Adaptive Security Management In Application Level Active Networks

Temitope O. Olukemi, Lionel Sacks and Ognjen Prnjat

Department of Electronic and Electrical Engineering, University College London

Abstract: As research continues in active networks, the need for security cannot be over emphasised. To successfully maintain an active network it is essential that node operators be able to protect their node and code running on their node. However with the dynamic nature of these networks, a system based on static security would severely limit that dynamism. This paper looks at an approach to providing flexible security management in application level networks (ALAN) in the context of the EU/IST project ANDROID (Active Network Distributed Open Infrastructure Development). We also look at a way of making the system adaptive based on immunological modelling.

1. Introduction

An active network [1] gives users the ability to load software components dynamically without explicit reference to any third party. There are two types of active network:

1. Discrete - In this approach there is a separation between the mechanism for injecting programs into a programmable node and the actual processing of the packets as they flow through the node. Users send a program to a node and this program would then be stored at the node. When a packet arrives at the node, its header is examined and a program is used to process the packet.
2. Capsule - The capsule approach leads to a more dynamic form of active network. Each packet (capsule) in such a network contains both data and a program fragment that may include embedded data. When a capsule arrives at an active node, its data content is processed using the program in the capsule.

Much of the work so far in active networking has concentrated on the capsule approach. In the ANDROID project, we're adopting the discreet approach to active networking for two major reasons:

1. Loading of programs can be more easily controlled.
2. Functionality can be more sophisticated without the size restriction imposed by the size of the capsule.

The activeness in the active network can be provided at different layers of the OSI protocol stack. When the active capabilities are provided at the lower layers we regard this as the pure form of active networking. However when the active capabilities are provided at the higher layers (such as the application layer) we regard this as the provision of active services. The advantage of the active service approach is that operation at the lower levels is not affected and deployment can be incremental and thus faster. ANDROID is focusing on the second approach.

This paper is divided into six sections. In the first section we present an introduction to the work we are doing. In section two we describe the ANDROID architecture, in the third section we go on to describe our security management system. In section four we discuss the runtime monitoring issues in our system, in the fifth section we describe the basis of our immunological modelling, while in section six we present our conclusions and future work.

2. ANDROID Architecture

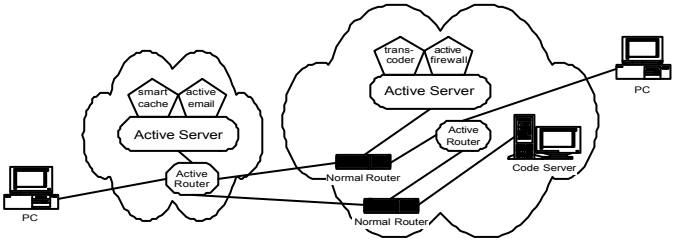

Figure 1 – ANDROID Architecture

Figure 1 shows the active architecture we're proposing in ANDROID. In addition to the usual network infrastructure we're proposing two additional devices – the active router and the active server.

Both are a form of active node. They possess the capability of running application level active services. However the services run on the active router are restricted and are chosen from a set of trusted services designated by the owner of the active router. The active server is more open and as such the security issues associated with it are greater.

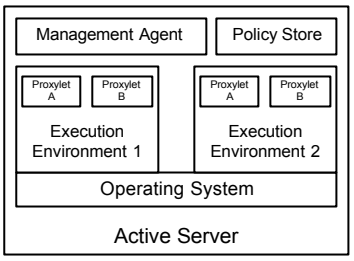

Figure 2 – Schematic of the Active Server Design

Figure 2 shows a schematic of the active server design. Each active node possesses an execution environment for proxylets (EEP). It is within these EEPs that our active code (proxylets) is run. The active server also possesses a number of management agents that handle issues such as security and resource management on that active server.

3. Security Management

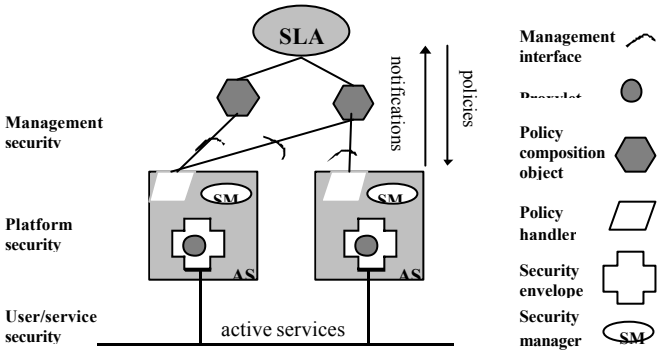

Figure 3 – ANDROID Security Architecture

The security architecture being developed for ANDROID is shown in Figure 3. Security is divided into three levels: user/service security, platform security and management security.

User/service security focuses on the end-to-end security of the active services. It involves issues such as user authentication to a service and securing inter-proxylet communications. Platform security refers to the security provided by the security manager and policies in action on the active servers. This aspect encompasses proxylet deployer authentication and access control, proxylet authentication and access control and runtime monitoring of the proxylet. The third aspect of security focuses on securing the flows of management information.

Figure 3 provides an overview of security in the ANDROID context. Service level agreements (SLAs), negotiated between service providers and active server operators are broken down into policies by the policy composition objects. There could be many layers of policy composition objects, with initially high level policies being composed and then at the lower levels, XML policies that will drive the management system being composed. The policies are fed into the active server at the management interfaces. At this point two sets of policies are fed into the system. One set of policies are associated with the deployer of the proxylet, while the second set of policies are associated with the proxylet itself. The security manager interprets the information from both sets of policies and the information they contain is used to create a security envelope within which the proxylet is run. Essentially this envelope is a Java policy file (Java is the platform programming language), which contains the access

control information pertaining to the proxylet. The security manager also performs a number of other security functions. It authenticates the proxylet deployer based on role/ID matching and it authenticates proxylets using the JAR signing (since the proxylets are Java archive – JAR files). The final task of the security manager is runtime resource usage monitoring of the proxylet.

4. Runtime Security Monitoring

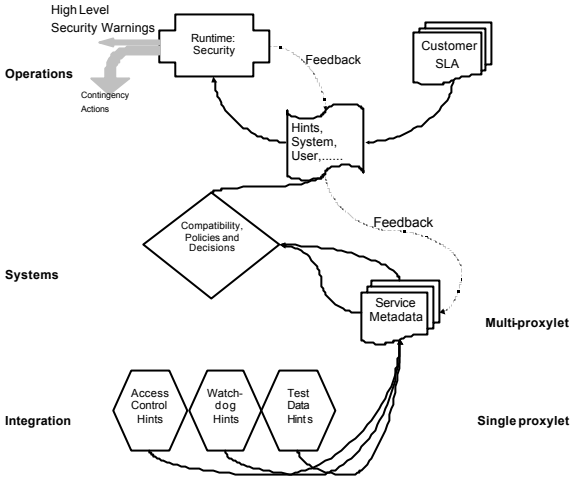

Figure 4 – Runtime Monitoring System

While Figure 3 shows how we provide the essentially static aspect of security, Figure 4 shows how the more dynamic runtime security is provided. Essentially with runtime monitoring, we want to profile the behaviour of proxylets and services. The profile of a proxylet's behaviour is built up from the integration level. At this level metadata is created for the individual proxylets. The metadata consists of hints related to the access control needs of the proxylet, and test data hints – resource utilisation information gathered from observing the proxylet running in a test environment. The information is integrated for a number of different proxylets providing a service to form the service metadata. At the systems level the metadata for the individual proxylets and the service metadata is composed into policies. At the operations level, these policies along with those created from customer SLAs are fed into the runtime security system. Warnings are generated and contingency actions are taken when anomalous behaviour is observed.

The system possesses two control loops. The first control loop feeds back information collected from the runtime system back to the policy composition phase. This particular control loop could lead to the redefinition of policies feeding into the runtime security system. A typical case would be one where the behaviour noticed was actually normal behaviour, which was previously unaccounted for in the initial policy set and so this resulted in a perfectly legitimate service or proxylet being killed.

The information feedback could continue to a lower level where the service and individual metadata for the proxylet is modified, so that future services which make use of a particular proxylet or service instance will take into account the newly noted behaviour.

In the next section we go on to describe the approach that we're taking into closing these control loops and thus make the security management system adaptive.

5. Immunological Modelling

Over the last few years there has been a great increase in interest in studying biologically inspired systems, such as neural networks and evolutionary computation. Recently there has been a growing interest in another area – immune systems. This has led to a new field of research called **Artificial Immune Systems** [2] [3] [4].

The immune systems itself is a complex of cells, molecules and organs which has proven to be capable of several tasks such as pattern recognition, learning, memory acquisition, generation of diversity, noise tolerance, generalisation, distributed detection and optimisation. Many of these are tasks, which we would want to map into our security management system. We are not trying to replicate the

immune system, what we are trying to do is develop computational techniques based on what we can learn from the immune system and apply them to our work.

Human Immune System	**ANDROID Network Immune System**
Antigen	Malicious Proxylet
Antibody	Best Solution Vector
Recognition of Antigen	Identification of anomalous behaviour
Production of antibodies from memory cells	Recalling a past successful solution
Lymphocyte differentiation	Maintenance of good solutions (memory) in database
T-cell suppression	Elimination of surplus candidate solutions
Proliferation of antibodies	Use of genetic operators to create new antibodies

Table 1 – Mapping between the Human Immune System and the ANDROID network immune system

Table 1 shows a mapping we have developed between the ANDROID network and the human immune system. A number of classical techniques already exist in the field of artificial immune systems such as self, non-self discrimination, the immune network [5], and clonal selection [6]. We are developing an extended model bringing in new ideas such as the innate and adaptive immune system and the intercellular signalling between cells in the immune system.

6. Conclusions and Future Work

In this paper we have considered the importance of security in an active network scenario where a network operator may have little or no knowledge of the state of his network at a particular instant in time. We have also presented an initial approach to tackle this issue in the context of the ANDROID project. The key area we have identified in devebping an adaptive management system how to close the control loops depicted in Figure 4. For this we are using a biologically inspired approach based on immunological modelling.

We are continuing the development of the immunological models for the security system. We plan to start developing computational models of our system and carry out simulations using Swarm [7], an agent-based modelling tool.

References

[1] D. Tennenhouse and D. Wetherall, “Towards as Active Network Architecture”, Computer Communications Review, Vol. 26, No. 2, 1996.
[2] J.E Hunt and D.E. Cooke, “Learning Using an Artificial Immune System”, Journal of Network and Computer Applications, 19, pp 189-212, 1996.
[3] D. Dasgupta, “Artificial Neural Networks and Artificial Immune Systems: Similarities and Differences”, Proceedings of the IEEE SMC, 1, pp. 873-878, 1997.
[4] S.A. Hoffmeyr, “An Interpretative Introduction to the Immune System”, In Design Principles for the Immune System and Other Distributed Autonomous Systems, (Eds.) I. Cohen and L.A. Segel, Oxford University Press, 2000
[5] N.K. Jerne, “The Immune System”, Scientific American, 229(1), pp 52-60, 1973.
[6] F.M. Burnet, “Clonal Selection and After”, In Theoretical Immunology, (Eds.) G.I. Bell, A.S. Perelson and G.H. Pimbley Jr., Marcel Dekker Inc., pp 63-85, 1978.
[7] The Swarm Development Group, Swarm, http://www.swarm.org