An Artificial Immune System and its Integration into an Organic Middleware for Self-Protection

Andreas Pietzowski
Institute of Computer Science
University of Augsburg
86159 Augsburg, Germany
pietzowski@informatik.uni-augsburg.de

Wolfgang Trumler
Institute of Computer Science
University of Augsburg
86159 Augsburg, Germany
trumler@informatik.uni-augsburg.de

Theo Ungerer
Institute of Computer Science
University of Augsburg
86159 Augsburg, Germany
ungerer@informatik.uni-augsburg.de

ABSTRACT

Our human body is well protected by antibodies from our biological immune system. This protection system matured over millions of years and has proven its functionality. In our research we are transfering techniques of a biological immune system to a computer based environment in order to design a self-protecting middleware which isn't vulnerable to malicious events. First off this paper proposes an artificial immune system (AIS) and evaluates optimal parameter settings. As the first research we show up the correlation between the size of a system and the length of the receptors used within antibodies for an efficient detection. Further on we describe the integration of the immune system into our organic middleware AMUN and afterwards we propose optimization techniques to minimize the memory space needed for storing the antibodies and to speedup the time needed for detecting malicious objects.

Categories and Subject Descriptors: C.2.4 [Computer Systems Organization]: Computer-Communication Networks, Distributed Systems

General Terms: Design

1. INTRODUCTION

To secure computer systems, current protection applications have to be aware of signatures from viruses or worms that occurred previously. Due to that restriction they cannot recognize or handle brand new intruders. Computer immunology opens up new ways and methods to recognize new intrusions like our biological immune system does and fits well into the research fields of autonomic [3] and organic computing [4] that explore the self-x properties. In our organic ubiquitous middleware research [5] we investigate self-protection techniques to cope with intentionally or unintentionally malicious peers or services. The goal of the biologically inspired technique is to develop an immune system which is permissive to good-natured middleware services and messages but can detect appearing malicious events as a first step towards self-protection.

2. THE DESIGN OF THE AIS

The basic requirement of a biological immune system is to distinguish between harmless objects called *selfs* and harmful objects or antigens called *nonselfs* [2]. In a computer-based immune system the matching works on bit strings (instead of protein patterns in the biological counterpart) and it should distinguish between self and nonself messages. We abided by the highly distributed biological paradigm to avoid failures when a central node is unreachable. Like in our body we built a thymus which is aware of all the S harmless self messages known in the system [1]. That is possible due to our middleware architecture [5]. The receptors of the antibodies are represented by a bit string of length R like in [2] but the antibodies also have a specific offset O where they start their comparison to the messages. Different values have to be considered to optimize the detection of nonselfs: The length of the self messages, the receptor length of the antibodies, the amount of different offsets among all antibodies where they start the comparison, and the choice and amount of antibodies distributed in the system. To evaluate our immune system approach we implemented the AIS into our middleware AMUN [5] and also separated key components from the system and built a simulation environment for faster evaluations.

2.1 The length of the messages

Instead of grappling around with messages of different and maybe infinite length we transformed all the messages appearing in the system to a unified length L by using a hash algorithm[1]. In this paper the word *message* is always meant to represent the hashed message.

2.2 The optimal length of the receptors

For determining the optimal length of the receptors we set up different simulator configurations with different amounts of self messages and tested the recognition of antibodies with different receptor lengths. These experiments demonstrated that it is not efficient to choose antibodies with a fixed or even a random length for the receptors. Instead it shows that this value has always to be adjusted due to the amount of self messages known in the system. With mathematical considerations we can also show that the receptor length should be approximately the binary logarithm of the amount of known self messages ($R \approx log_2(S)$) to reach a suitable detection rate. The best value depends on how the bit patterns are distributed within the self messages.

[1]We used MD5 to get messages with 128 bits in length. In case of too many duplicates another hash algorithm with more than 128 bits should be chosen.

2.3 The amount of offsets

In our research we tested different kinds of offsets within the messages in order to get knowledge about this parameter. To put it in a nutshell the best method is to use every possible offset from position 0 to $L - R$. That way it is nearly impossible for intruders to use the existing holes [2] for intrusion because the receptors overlap witin regions of the message.

2.4 The right choice of antibodies

When not generating all possible antibodies, it is important how many antibodies to create and at which offsets. In our middleware we are able to generate the antibodies in a structured way because we are aware of all self messages. Our simulation tests showed that the best detection rate can be reached when creating an equal amount of antibodies for every offset. The experiment also showed that the detection of nonselfs lead up to 99.3% when using 5000 antibodies of a receptor length of nine bits and equal distributed offsets among all antibodies in an environment of 1000 self messages.

3. INTEGRATION IN AMUN

The Autonomic Middleware for Ubiquitous eNvironments (AMUN) is a message-based middleware based on the peer-to-peer system JXTA [5]. To realize the self-x properties from organic computing [4] the middleware is extended by an organic manager and interfaces which add monitoring capabilities. Different services are started among the nodes of the system depending on a given configuration and communicate by messages. Message monitors can operate on incoming or outgoing messages and can check the messages if they belong to self or nonself. The intrusion detection system in AMUN currently consists of three components:

The **thymus** is realized as a service in the middleware and is the backbone of the whole immune system. It has the ability to ask its local services for all message types known from the configuration, to ask remote thymuses about the message types used on that node, and to generate antibodies with respect to the systemwide known message types and to suggest a suitable receptor length of the antibodies to achieve an optimal recognition rate of intrusive message types. Generating antibodies is only done on dedicated nodes because it is very time consuming.

The component called **immun** service embodies the instance which decides if some intruding message should be accepted or treated as nonself. Therefore this module is also realized as a middleware service. It is able to receive antibodies which are generated and spread by the dedicated thymuses around the middleware.

The last component in this architecture is the **intrusion detection monitor** which is placed in front of the incoming message monitor queue within a node. Every incoming message is checked automatically for its validity by sending it to the immun service. As for now the intrusion detection monitor only checks for malicious messages and can block them but does not defend against the intrusion yet. Tracking back a malicious message to its originator results in detection of a harmful service.

4. OPTIMIZATIONS

Optimizations were done in our research both within space and time issues. We propose a technique for memory requirement minimization which minimizes the amount of antibodies to be stored in the system. Antibodies with the same offset and different receptors can be merged to one antibody if they differ at only one position on the receptors. In that case a new antibody can be created with a *wildcard* at this specific position which replaces the others. Also antibodies with wildcards can be merged together when two receptors have the same wildcard positions and differ at exactly one further position. Unlike in biology we have to cope with some limitations due to the one-dimensional architecture of computer memory. To achieve a more efficient comparison than walking through an array we store the antibodies in binary trees because many antibodies correspond to other antibodies in specific parts of their receptor. If there is a complete way starting from the root down to the bottom of the tree one antibody stored in that tree matches the incoming pattern. With the binary tree we could decrease the time complexity to $O(P \cdot R)$ where P is the amount of possible offsets.

5. CONCLUSIONS AND FUTURE WORK

The tests showed that the recognition rate was very good when choosing a suitable receptor length and an appropriate amount of antibodies. We also looked closer at the speed of detecting selfs and non-selfs and found a way to reach a speed up of 30 by using a binary tree and we were able to eliminate about 30% of the antibodies by merging them together. Our middleware AMUN is extended by an organic manager and interfaces which add monitoring capabilities. As for now the generation and distribution of antibodies is implemented in the middleware. We tested the quality of the recognition of nonself messages and reached a recognition rate of up to 99.3%. We also proposed some improvements and optimizations which lead to lower memory usage and faster detection. In a next step we want to integrate a reaction system around the detection mechanism. Thus the middleware will be able to deactivate malicious services or shut down infected nodes in the network to prevent other nodes from the intrusion. This leads to a so-called self-healing mechanism which recovers the system after a successful defeating of malicious events.

6. REFERENCES

[1] Leandro Nunes de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Approach.* Springer-Verlag, 2002.

[2] Steven A. Hofmeyr and Stephanie Forrest. Architecture for an Artificial Immune System. In *Evolutionary Computation 8*, number 4, pages 45–68, Massachusetts Institute of Technology, 2000.

[3] Jeffrey O. Kephart and David M. Chess. The Vision of Autonomic Computing. *IEEE Computer Society*, pages 41–50, January 2003.

[4] Christian Müller-Schloer. Organic Computing Initiative. http://www.informatik.uni-augsburg.de/lehrstuehle/sik/research/organiccomputing/download/OC-english.pdf, April 2004.

[5] Wolfgang Trumler, Faruk Bagci, Jan Petzold, and Theo Ungerer. AMUN - autonomic middleware for ubiquitous environments applied to the smart doorplate. *Advanced Engineering Informatics*, (19):243–252, April 2005.