Active Platform Security through Intrusion Detection Using Naïve Bayesian Network for Anomaly Detection

Abdallah Abbey Sebyala†, Temitope Olukemi ‡, Dr. Lionel Sacks‡
Department of Electronic and Electrical Engineering,
University College London, Torrington Place, WC1E 7JE. England, UK
Email: mt01005@ee.ucl.ac.uk

Abstract: *Recently Active Networks were proposed as the approach that will enable quick introduction of new services in the current telecommunication infrastructure. This approach allows third party executable codes (proxylet) to be deployed into the network, which creates a big security risk. In this paper we present the application of Bayesian technology in the development of an anomaly detection system for proxylets. This system will be incorporated into our Intrusion Detection System (IDS) that will provide runtime security to ensure active platform integrity is maintained while running third party executable codes.*

Keywords: Active Network; Intrusion detection; Anomaly detection, Bayesian Network

1 Introduction

In today's fast changing competitive telecommunication environment with increasing customer demands and awareness, quick development and introduction of new services is crucial to stay ahead of your competitors. With the current telecommunication infrastructure it is becoming completely difficult if not impossible to meet the current demand for new services, because the deployment of new services is limited by the slowness of the standardization process and inflexibility of this communication infrastructure in which the transport and control function are vertically integrated. To overcome this problem an approach was proposed in form of Active Networks (ANs). This approach allows third party executable codes (proxylet) to be deployed into the network. This provides open access for service developers to service component and control that has always been in the hands of network operators. Surely this poses a problem in terms of security requirements of the systems and the network at large.

This paper presents our first attempt to develop an active platform runtime security system, it starts with an introduction to Active Network and our security architecture in section 2. Section 3 begins with a brief introduction to the concept of Intrusion Detection and highlights our Intrusion Detection System (IDS) model, giving the Anomaly detection System as the main focus of this paper. Section 4 summarises our progress and describes directions for future work.

2 Theoretical background

2.1 Active network

Active networks are characterised by dynamic programmability where the users/third party service providers can programme network hardware to manipulate information flowing through them. The network is active in the sense that it can dynamically perform computation on the user data, which creates a much more flexible network infrastructure with increased capabilities compared to traditional passive networks. In passive networks packets are just delivered from one end-point to another. Processing within the network is limited to routing, congestion control, flow control and other quality of service (QOS) schemes. This makes it difficult to quickly integrate new technologies and accommodate new services that may require computation within the network, such as firewalls, web proxies, in the traditional passive network infrastructure.

2.2 Active Network architecture

There are several architectural approaches to the realisation of active networks depending on the location of the executable codes. Our architecture provides two distinct flavours of activeness; the Active Router and the Active Server (AS). These differ in amount of control and flexibility they offer, the Active Routers have tight restrictions since an incorrect operation can have a serious impact to many users, so proxylets deployed here must be from a trusted source such as the owner or operator of the router. While Active Servers are more

flexible, undesirable operations have limited consequences, so control is delegated to general users. Thus third party executable codes are deployed on the AS in our approach, as shown in figure 1, [1]. Allowing third party executables to be deployed at the server level means an incorrect operation will affect limited numbers of users, but it does not remove the security threat imposed by the active network open access approach.

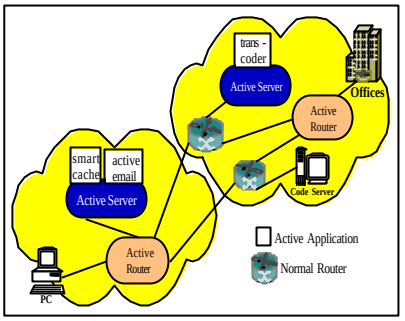

Figure 1: Active network architecture

2.3 Android security architecture

To address this security problem the high-level security architecture in figure 2 was defined. Here security falls into three clear categorises; User security, this addresses user security requirements. Platform security, this focuses on the security of the actual active server, which is provided by the management elements and policies in action on the server. And finally management security, this focuses on the security of the management information flow and directories [2].

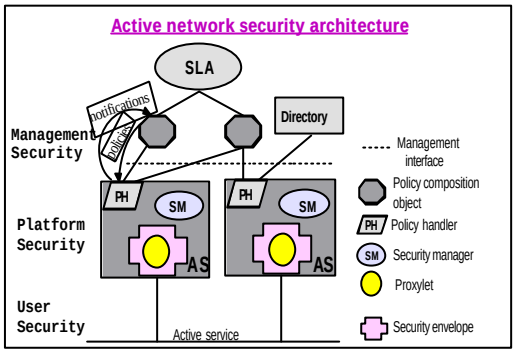

Figure 2: Active network security architecture

In our security architecture, platform security encompasses three major security measures; Proxylet deplorer authentication, Proxylet authentication and Run-time proxylet resource monitoring. This paper focuses on the Runtime proxylet security, a measure to ensure system integrity is maintained in case the first two-security measures are bypassed. It involves the policing of proxylet behaviours or actives on the AS in order to ensure that a proxylet stays within its behavioural security envelope. Thus it acts as an IDS that uses behaviour-based anomaly detection to provide a third line of defence.

2.4 Intrusion Detection

An Intrusion Detection System (IDS) polices or inspects the activities on the system for suspicious behaviour or patterns that may indicate system attack or misuse. There are two main categories of intrusion detection techniques; Anomaly detection and Misuse detection. The former analyses the information gathered and compares it to a defined baseline of what is seen as “normal” service behaviour, so it has the ability to learn how to detect attacks that are currently unknown. Misuse detection is based on signatures for known attacks, so it’s only as good as the database of attack signatures that it uses for comparison.

3 IDS model

Our IDS security model is made up of three major functional components as shown in figure 3; the monitoring system, which monitors the service element resource utilisation. Anomaly detection system, this is the brain power of our IDS, it uses the measured system information such as CPU and memory utilisation to infer the chance of proxylet being “bad”. Finally the Immune system, this provides the adaptive capabilities to the security system and allows policy distribution [2].

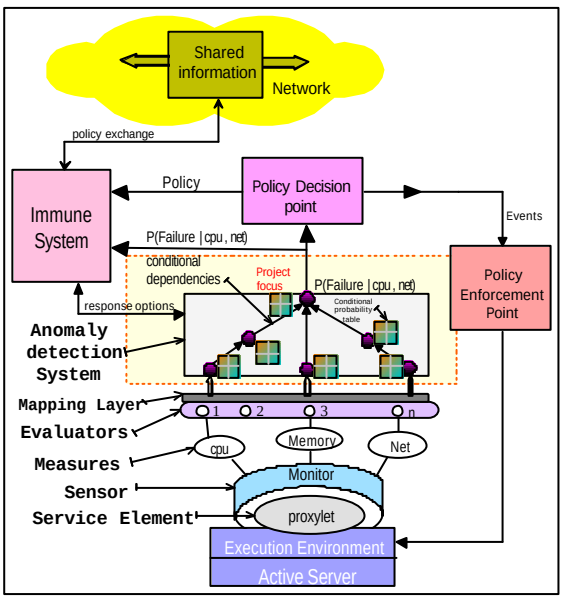

Figure 3: Schematic diagram for our IDS model

3.1 Anomaly Detection model

Our implementation of the anomaly detection system involves the application of Bayesian technology to the measured system attributes. Bayesian technology is well established in the medical diagnosis field and is almost a standard tool for modelling complex problems involving uncertainty in artificial intelligence community. The most common example of Bayesian application is Microsoft Office Assistant application.

A Bayesian Network (BN) is a graphical model that encodes probabilistic relationships among dependent set of variables. Its main distinguishing feature to classical statistical inference approaches is the use of subjective or personal beliefs (prior probabilities) directly into the analysis [3, 4].

A BN will be incorporated into our IDS as anomaly detector. This approach is based on the assumption that service elements (proxylets) have resource utilisation regularities that can be detected and identified as belonging to a particular service element. This allows us to build a BN, which provides some probabilistic reasoning about the state ("bad" or not) of a running proxylet from the current observed system profiles. By bad proxylet we mean a proxylet whose profile is outside the expected normal profile for that particular class of service elements.

3.2 Development of Anomaly detection system Model

The structural arrangement of our model follows from a much simpler Bayes model that has an impress record in medical diagnosis problems, known as naïve Bayes model. To see how our problem can be tackled, we began by developing a simple proxylet classifier model shown in figure 4. Suppose we have "bad proxylet" as an unobserved class denoted by the variable B, with the binary states (True or False). And a random variable $\varepsilon \subseteq \{e_1 = \text{CPU utilisation(C)}, e_2 = \text{memory utilisation (M)}\}$, which represents observed system attributes. For simplicity let us keep *e* discrete with states true or false by assuming if CPU utilisation is above 90%, or memory utilisation is exceeds 70% the running proxylet is classified as a bad one, (B=T). To complete our model construction process each variable (node) was assigned a probability of observing a certain state, resulting into the Conditional Probability Distributions (CPDs) shown in figure 4.

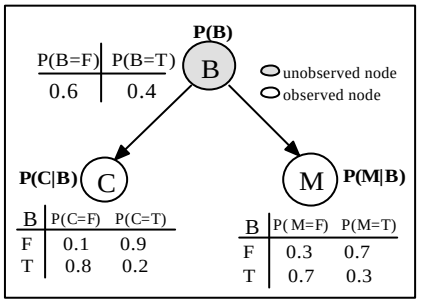

Figure 4: A naïve Bayesian proxylet classifier model.

This model relies on two simplifying assumptions; the observed attributes (C,M) are assumed to be conditionally independent given the class B, and there no hidden attributes influence the prediction process, hence the name naïve [5, 6]. The CPDs shown form the basis for computation, so one simply uses the Bayes rule[1] to compute the probability of a proxylet being bad (B=T) given the evidence $\varepsilon \subseteq \{e1 = \text{T}, e_2 = \text{T}\}$, that is to say probability $P(\text{B=T}|\ \text{C=T, M=T}) = P(\text{B=T}|\varepsilon)$;

$$P(\text{B} = \text{T} \mid \text{e}) = P(\text{B} = \text{T}) \times \frac{P(\text{e} \mid \text{B} = \text{T})}{P(\text{e})} = \frac{\prod_{e \subseteq \text{e}} P(e \mid \text{B} = \text{T}) P(\text{B} = \text{T})}{\sum_{b=T}^{F} \prod_{e \subseteq \text{e}} P(e \mid \text{B} = b) P(\text{B} = b)} = \frac{0.024}{0.402} = 0.0597$$

The above model was implemented in Matlab and various inferences were computed by processing the encoded knowledge using the Junction tree algorithm offered by an open source Matlab package, Bayes Net Toolbox (BNT) [7].

Since the system we are trying model is far more complex than can be represented solely by discrete parameters, we extended this discrete model to hold both discrete and continuous variables. The unobservable variable B remaining discrete since we want our anomaly detector to also act as an aggregator of all the observed attributes. For the continuous attributes it is common practice to assume that within each state of class B, the observed attributes follows a normal (Gaussian) distribution. The condition distribution of each observed attribute can be defined through its mean and standard deviation given the state of B. From such estimates the conditional probability of an observed value (*e*) can be computed as shown below.

$$P(\text{e} = e \mid \text{B} = b) = g(e; m_b, s_b), \quad \text{where} \quad g(e; m_b, s_b) = \frac{1}{\sqrt{2ps}} \text{e}^{-\frac{(e-m)^2}{2s^2}} \qquad (1)$$

The Gaussian-valued nodes were created as shown in figure 5, where $N(\mu_b, \sigma_b)$ defines a normally distributed continuous node with a mean μ_b and standard deviation σ_b, which allows us to encode our expert knowledge about the system.

To test this model a random sample of data was generated and presented to the model as the observed CPU utilisation. The results of the model are shown in figure 6a and 6b. Figure 6a displays the output corresponding to the probability of a running proxylet being bad (B=T), given the memory consumption (e2) of 60% and 80% and CPU utilisation profile shown by the blue solid line.

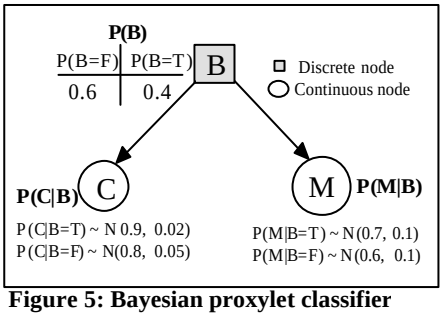

Figure 5: Bayesian proxylet classifier model with Gaussian estimated CPDs.

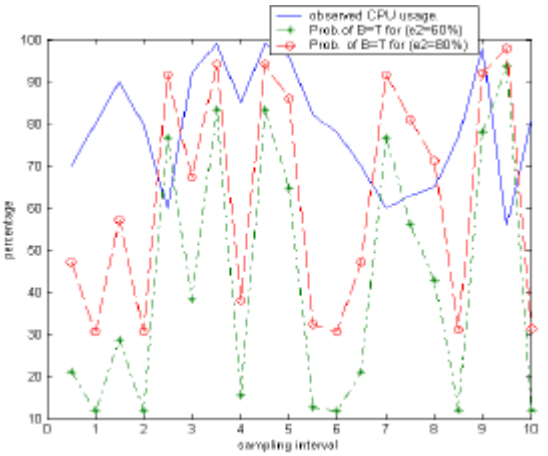

Figure 6a: Plot of the probability of B=T for the given CPU profile (solid line) and rising memory usage.

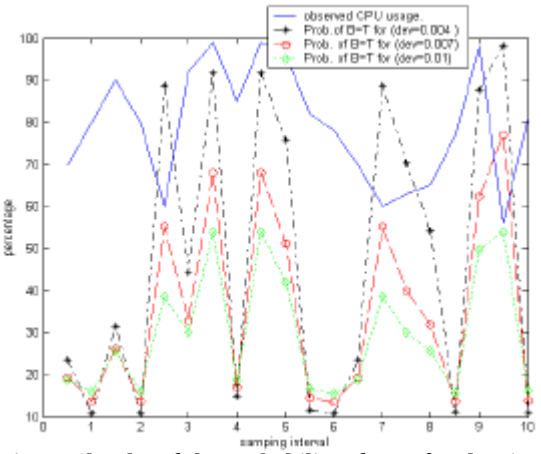

Figure 6b: Plot of the probability of B=T for the given CPU profile (solid line) and model of different s_c values.

In this figure (6a) we can see that when the CPU utilisation is between 70 and 90 percent, the probability of B =T is very low since this is classified as the utilisation for a normal proxylet, and outside this range the probability is much higher. It also shows that the probability of B =T increases as an effect of observing memory consumption higher than expected for the normal proxylet. The result in figure 7b shows the effect of changing the CPU variance in the model. We can see that increasing the variance reduces the variance of the model output, a feature when modelled could prove useful in the attempt to reduce the number false alarms. From these results we can see that, by fine tuning this model and using better CPD estimates one can effectively classify service elements from the observed resource utilisation profiles.

4 Conclusions and future work

We have presented our IDS model which aims at autonomously assessing the state of the running service elements using a probabilistic reasoning model as the anomaly detector. As this is our first attempt to develop a reasoning system, the Bayesian approach has given interesting results in a short time. We believe it provides a promising step towards our challenge of developing an adaptive runtime active platform security system.

To realise the full potential of this approach more work still need to be done, so for the near future we intend to extend this work in two ways. Firstly we intend to increase the number of nodes allowing more sophisticated statistical indicators such as CPU variance to be modelled. This will give a more detailed analysis and greater sensitivity to the reasoning system. We also intend to apply parameter learning techniques to improve our CPD estimates and make the anomaly detector more accurate.

Acknowledgements

Many thanks to Dr. Lionel Sacks and Temitope Olukemi for their support, I also wish to thank the UCL Advanced Communications Systems Engineering (ACSE) Group for creating a good working environment.

References

[1] Ian W Marshal, ("An architecture for application layer active networking", *IEE*, London, 2000.

[2] Ognjen Prnjat, et. al., "Integrity and Security of the Application Level Active Networks", UCL, London, 2000.

[3] Bruce W. Morgan, "An Introduction to Bayesian Statistical Decision Processes, New Jersey, 1968.

[4] David Heckerman, "A tutorial on learning with Bayesian Network", *Microsoft Research*, Redmond, 1996.

[5] Robert G. Cowell A. Philip Dawid, "Probabilistic Network and Expert Systems", *Springer*, New York, 1999.

[6] George H. John, et.al, "Estimating Continuous Distributions in Bayesian Classifiers", *Morgan Kaufmann publishers*, San Mateo, 1995.

[7] Kevin. P. Murphy, "Bayes Net Toolbox", www.cs.berkeley.edu/~murphyk/Bayes/bnt.html, 2002.