ARTIFICIAL IMMUNE SYSTEM FRAMEWORK FOR PATTERN EXTRACTION IN DISTRIBUTED ENVIRONMENT

Rafał Pokrywka
Institute of Computer Science, AGH University of Science and Technology, al. Mickiewicza 30, 30-059 Kraków, Poland
Softq s.c. Szuwarowa 8/40, 30-384 Kraków, Poland

Keywords: Pattern extraction, Artificial immune system, Multiagent system.

Abstract: Information systems today are dynamic, heterogeneous environments and places where a lot of critical data is stored and processed. Such an envrionment is usually build over many virtualization layers on top of backbone which is hardware and network. The key problem within this environment is to find, in realtime, valuable information among large sets of data. In this article a framework for a pattern extraction system based on artificial immune system is presented and discussed. As an example a system for anomalous pattern extraction for intrusion detection in a computer network is presented.

1 INTRODUCTION

In heterogeneous environments like information systems, with multiple virtualization layers and service providers build on top of hardware and network, the quantity of information is huge. Obviously information is constantly processed through various workflows and procedures which creates, at the end, value for the end user. But in particular situation only relevant fragments of information carry the core data which is interesting within particular context and use case. Those fragments are buried in overflow of irrelevant details. Additionally useful information can also be that who, what and when sends, receives and process data. Some parts of general information about activities stays on sender, receiver and within transporting infrastructure but what is often missed during data processing and analysis is the correlation between them.

In this article a framework for pattern extraction system is proposed. The framework architecture is based on multiagent system concept. There are also analogies to the natural immune system used - introduction to immune system is provided below. The system consists of a common layer that abstracts infrastructure specific information and creates distributed environment for a second layer of agents which are able to make decisions and are also able to migrate through the whole network carrying necessary piece of extracted information. Information retrieval is not always correct in a sense that systems like this sometimes miss to correctly extract and classify data. Such errors can be potentially dangerous especially in intrusions detection and are called false alarms or type I errors. This problem is handled by the architecture through applying a method for reducing false alarms rate, called Layered Filtering which is described in (Pokrywka, 2008). The method helps to decrease a base-rate fallacy influence on a false alarm rate. The fallacy is connected with base rates of normal and abnormal events and is directly implicated by Bayes theorem. It has been pointed out by Axelsson in (Axelsson, 2000).

2 OVERVIEW OF THE NATURAL IMMUNE MEMORY

The human immune system is designed to protect an organism against pathogens. The most important elements of the system are lymphocytes which in great number circulate through the body and detect foreign cells. Basically there are two types of lymphocytes: T and B. They both cooperate in detection and elimination of pathogens. They also cooperate in creation of immune memory which we would like to model.

Immune memory is created during the primary immune response. This response is usually slow and has low intensity. The secondary immune response is initiated by *memory cells* and is very fast, intensive and often runs without any clinical symptoms

Pokrywka R. (2010).
ARTIFICIAL IMMUNE SYSTEM FRAMEWORK FOR PATTERN EXTRACTION IN DISTRIBUTED ENVIRONMENT.
In *Proceedings of the 5th International Conference on Software and Data Technologies*, pages 179-183
DOI: 10.5220/0002866001790183
Copyright © SciTePress

of infection. Additionally the fact that it is triggered by infection with new, but similar to previously seen, pathogens means that immune memory is associative.

The primary response consists of several steps. First the lymphocyte of type B detects newly encountered pathogen, which is presented by antigen processing cell (APC). APCs are the first cells which processes pathogens in a certain way - they look for small chunks of amino acid chains and bind them to molecule complex (MHC) which is then presented to lymphocytes. Next B-lymphocytes receive confirmation of positive detection from lymphocytes T, which are activated only in the presence of cytokines molecules. Cytokines are signals of the situation in the body and are released for example when body tissue is damaged. This confirmation from T-lymphocytes is called co-stimulation. After activation B-lymphocyte starts to mutate with a very high rate until it becomes specific to this one particular kind of pathogens. After the completion of this step the specific B-cell proliferate and differentiate into plasma cells and *memory cells*. The former secrete antibodies which participate in elimination of the pathogen, the later *memorise* the pathogen. Memory cells do not need any co-stimulation to become activated and to mount the secondary response, during which they rapidly proliferate and differentiate into plasma cells. This results in a fast increase of number of antibodies and a very fast elimination of the threat. A detailed description of the entire immune system can be found in (Hofmeyr, 2000), (Wierzchoń, 2001) and (Kim, 2002).

3 MULTIAGENT SYSTEM FOR PATTERN EXTRACTION

The system is explained a little more by example of an Intrusion Detection System extracting patterns of anomalies. Let's consider a sample network. On each host there are installed several agents called Detector Agents or simply Detectors. These agents forms first layer of system events processing and follows analogy to APC cells of immune system. Processed events are then send further to the Mobile Agents layer or better: lymphocytes. These agents are not limited to information from detectors from one host but can use information from agents placed around the entire environment and on each virtualization layer - hence distributed. The idea is that these agents can view all data and capture correlated events from the whole network. Figure 1. presents schema of the system.

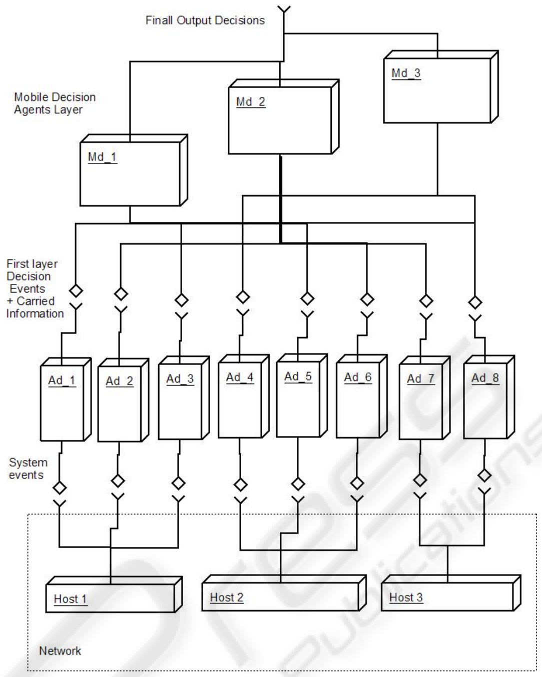

Figure 1: Layered filtering schema.

3.1 APC Detector Agents

For the example Intrusion Detection System (IDS) detectors $Ad_1, ..., Ad_n$ are agents installed on particular hosts and responsible for detecting anomalies within one aspect of the system activities. These can include:

- process's system calls (grounds for this choice can be found in (Warrender et al., 1999))
- process's opened file handlers
- process's opened ports
- list of processes

These agents signal symptoms of anomalies and outputs additional event information like its source, outcomes, etc. The Detection Rate is important in case of these agents and it should be as high as possible. However the requirements for False Alarm Rate can be relaxed significantly - following results from (Pokrywka, 2008). The agent population is constant on each host and does not change with time. Each detector can be of one particular type - which is important as types of detector are used by the second layer. These agents decrease the event quantity for processing by upper layer and changes significantly the base rates of events (see (Pokrywka, 2008) for details).

3.1.1 Example Detector in Anomaly Extraction System

The example detector (APC cell) is responsible for monitoring system calls made by process. Analysis of conditional entropy of the sequences suggests that it is reasonable to choose one of the statistical model which incorporates conditional probability to build a profile of system calls sequences. Markov chain with variable order is the choice here. This model provides a probability of the next system call having seen previous n. In this case n is a length of history and depends on the actual context — hence variable order. The profile is created during the training phase from the sequences of system calls representing normal behaviour according to the algorithm presented in (Ron et al., 1996). An overview of Markov models can be found in (Bengio, 1999). The detector provides a probability of the next system call.

3.2 Lymphocytes Mobile Agents

Mobile agents $Md_1,...,Md_n$ forms the second layer of the system architecture. They use signals from detectors as a trigger for further processing of events. Initial mobile agent population is empty. The first agent is born after first signal from a detector. A newly born agent captures all processed information from all APC detectors in a given time slot T_s - it can be suspected that these events are correlated and a system wide information could be captured. The captured pattern becomes the system output. Because of the possibility of wrong information corellation - a false alarm - this newly created agent needs a second signal to become fully functional. For now this signal comes from external source - usually an administrator. After the administrator's decision the agent is turned into one of the two types:

- *pattern specific* — in case when valuable pattern has been found and in the same time it becomes an immune memory agent.
- *anergic* — in case it was a false alarm. This agent's aim is to remember false alarm pattern and prevent future appearances of this pattern in system output.

3.2.1 Life Energy

To maintain a stable in number population of mobile agents and also to build a population which can dynamically adapt to the most current information maintained by the environment an idea of life energy is used. At the beginning an agent receives some fixed amount of life energy E_p. In each step the energy is decreased by 1. A certain probability of death of the agent is also determined, which depends on the actual level of the agent's life energy. The probability is given by the following function:

$$p(x) = \begin{cases} -\frac{1}{E_s}+1 & \text{for} x < E_s \\ 0 & \text{for} x \geq E_s \end{cases} \tag{1}$$

where E_s is an arbitrarily chosen value. When the agent happens to detect something its life energy increases also by arbitrarily chosen value E_a. This mechanism makes it possible to delete agents which detect obsolete patterns. It also keeps alive those agents responsible for detection of frequently occurring patterns.

3.2.2 Example of a Lymphocyte Agent

The probability output of APC detector agent is mapped to some amount of penalty points. It is done through a penalty function $\xi : [0,1] \rightarrow R$. To be useful this function should give a large amount of penalty points for probability equal to 0 and small amount if the probability is close to 1. In my research the following function was used:

$$\xi(x) = \frac{a}{x+b} + c \ . \tag{2}$$

The a parameter controls how convex the function is, b and c parameters modifies the amount of points the function returns for a given probability. All three parameters must be carefully chosen. In each step the computed penalty points are added to a penalty account Ξ. To limit aggregation of history the penalty account is also multiplied by dumping factor $\zeta < 1$ (4).

$$\begin{aligned} \Xi_0 &= 0 & (3) \\ \Xi_{i+1} &= \zeta(\Xi_i + \xi_{i,i+1}) & (4) \end{aligned}$$

where $\xi_{i,i+1}$ is the amount of the penalty points computed when going from i-th step to $(i+1)$-th step. If the penalty account exceeds a certain threshold τ a pattern is stored. Figure 2 presents example plot of the penalty account and the threshold. From observations it can be concluded that each process has its own characteristic plot of behaviour that is a very important feature.

Subsequent values of the penalty account form a *signal* which is unique for a particular process and is also unique for an information pattern. It is reasonable though to remember and analyse this signal and through this recognise patterns. For this purpose a very simple neural network is used. Since the agent is (in this version) responsible only for detection of one particular pattern, and with the assumption that it is in linearly separable set, its neural network consists

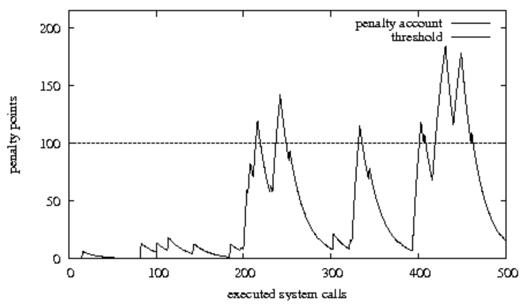

Figure 2: Example plot of the penalty account and the threshold.

of one ADALINE neuron. The pattern is represented by a vector of m last values of the process signal. The neuron also has m inputs and m input weights. For now it is assumed that the length of the pattern is constant.

The activation of the neuron is given by the following formula:

$$y = \sum_{i=1}^{m} w_i xi \tag{5}$$

where w_i is the i-th weight and x_i is the i-th input value. Inputs are often written simply as vector X and weights as vector W. From properties of ADALINE the more input vector X is similar to the weight vector W the greater the activation.

The neuron is learned through delta-rule which is an iterative process of adjusting the vector of weights according to the formula below:

$$W' = W + \eta\delta X \tag{6}$$

$$\delta = t - y \tag{7}$$

where η is the learning ratio, t is the target value that the neuron should give for this particular input vector X, y is the actual neuron's output for the input vector X. The agent's decision is positive when the current output is close enough to the target value, for example when it is $s = 99\%$ of the t. On the choice of s depends agent's capabilities of generalisation. The s parameter defines also patterns similarity. In other words the agent groups similar patterns of similar anomalies (for simplification it is assumed that patterns are linearly separable).

4 RESULTS SUMMARY

Tests of the example agent implementations have been performed on sample traces of system calls. At first a learning phase have been conducted on a fraction of samples. After that tests have been performed on rest of data. The population of pattern specific agents, which represent extracted patterns, have grown rapidly at the beginning and then stabilized which is normal as there is limited number of anomalies to extract. The information from multiple hosts have not been used in this tests so the memory agent population have been isolated for a given host.

5 CONCLUSIONS AND FURTHER WORK

In this article a framework for Multiagent Pattern Extraction system has been proposed. It emerged from previous works on Anomaly IDS system presented in (Cetnarowicz et al., 2006). The first important aspect are the properties of the first layer formed by APC detector agents which decrease events quantity and base-rates for further processing. One of the assumptions and requirements for APC detectors was that they have little memory and computing power requirements what increases processing speed. Reduced events quantity makes it possible to use more sophisticated detectors, with greater resource requirements, in subsequent layers. The second aspect is the ability to gather information from distributed environment and - with detector layers - to mimic the virtualization layers of the system under monitoring if necesary. In case of several different service providers used in monitored system adequate detector types can be used. At the end multiagent system is naturally parallel and distributed which opens wide range of possible applications.

In the presented example only two layers were used but there is no limit to that number. The results from (Pokrywka, 2008) suggest that the number of layers depends heavily on area of application as well as characteristics of detectors used.

Considering the possible implementation methods of APC detectors there could be capabilities of modern, heavilly parallelized GPUs utilized. This should greately increase overall performance.

For further improvements great source of inspiration comes from natural immune systems and it is planned to include artificial immune system algorithms into the framework like hypermutation and cytokines carring context information. The context of the event is important for its semantics and possible detector outcome.

There is still a lot of work especially in aspect of agents types and number of layers as well as corelation of information coming from different sources. Definatelly a comparative evaluation in one of the application domains is needed as well as more examples of possible application domains but for now the results seems to be promising.

REFERENCES

Axelsson, S. (2000). The base-rate fallacy and the difficulty of intrusion detection. In *ACM Trans. Inf. Syst. Secur. 3*, pages 186–205.

Bengio, Y. (1999). Markovian models for sequential data. In *Neural Computing Surveys 2*, pages 129–162.

Cetnarowicz, K., Rojek, G., and Pokrywka, R. (2006). Intelligent agents as cells of immune memory. In *International Conference on Computational Science (3)*, pages 855–862. Springer.

Hofmeyr, S. (2000). *Design Principles for the Immune System and other Distributed Autonomous Systems*, chapter An Interpretative Introduction to the Immune System. Oxford University Press.

Kim, J. (2002). *Integrating Artificial Immune Algorithms for Intrusion Detection.* PhD thesis, Department of Computer Science, University College London.

Pokrywka, R. (2008). Reducing false alarm rate in anomaly detection with layered filtering. In *International Conference on Computational Science (1)*, pages 396–404. Springer.

Ron, D., Singer, Y., and Tishby, N. (1996). The power of amnesia. learning probabilistic automata with variable memory length. In *Machine Learning 25 (2–3)*, pages 117–149.

Warrender, C., Forrest, S., and Perlmutter, B. (1999). Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy*, pages 133–145.

Wierzchoń, S. (2001). *Sztuczne systemy immunologiczne. Teoria i zastosowania. (in polish).* EXIT, Warsaw.