* unknown intruders is computer immune system. An immunological approach to change detection is the basis of our analysis how to obtain a protection which will detect an *a priori* unknown undesirable activity in open computer system.

This paper is organized as follows: Section 2 presents the main issues of an immunological approach to computer security. Section 3 discusses open computer systems and effectiveness of immunological *self*/*non-self* discrimination. Section 4 presents ethically-social mechanisms that are the point of our interest, and also a conception of using this mechanism in the research on computer security. This conception is presented at general level in Section 4 and at lower level in Sections 5 and 6. M-agent architecture of multi-agent systems is used which is introduced among others in [2, 3]. In Section 7, a part of interesting us ethically-social mechanisms that have been implemented is given. Section 7 describes the social profile

at low (implementation) level. Section 8 presents the experiments done on the implemented multi-agent system with and without functioning of the social profile. In Section 9, conclusions of the discussion and experiments described in this paper are given.

2 RELATED WORK – AN IMMUNOLOGICAL APPROACH TO CHANGE DETECTION

The problem of computer system protection is viewed as an instance of the more general problem of distinguishing *self* (legitimate users, data, etc.) from *non-self* (unauthorized users, viruses, etc.). The algorithm of protection with immunological approach has two phases (as presented in [6]):

1. Generate a set of detectors. Each detector is a string that does not match any part of the protected data.
2. Monitor the protected data by comparing them with the detectors. If a detector is ever activated, a change is known to have occurred (e.g. a change caused by a virus or by an unauthorized user).

In the immunological approach, the detection problem is reduced to the problem of detecting whether or not a string has been changed, where a change could be a modification to an existing string or a new string being added to *self*. The string could be a string of bits, a string of assembler instructions, a string of data, etc. *Self* is defined as the string to be protected and *other* is defined as any other string.

2.1 Generation of Detector Set

To generate valid detectors, first the *self* string is split into segments of equal size. This produces the collection S of *self* (sub)strings to be protected (S contains all the substrings). The second step is the generation of collection R_0 which contains random strings of equal length as the strings in the collection S. Strings from R_0 that match *self* are eliminated. Strings that do not match any of the strings in S become members of the detector collection R. This procedure is called *censoring*.

2.2 Monitoring Protected Strings for Changes

Once a collection R of detector strings has been produced, the state of *self* can be monitored by continually matching strings in S against strings in R. This is achieved by choosing one string from S and one string from R and testing the matching of these two strings. If ever a match is found, then it is concluded that S has been changed – a *non-self* string is detected.

2.3 Matching

A perfect *match* between two strings of equal length means that at each location in the string, the symbols are identical. The matching requirement can be relaxed by using a matching rule that looks for r contiguous matches between symbols in corresponding positions. Two strings x and y match if x and y agree (match) at at least r contiguous locations. The matching rule can be applied to strings defined over any alphabet of symbols.

3 OPEN SYSTEMS AND AGENT TECHNOLOGY

Present computer systems become more and more complex and related to each other. Large nets used in transport, banking or insurance, as well as almost all devices connected to the Internet, are open systems. To have full knowledge about topology or current state of such system is impossible even for the owner or administrator. Central control of open system does not exist or is ineffective. There is no complete theory of open systems [8].

Rapidly developing agent technology allows full flow of resources among open computer systems. Autonomous agents can yet freely migrate in the net without knowledge of the owner or administrator. Agents can also execute their tasks without anybody's knowledge. These tasks can be both useful for the owner and destructive for the system. Two kinds of agents can be distinguished; although coming outside of the system on which they operate, they can fulfil two extremely different functions:

- intruder migrating and searching for a system, which it can attack
- agent sent by a friendly system to improve protection.

The two quoted examples do not cover the whole spectrum of possibilities, if one tries to relate agent functions to its origin. An agent created in the native environment and with the help of means available in this environment can be used to attack, destruct or blockade the computer system.

The new ways of designing of systems which, despite openness, would be able to survive the attacks, seem necessary. One should search for methods, which will allow, even for a damaged system, to continue its principal task, even at the cost of sacrifying some of its components [8]. Such systems should also allow to distinguish undesirable intruders, even if earlier they have been classified as *self*. The access to resources, also *non-self* ones, which are desirable and useful, should be enabled. It seems that all resources, more precisely, exact working of all resources, should be constantly monitored and on this basis the resources which are damaging or undesirable should be eliminated.

3.1 Immunological Approach to Safety of Open Computer Systems

The previously quoted example of two agents, intruder and friendly agent which migrate in net and want to get to system without knowledge of administrator, is possible in open computer systems. Security system with immunological approach [6, 7, 9, 11] permits the following distinction of resources:

- *self*, treated as desirable,
- *non-self*, treated as undesirable intruders.

Immunological mechanisms do not appear useful to investigate the above case of two migrating agents. Both resources mentioned would be classified by immunological system as *non-self*, which means (and it is treated accordingly) undesirable supplies. Treatment of a friendly but *non-self* agent as undesirable is not proper. However, it would be proper to divide computer resources as follows:

- *good*, desirable
- *bad*, undesirable.

The above notions of *good* and *bad* do not have absolute meaning. *Good* resource is a desirable resource for a definite computer system, in which evaluation takes place. *Bad* resource is an undesirable resource for a given system, although it can happen that in a different computer system it would be greatly desirable. Obviously the situation that a *good* resource in a certain system would be undesirable and valued as bad in a different system is also possible.

In open systems, resources recognized through immunological system as *non-self* can actually be desirable as well as useful. Resources defined as *self* do not necessarily have to be useful and *good* for the system, although resources recognized as *non-self* do not have to be harmful, i.e. *bad* (Figure 1). The division *good*/*bad* is not equivalent to *self*/*non-self*.

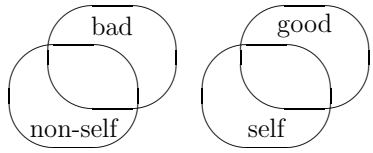


Fig. 1. *Non-self* is not equal to *bad*, *self* is not equal to *good*

In open systems, the more adequate discrimination turns out to be the distinction between *bad* – harmful and *good* – useful resources; however, the origin of supply (*self*/*non-self*) is not essential here. Distinguishing between *good* and *bad* can be accomplished only on the basis of observation of activity or intentions of the acting resource. Recognizing the resource structure, which allows the division into *self* and *non-self*, will not cause distinguishing between *good* and *bad* resources.

4 ETHICALLY-SOCIAL APPROACH TO BEHAVIOR ESTIMATION

In this section, M-agent architecture of multi-agent systems is used. M-agent architecture is introduced among others in [2, 3]. Activity of an autonomous agent in M-agent architecture is formed through profiles. Every profile is a different point of view according to which the agent takes its actions. The intellectual profile means the group of goals, the decisions made and actions realized in the sphere of basic jobs of an agent, for whose realization the agent was created. Agents can have other profiles too, e.g. the energetic profile, which serves to determine the existence of a given agent in the environment depending on its "energy", which the agent can draw in dependence on the actions it takes.

4.1 Social Phenomena Related to Security

On the one hand, the behavior of an individual within society is specified by ethical rules, and on the other by the rules of social living together [12, 13]. A human being taking decision about his/her action is influenced first of all by his/her ethical rules. The effects of this human action are, however, observed through other individuals in the society, who suitably, according to their social norms, react on such behavior. The human being obviously watches those reactions and, if the reactions indicate his/her *bad* behavior, s/he changes his/her own ethical norms. Society has an impact on the ethical system of such individual. Various factors of the environment can also have an impact; nobody would, for instance, accept ethical norms that would make him or her impossible to survive ("do not eat").

The above model of ethically-social mechanisms functioning in a society allows to develop ethical rules. The model also can be transferred to computer environment, and such ethical and social rules can be useful e.g. to control the access to computer resources by competing processes (as presented in [5]).

4.2 Concept of Using Social Phenomena in Secure Computer Resources Research

The main idea of ethically-social estimation of an autonomous agent behavior is the addition of two profiles: the ethical and the social ones. The ethical profile would cover operations that aim at:

- construction of "notion" about itself on the basis of one's ethical rules,
- fulfilment of one's ethical rules,
- updating and learning of ethical rules on the basis of environment observation, particularly of those environment elements which are agents.

The group of goals, decisions, operations and knowledge relating to social life mechanisms will be named the social profile. These will be the operations that aim at:

- fulfilment of living together rules in society,
- observation of environment, especially agents,
- "punishing" of the agents that do not fulfil the rules of living together.

The social and ethical profiles are inseparable and mutually supplementary. Social profiles of agents from the environment exert an indirect influence onto ethical profile of the agent concerned, as shown in the last subsection. Only thanks equipping every computer with resources within these two profiles it is available to keep social and moral order in open systems, and thus to create mechanisms similar in safety to those functioning well in known societies.

Ethical and social profiles can make up a supplement of every autonomous agent in M-agent architecture. In one of the most basic cases an agent $a = (a_i, a_e, a_s)$ would consist of three profiles (Figure 2) defined as follows:

- intellectual profile agent $a_i = (M, Q, S, I, X, L, m, q, s)$, where: m is the model of environment, m' is the model of foreseen environment after realization of strategy s, S is the configuration of possible strategies, M is the set of models of environments, q is the goal of agent a, Q is the configuration of possible goals of agent a, I is the operator of environment observation, X is the operator of strategy s realization, L is the operator of agent's learning, $V = (E, A, C)$ is the environment;
- ethical profile of agent $a_e = (M_e, Q_e, L_e)$,where: M_e is the set of ethical states m_e of agent a, Q_e is the set of goals and "anti-goals" (ethical rules) q_e of agent's ethical profile, L_e is the operator of agent's ethical learning $L_e : Q_e \times V \to Q_e$.
- social profile of agent $a_s = (M_s, Q_s, L_s)$, where: M_s is the set of social states m_s of agent a, Q_s is the configuration of goals q_s of agent's a social profile, L_s is the operator of social observation $L_s : M_s \times V \to M_s$.

4.3 Trust and Belief Versus Ethically-Social Behavior Estimation in Multi-Agent Systems

In [10] the author's view on the topic of security in open computer systems including open multi-agent systems according to the pervasive computing paradigm is presented. The opinion that existing security solutions are inadequate for open environments is given, which is similar to our view of security in open computer systems. The authors consider the use of agent methodologies, trust management, exchange of beliefs, delegation of permissions, obligations and credibility in relation to security of pervasive computing environments.

The *Vigil* security infrastructure [10] provides authentication of users and also allows users to delegate their rights and beliefs to other users. Classical trust management is extended by incorporating conditional delegation which allows conditions to be placed on execution of access right, re-delegation and negotiable delegation. An agent may receive delegation, but, in turn, has to fulfil certain obligations and/or

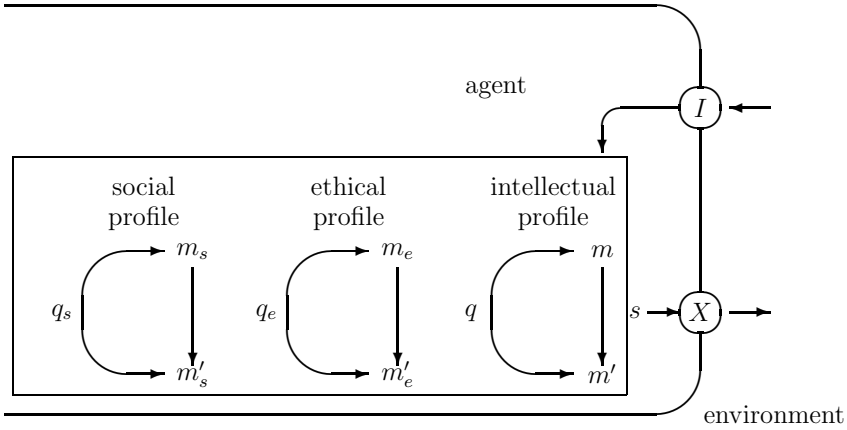


Fig. 2. Concept of social, ethical and intellectual profiles in M-agent architecture

prohibitions. A policy is designed which contains basic/axiomatic rights attributed to users, which are associated with roles, rules for delegation and rules for checking the validity of requests. Security infrastructure uses x.509 certificates that allow authentication of users and agents.

The ethically-social behavior estimation presented in this paper differs much from the security infrastructure presented in [10]. We do not make any assumptions about basic or axiomatic rights of agents. We would like to propose an infrastructure in which behavior of users or agents would be evaluated. Only the behavior evaluations are the basis for deciding whether an agent or a user is trustworthy and can act inside a system. This induces the necessity of enabling all agents to get in and act for some period of time inside a protected system. Our work does not exclude the facts presented in [10] – behavior evaluation could allow to trust an agent which does not have any basic/axiomatic rights and none agent or user would like to delegate its rights to such agent. It seems that behavior evaluation could allow to resign the axiomatic rights that have to be attributed to users or agents in *Vigil* security infrastructure.

5 SOCIAL PROFILE

Social profile is a class of agent activity with the goal to observe other agents in a society and possible other elements of environment. Those observations should be done to distinguish individuals, which do not fulfil social and ethical rules and whose behavior is unfavorable, incorrect or bad for the observer. Such distinguished *bad* individuals should be adequately treated (e.g. convicted, avoided); this should also be formed by social profile.

Social profile is defined as $a_s = (M_s, Q_s, L_s)$, where:

- M_s is the set of social states m_s of agent a,
- Q_s is the configuration of goals q_s of agent's a social profile,
- L_s is the operator of social observation $L_s : M_s \times V \to M_s$.

5.1 Social State

Social state m_s of agent a is represented as a vector $m_s = [m_s^1, m_s^2, \ldots, m_s^{j-1}, m_s^j]$, where:

- j is the number of neighboring agents; neighboring agents are agents, which are visible for agent a,
- m_s^k is the factor subordinated to neighboring agent number k. This factor can be a number of any range.

Example 1. $m_s = [1, 5, 2]$ – agent a noticed 3 neighboring agents, Agent 1 is *good*, Agent 2 is *bad* and Agent 3 is "rather good".

5.2 Social Observation Operator

To distinguish *bad* individuals (estimation of behavior), immunological system mechanisms can be used. Immunological intruders detection in a computer environment has to be done on the basis of certain characteristic structures. In the case of behavior observation, such structures can be chains of actions made by an observed agent; the chains have length l.

Example 2. If possible actions of an agent in an environment will be indicated by `A`, `B` and `C`, and length of chains will be set to $l = 3$, then example of such action chain can be (`ABA`) or (`CCA`).

5.2.1 Observation of Neighboring Agents' Actions

The way in which agent a will recognize (notice, but not estimate) the action undertaken by neighbors should be defined. Intellectual profile of agent a can be used. Agent a observes environment V and creates model $m = I(V)$. Event Z changes model m_k in m_l. Every action taken by neighbor can be described as change of the environment model of agent's a intellectual profile. The same event Z can be the result of different agents' actions, which potentially can change m_k in m_l.

Observed event Z transposes itself onto action D of agent a, when the performer of this event is known. The performer can be recognized when signs of the unit, which has changed model m, will remain in environment V.

5.2.2 Generation of Detectors' Collection

In order to generate collection of detectors R, own collection W is necessary. This collection includes correct – *self* resources. This collection W should consist of actions' chains undertaken by the agent – observer. This is correct due to the assumption that actions undertaken by an agent are evaluated by the agent as *good*. Those actions are the result of influence of three profiles: intellectual, ethical and (to a lesser degree) social. Social profile does not influence the actions of an agent so strongly, because it forms "sociable life" – for example, with whom does an agent co-operate or whom does an agent avoid. Observed actions primarily indicate the ethical rules of observed agent and its intellectual goals.

Example 3. Agent can undertake actions `A`, `B` and `C`. Let us assume that intellectual profile orders the agent to execute any operations and its ethical norms forbid repeating the same operation. The agent takes actions in a different order, never repeating any of them. This is the agent's collection $W = \{$(`ABA`), (`ABC`), (`ACA`), (`ACB`), (`BAB`), (`BCA`), (`BCB`), (`CAB`), (`CAC`), (`CBA`), (`CBC`)$\}$, assuming length of chains $l = 3$.

5.2.3 Detector Generation Algorithm

The detector generation algorithm refers to the method used to generate T-lymphocytes. From the collection of randomly generated chains R_0 the chains which match any chain from collection W are rejected. Chains from collection R_0, which will pass such negative selection, create the set of detectors R. This process is shown in Figure 3.

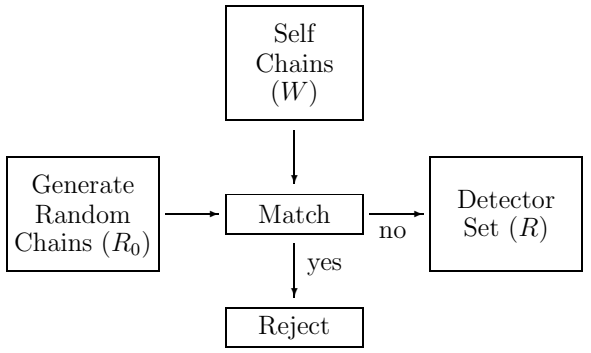


Fig. 3. The detector generation process

Example 4. W – collection from example 3; $R_0 = \{$(`AAA`), (`AAB`), (`ABA`) ... (`CCC`)$\}$ – random generated chains of length 3; $R = R_0 - S = \{$(`AAA`), (`ABB`), (`BBA`), (`BBB`), (`BAA`), ..., (`CCC`)$\}$ – collection of detectors.

5.2.4 Behavior Estimation of Neighboring Agents

The first stage is neighbor observation, during which actions (and their order) executed by neighboring agents are remembered. Those remembered actions create sequence N of any length. Every subsequence n of length l of sequence N is compared with every detector r from set R, as shown in Figure 4. If chains n and r match, it means *bad*, unfavorable actions.

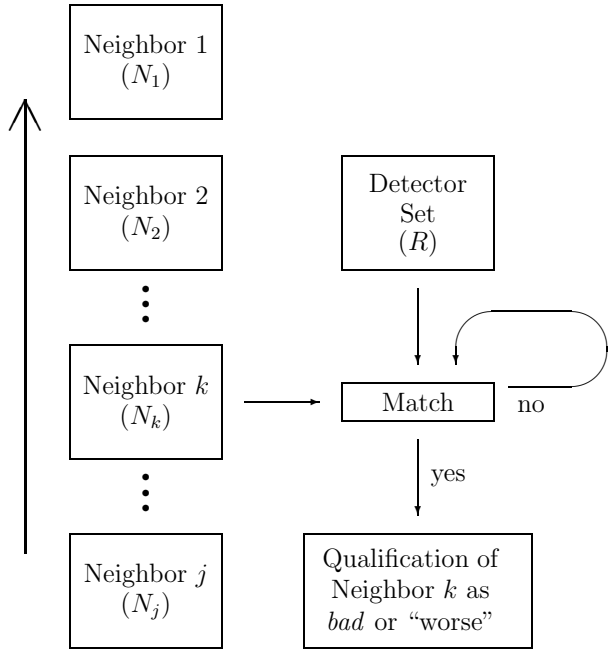


Fig. 4. Process of neighboring agents behavior estimation

The number of matches and of detectors which match should be counted. On this basis, the behavior should be estimated – social state m_s of agent-observer is modified to state m'_s. Influence on social state m'_s should also have the existing state m_s because of the operator $L_s : M_s \times V \to M_s$.

Example 5. Agent a notices 3 neighboring agents and its social state $m_s = [1, 5, 2]$; it starts the behavior estimation process of the neighboring agents:

- Agent 1 undertook actions `BAACBABCBCA` – 2 matches,
- Agent 2 undertook actions `BACABCBCACA` – 0 matches,
- Agent 3 undertook actions `BAAAAACBCAA` – 6 matches.

Social state of agent a is modified: $m'_s = [1, 3, 5]$.

5.3 Treatment of Neighboring Agents

The way how neighboring agents are treated is described by Q_s – configuration of goals q_s of agent's social profile. Configuration of goals of an agent is constant, with possible goal adaptation. Goal adaptation means the situation, in which one agent adapts the goals of another agent or of a group of agents.

Example 6. Configuration of goals q_s of agent a:

- indifferent treatment neighboring of agent number g, if $m_s[g] \leq 2$,
- avoidance of the neighboring agent number h, if $m_s[h] > 2$,
- liquidation of the neighboring agent number i, if $m_s[i] > 4$.

Continuing example 5, the agent with the above goals and state $m'_s = [1, 3, 5]$ will indifferently treat Agent 1, avoid Agent 2 and try to liquidate Agent 3.

6 ETHICAL PROFILE

Ethical profile is a class of agent activity with the goal to fulfil one's ethical rules and update or learn the ethical rules on the basis of environment observation, particularly of those environment elements which are agents.

Ethical profile of agent a is defined as $a_e = (M_e, Q_e, L_e)$, where:

- M_e is the set of ethical states m_e of agent a,
- Q_e is the set of goals and "anti-goals" (ethical rules) q_e of agent's ethical profile,
- L_e is the operator of agent's ethical learning $L_e : Q_e \times V \rightarrow Q_e$.

The operator of agent's ethical learning allows to change ethical standards in order to be "better" treated in a society.

6.1 Possible Simplification

The above ethical-social model can also function without the operator of agent's ethical learning. In this situation agents, which:

- are qualified as *bad*,
- are in minority,
- are not able to change ethical norms (without operator of agent's ethical learning)

would be *eliminated*. It should be taken into consideration that the notion *good*/*bad* is relative and depends on estimating agent's ethical rules. It can happen that *bad* (for anyone) agent will survive ("win") because of existing superiority of *bad* agents' group as to number.

Example 7. There are 5 agents in the environment. They can undertake actions `A`, `B` and `C`. Let's assume that:

- agent 1:
  - the intellectual profile orders the agent to execute any operations,
  - its ethical standards do not forbid repeating the same operation;
- agents 2, 3, 4 and 5:
  - the intellectual profile orders them to execute any operations,
  - the ethical norms forbid repeating the same operation.

Agent 1 undertakes actions `BBBB...`, so its behavior is estimated by the neighboring agents (more exactly: their social profiles) as *bad* (e.g. $m_s[agent1] = 5$ for every neighbor in the environment). Let us assume that all agents have the configuration of social goals q_s shown in example 6. Agents 2, 3, 4 and 5 will try to liquidate agent 1. Agent 1 will be destroyed.

7 SOCIAL PROFILE IMPLEMENTATION

In our simulations the above social profile was implemented. The goal of social profile functioning is to divide agents into *bad* or *good* agents in a society of agents. For that reason the social profile is named division profile in this implementation. In this profile, the main assumptions about a social profile are fulfilled; however, there are some additional aspects related with the implementation. These aspects are given below.

7.1 Detectors

The detector generation method is analogous to that presented in the social profile section. In order to generate a set of detectors R, own collection W should be specified. This collection W should consist of action-object sequences of length l, which are undertaken by the agent-observer. Presuming that h last actions undertaken by every agent are stored, own collection W will contain $h - l + 1$ elements. From set R_0 of generated sequences of length l those reacting with any sequence from collection W are rejected. Set R_0 contains every possible sequence (but it is also possible to use a set of randomly generated sequences). Sequence reaction means that elements of those sequences are the same. Sequences from set R_0 which will pass such a negative selection create a set of detectors R.

7.2 Behavior Estimation of Neighboring Agents

The first stage is neighbor observation during which the actions (and their order) executed by neighboring agents are remembered. Those remembered actions create

sequence N of presumed length h. After the next stage of detectors generation, the generated detectors are used to find *bad*, unfavorable agents. Every subsequence n of length l of sequence N is compared with every detector r from set R, as shown in the social profile section in Figure 4. If sequences n and r match, it means that *bad*, unfavorable actions have been found. Sequence matching means that the elements of the sequences compared are the same.

The number of matches for every observed agent is counted. On this basis behavior is estimated – division state $m_d = (m_d^1, m_d^2, \ldots, m_d^{j-1}, m_d^j)$ of agent-observer is modified to the $m_d{}' = (m_d^{1'}, m_d^{2'}, \ldots, m_d^{j-1'}, m_d^{j'})$, where j is the number of agents in the environment, $m_d^{k'}$ is assigned to the number of counted matches for agent number k.

7.3 Configuration of Agent's Division Profile Goals

The way neighboring agents are treated is described by Q_d – configuration of goals q_d of agent's division profile. In the implemented system the configuration of goals consists of only one goal – liquidation of neighboring agent (or agents) number k, if $m_d^k = \max(m_d^1, m_d^2, \ldots, m_d^{j-1}, m_d^j)$.

7.4 Configuration of Agent's Division Profile Strategies

Tha actions which should be undertaken by agent a in order to treat agent number k in the way described by the goal configuration are specified by S_d – the configuration of strategies s_d of agent's division profile. The configuration of strategies of the agent is constant, and in the system described the configuration of strategies consists only of one goal: if the goal is to liquidate agent number k, a demand of deleting agent number k is sent to the environment (coefficient o_d equal to the m_d^k is attributed to this demand).

This configuration of strategies presumes an intervention of system's environment in the liquidation of the agent. In the system described, the environment calculates the sum of coefficients for every agent separately attributed to demands, and liquidates all agents which have the maximum sum of coefficients; this sum is larger than constant OU. Periodically, after a constant time period, the calculated sums of coefficients are set to 0. The constant coefficient OU is introduced in order to get tolerance for the behavior evaluated as *bad* in a short time, or is evaluated as *bad* by a small number of agents.

8 EXPERIMENT

In the computer system there are some operations which must be executed in couples, such as open and close a file, connection request and disconnection request. There are a lot of attack techniques which consist in doing only one part from a couple (or trio...) of obligatory operations (for example so-called SYN flood attack [14]). A system with two types of agents is simulated:

- type g=0 agents – good agents which perform some operations in couples (e.g. open, open, open, close, open, close, close, close);
- type g=1 agents – bad agents (intruders) which perform only one from a couple of some operations (e.g. open, open, open, open, open, open, open, open).

In the simulation there is no possibility of distinguishing the type of an agent on the basis of the agent's structure. Thus, the only possibility to distinguish whether the agent is good or bad is to observe the agent's behavior and process the actions observed (actions-objects).

8.1 Results: Intruders Inside an Environment

In this part of research three cases were addressed:

- a case with only type $g = 0$ agents in the system without division profile mechanisms – initially there are 50 type $g = 0$ agents in the system, which do not have any security mechanisms;
- a case with type $g = 0$ agents and type $g = 1$ agents without division profile mechanisms – initially there are 35 type $g = 0$ agents and 15 type $g = 1$ agents, all agents do not have any security mechanism;
- a case with type $g = 0$ agents and type $g = 1$ agents with division profile mechanisms – initially there are 35 type $g = 0$ agents and 15 type $g = 1$ agents, all agents in the system are equipped with the division profile mechanisms with parameters $h = 18$, $l = 5$, $OU = 300$.

In those three cases, the system was simulated to 300 time periods and 10 simulations were performed. The diagram in Figure 5 shows the average numbers of agents in separate time periods.

In the two cases of system with agent without division profile mechanisms: if there are not any intruders in the simulated system, all type $g = 0$ agents can exist without any disturbance. The existence of intruders in the system causes problems with executing tasks of type g=0 agents which die after some time periods. *bad* agents still remain in the system that is blocked by those *bad* agents.

In the case of system with agent with division profile mechanisms: in the environment last 18 actions undertaken by every agent are stored. After 18 actions have been undertaken by every agent, detectors of length $l = 5$ are constructed. Agents use their division profile mechanisms to calculate which neighboring agent they want to eliminate. Agents demand to eliminate the neighbors which have the maximum of detector's matchings. Agents present their demands to the environment with the number of matchings. The environment counts matchings in the demands presented and eliminates agents as it was presented in the description of division profile mechanisms. The constant OU is set up to 300.

As proved by the results presented in Figure 5, after detectors were constructed, intruders were distinguished due to the division profile mechanisms. Simultaneously,

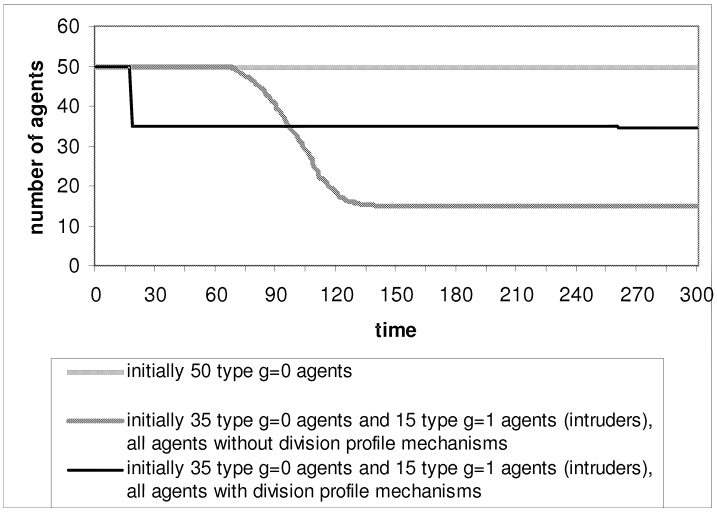


Fig. 5. The system without intruders, intruders inside the system, intruders inside the system with agents with built-in division profile

the distinguished agents were deleted; this allows type $g = 0$ agents to function freely. More details of the functioning division profile mechanisms in this test were presented in [4].

8.2 Results: Intruders Penetrating the Environment

Mobile intruders were simulated – *bad* agents which can get to the agent system. Initially there are 40 type $g = 0$ agents. After 20 constant time periods new agents get into the system. In every time period one intruder is getting into the system; this process occurs to 80^{th} time period, thus 60 type $g = 1$ agents are getting. Two cases were simulated:

- agents without any security mechanisms;
- all agents equipped in division profile mechanisms with parameters $h = 18$, $l = 5$, $OU = 300$ (mobile agents are also equipped in division profile mechanisms) – after 18 actions have been undertaken by every agent, detectors are constructed, so after this had happened the agents can distinguish *bad* and *good* agents.

The system was simulated to 300 time periods and 10 simulations were performed. The diagram in Figure 6 shows the average numbers of agents in separate time periods.

Knowing that all mobile agents are *bad*, it seems that all getting agents should be immediately eliminated. This is not true, because agents distinguish *bad* agent on the basis of behavior estimation. It is possible to evaluate agents which have

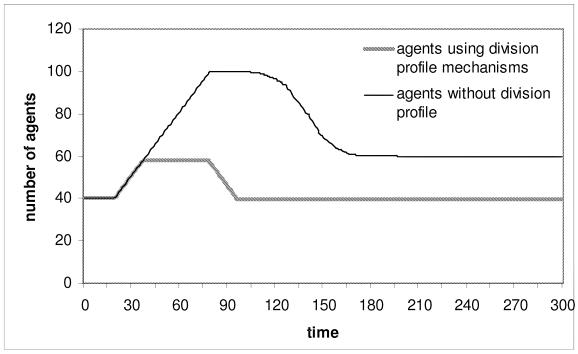


Fig. 6. The system with mobile intruders penetrating the system, agents without or with build-in division profile

presented behavior – have undertaken 18 actions required by division profile mechanisms. Every new *bad* agent getting to the system is destroyed after it has presented its behavior – has undertaken 18 actions. Because agents undertake one action in one constant time period Δt, every new *bad* agent is killed after 18 constant time periods Δt of its functioning.

8.3 Results: Heterogeneous Acting Intruders

Tests with type $g = 1$ agents which act in more heterogeneous way as in previously presented tests were made as well. *Bad* agents perform two of several possible operations (like type $g = 0$ agents), but one of these operations is mostly undertaken. Type $g = 1$ agents perform one operation in 90 per cent of all cases and perform another operation in 10 per cent of all cases (type $g = 0$ agents perform one operation in 50 per cent of all cases and perform another operation in 10 per cent of all cases). Two cases were simulated:

- agents without any security mechanisms;
- all agents equipped in division profile mechanisms with parameters $h = 18$, $l = 5$, $OU = 300$.

The system was simulated to 300 time periods and 10 simulations were performed. The diagram in Figure 7 shows the average numbers of agents in separate time periods.

The existence of heterogeneous acting intruders causes problems with executing tasks of type $g = 0$ agents. The problems are similar to those presented for intruders that perform only one action. As verified by the results presented in Figure 7, distinguishing of *bad* agents is not so rapid as in the case of homogeneous acting intruders. This effect is caused by the heterogeneity of actions which form the behavior of intruders. It can happen for some time periods that the behavior of

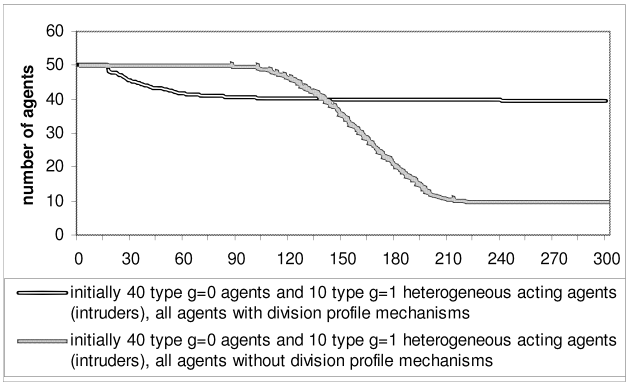


Fig. 7. The system with heterogeneous acting intruders, agents without or with built-in division profile

heterogeneous intruder is very similar to that of type $g = 0$ agent. This effect is intensified by the random mechanism of decision making about performed actions. Intruders are distinguished in time periods in which their last 18 undertaken actions are different enough from the actions undertaken by type $g = 0$ agents ($h = 18$). After all *bad* agents were distinguished and deleted, all type $g = 0$ agents function freely.

9 CONCLUSION

This article presents the concept of using the mechanisms acting in societies, which permit the functioning of an individual facing a danger coming from other members of the society. It is proposed to equip every computer resource with such mechanisms, which provide safety of the system and its components without centrally acting tools of an administrator. Such mechanisms should also enable dynamic adaptation to newly arising threats. The security solutions presented form new security paradigms which could be stated as follows:

- equip all system resources (e.g. agents, programs) with security mechanisms,
- security mechanisms should be based on activity observation rather than looking for some fragments of code (signatures),
- design the environment of computer system in such a way so as to support security mechanisms with which system's resources are equipped.

In this paper, security mechanisms with immunological approach were presented which fulfil the above security paradigms. All these security mechanisms were called a division profile. The conception presented was simulated, and the results obtained confirm the effectiveness of this solution. The simulation enables to anticipate how the described mechanisms will function in the real world of computer systems.

Security mechanisms designed on the basis of the presented conception have such advantages as detection of previously unseen danger activities, detection based on activity observation, and decentralized detection.

REFERENCES

[1] CERT Coordination Center: Overview of Attack Trends. Carnegie Mellon University, available on: `www.cert.org`, 2002.

[2] Cetnarowicz, K.: M-Agent Architecture Based Method of Development of Multi-agent Systems. Proc. of the 8th Joint EPS-APS International Conference on Physics Computing, ACC Cyfronet, Krakow, 1996.

[3] Cetnarowicz, K.—Nawarecki, E.—Żabińska, M.: M-Agent Architecture and Its Application to the Agent Oriented Technology. Proc. of the DAIMAS '97, St. Petersburg, 1997.

[4] Cetnarowicz, K.—Rojek, G.: Unfavourable Beahvior Detection with the Immunological Approach. Proceedings of the XXVth International Autumn Colloquium ASIS 2003, MARQ, Ostrava, 2003, pp. 41–46.

[5] Cetnarowicz, K.—Rojek, G.—Werszowiec-Plazowski, J.—Suwara, M.: Utilization of Ethical and Social Mechanisms in Maintenance of Computer Resources' Security. Proceedings of the Agent Day 2002, Belfort, 2002.

[6] Forrest, S.—Perelson, A. S.—Allen, L.—Cherukuri, R.: Self-Nonself Discrimination in a Computer. In Proc. of the 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos, 1994, pp. 202–212.

[7] Forrest, S.—Perelson, A. S.—Allen, L.—Cherukuri, R.: A Change-Detection Algoritm Inspired by the Immune System. IEEE Transactions on software Engineering, IEEE Computer Society Press, Los Alamitos, 1995.

[8] Gibbs, W. W.: How to Survive in Dangerous World? Świat nauki, Wydawnictwo Prószyńska i s-ka, 2002, in Polish.

[9] Hofmeyr, S. A.—Forrest, S.: Architecture for an Artificial Immune System. Evolutionary Computation, Vol. 7, 2002, No. 1, pp. 45–68.

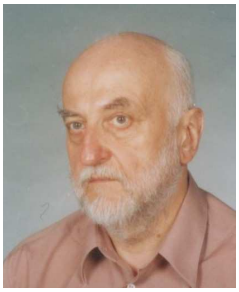
[10] Kagal, L.—Undercoffer, J.—Perich, F.—Joshi, A.—Finin, T.—Yesha, Y.: Vigil: Providing Trust for Enhanced Security in Pervasive Systems. University of Maryland, Baltimore County, available on: `citeseer.ist.psu.edu/kagal02vigil.html`, 2001.

[11] Kim, J.—Bentley, P.: Negative Selection within an Artificial Immune System for Network Intrusion Detection. The 14th Annual Fall Symposium of the Korean Information Processing Society, Seoul, October 13–14 2000.

[12] Ossowska, M.: Moral norms. Wydawnictwo Naukowe PWN, Warszawa, 2000, in Polish.

[13] Ricken, F.: General Ethics. Wydawnictwo ANTYK – Marek Derewiecki, Kety, 2001, in Polish.

[14] Schetina, E.—Green, K.—Carlson, J.: Security in Network. Wydawnictwo HELION, Gliwice, 2002, in Polish.



Krzysztof Cetnarowicz received M. Sc. in electrical engineering at the Faculty of Electrotechnics, Automatics and Electronics of the AGH – University of Science and Technology of Cracow in 1971. In 1976 he graduated in mathematics at the Faculty of Mathematics, Physics and Chemistry of the Jagiellonian University in Cracow. In 1977 he obtained the Ph. D. degree at the Faculty of Electrotechnics, Automatics and Electronics of the Stanislaw Staszic AGH – University of Science and Technology of Cracow. He is professor at the Institute of Computer Science at the AGH – University of Science and Technology of Cracow. He is the author of patents and more than 60 papers in computer science (multi-agent systems, artificial intelligence, evolution of multi-agent systems, image processing, simulation).

Gabriel Rojek received the M. Sc. in computer science at the Faculty of Electrotechnics, Automatics, Computer Sciences and Electronics of the AGH – University of Science and Technology of Cracow in 1999. In 2004 he obtained the Ph. D. degree at the Faculty of Electrotechnics, Automatics, Computer Sciences and Electronics of the Stanislaw Staszic AGH – University of Science and Technology of Cracow. He works in the Department of Computer Science in Industry at AGH University of Science and Technology. He is the author of papers in computer science in the field of multi-agent systems and computer security.