An Agent-based Bayesian Forecasting Model for Enhanced Network Security

J. PIKOULAS, W.J. BUCHANAN, *Napier University, Edinburgh, UK.*
M. MANNION, *Glasgow Caledonian University, Glasgow, UK.*
K. TRIANTAFYLLOPOULOS, *University of Warwick, UK.*

Abstract

Security has become a major issue in many organisations, but most systems still rely on operating systems, and a user ID and password system to provide user authentication and validation. They also tend to be centralized in their approach which makes them open to an attack. This paper presents a distributed approach to network security using agents, and presents a novel application of the Bayesian forecasting technique to predict user actions. The Bayesian method has been used in the past on weather forecasting and has been expanded so that it can be used to provide enhanced network security by trying to predict user actions. For this a system can determine if a user is acting unpredictably or has changed their normal working pattern. Results are also given which show that the new model can predict user actions, and a set of experiments are proposed for further exploitation of the method.

1. Introduction

Computer security is a major concern for organizations. Whilst security violations can be caused by external users (*hackers*), Carter and Catz [1] have shown that the primary threat comes from individuals inside an organisation. Hence much more emphasis has to be placed on internal security mechanisms.

External network attacks can be categorised [4] into IP spoofing attacks [5], Packet-sniffing [6], sequence number prediction attacks and trust-access attacks. Categories of internal attack include Passwords attacks [7], session hijacking attacks, shared library attacks, social engineering attacks, and technological vulnerability attack.

Computer network security programs can be categorised as follows [3]:

- **Security enhancement software**. This enhances or replaces an operating system's built-in security software (for example, Mangle It, Passwd+ and Shadow).
- **Authentication and encryption software**. This encrypts and decrypts computer files (for example, Kerberos, MD5, RIPEM, and TIS Firewall Toolkit).
- **Security monitoring software monitor**. This monitors different operations of a computer network and outputs the results to system administrators (for example, Abacus Sentry, COPS, Tripwire and Tiger).
- **Network monitoring software**. This monitors user's behaviour or monitors incoming or outgoing traffic (for example, Argus, Arpwatch and ISS).
- **Firewall software and hardware**. This runs on the Internet/intranet entrance to a computer network, and checks all incoming network traffic for its contents at the network and transport layers of the OSI model. At the network layer, typically the Internet Protocol (IP) addresses are filtered for their source and/or destination, and at the transport layer, the TCP ports and monitored (thus FTP and TELNET traffic could be blocked for incoming data traffic, but SMTP (electronic mail) could be allowed).

These methods are generally **centralised** applications with no real time response and have no mechanism to foresee future user events. These methods also have a central focal point for security (typically a main server), which could itself become the focus of an attack (such as a denial-of-service attack, where the server is bombarded with hoax requests, which eventually reduces its quality of service to its clients).

The method involved in this research is **distributed**, and does not depend on a central point of failure. It also gathers user behavioural information and it makes a prediction on what the user might do in the future. This paper presents a distributed approach to network security using agents, and presents a novel application of the Bayesian forecasting technique to predict user actions. The Bayesian method has, in the past, been used for weather forecasting and has been expanded so that it can be used to provide enhanced network security by trying to predict user actions. For this a system can determine if a user is acting unpredictably or has changed their normal working pattern. Results are also given which show that the new model can predict user actions, and a set of experiments are proposed for further exploitation of the method.

In choosing a computer network security solutions, the dominant issues are: cost; the desired level of security: and the characteristics of the existing operating system envi-

ronment. Three mechanisms for illegal behaviour detection are commonly used in computer network security programs [8], and can be applied to all five categories of computer security program.

Statistical Anomaly Detection

Statistical anomaly detection systems analyse audit-log data to detect abnormal behaviour [9]. A profile of expected online behaviour for a normal user is predefined and derived from how an organisation expects a user to behave and from a system administrator's experience of the way a user is expected to use the resources of a system. Typically, the audit logs are analysed and processed for statistical patterns of events for typical operations for which to determine usage patterns. These patterns are compared to the user's profile.

The system then warns the administrator that there has been a possible intrusion when a profile is different to a usage pattern. The major drawback with this technique is that it cannot predict extreme changes in user behaviours, as changes in a user's behaviour normally identify a security breach.

Rule Based Detection

Rule-based detection systems use a set of generalised rules that define abnormal behaviour (10,12,13). These rules are formed by analysing previous different patterns of attack, by different people. The drawback of this system is that the basic rules are predefined by system administrators, and cannot detect any new attack techniques. If a user exhibits behaviour that is not prescribed by the existing rules, the user can harm the system without being detected.

Hybrid Detection

Hybrid detection systems are a combination of statistical anomaly detection and rule-based detection systems. These, typically, use rules to detect known methods of intrusion and statistical based methods to detect new methods of intrusion.

CMDS (Computer Misuse Detection System) [14] is a security-monitoring package that provides a method to watch for intrusions, such as bad logins or file modifications. It also monitors for the difficult detection problems such as socially engineered passwords, trusted user file browsing and data theft that might indicate industrial espionage. CMDS supports a wide variety of operating systems and application programs. The drawback of this system is that it uses statistical analysis to make additional rules for the system. This is a drawback, as it can only detect attack patterns that have been used in the past and identified as attack patterns, or predefined by the system operators. It also generates long reports and graphs of the system performance that require to be interpreted by a security expert.

4 Bayesian Intrusion Detection System

We used a Bayesian multivariate statistical model because our problem is a linear multivariate problem and it is simpler faster and more accurate to use a linear model than a non-linear model like neural networks [24]. In order to test this an intelligent agent security enhancement software system was constructed, in which a core software agent resides on one server in a Windows NT network system and user end software agents reside in each user workstation. The software for each type of agent was written in SUN Java JDK Version 1.2 on a Microsoft Windows NT Version 4 environment running over a 10/100 Mbps network. There is one server and 10 clients. Figure 1 shows a core agent communicating with many user agents. A communication thread is a unique process that the core agent creates to transmit data to the user end agent in response to message transmitted from the user end agent. Unique processes enable the core agent to communicate with each user agent effectively and efficiently thereby enabling a fast response to network monitoring. Once the core agent has responded to a user agent, the process is killed.

The system uses a hybrid detection technique, where invalid behaviour is determined by comparing a user's current behaviour with their typical behaviour and by comparing their current behaviour with a set of general rules governing valid behaviour formed by systems administrators. Typical behaviour is contained in a user historical profile.

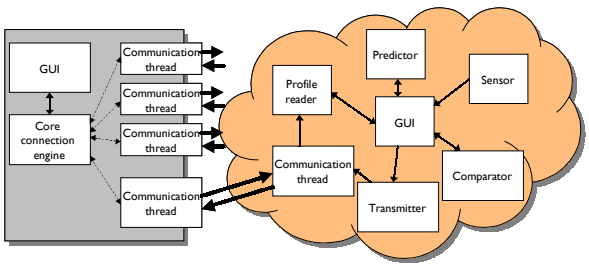

Figure 1: Agent Environment Topology

The user agent software has four components:

- **A sensor**. The sensor monitors the various software applications (such as a word processor or a spreadsheet) that are currently being run by the user on that workstation. When a user logs-in the sensor polls the user's activity every five seconds and records the user's identifier and each application's name and process identifier.
- **A transmitter**. After the first polling by the sensor, the transmitter sends this information to the core agent. The core agent then responds by sending a user

historical profile. With an audit-log file for a period of one month, we observed that the size of an average user profile was between 400 KB and 600 KB, with a download time of between three and five seconds.

- **A profile reader**. The profile-reader reads the user's historical profile.
- **A comparator**. This compares the user's historical profile with the information read by the sensor. If the current behaviour profile does not fall within the accepted behaviour pattern defined by the user historical profile, the comparator provides the transmitter with the following information: user identifier, invalid behaviour type and corresponding invalid behaviour type data.. This is then sent to the core agent.

When invalid behaviour occurs, several courses of action are available, such as:

1. Warning message to the system administrator or end user.
2. Kill the specific application that has caused invalid behaviour.
3. Prevent the end user from running any further applications.

Cases 2 and 3 can be achieved locally at the client workstation, and in Case 1, the user agent informs the core agent and the core agent informs the systems administrator. The user agent terminates when a user logs off.

Figure 2 shows the complete model for the forecasting system, where a core agent reads the user profile, which is then received by the user agent. The user agent then predicts the usage against the forecast. Eventually when the user logs off the user profile is updated and sent back to the core agent.

In the traditional method of forecasting, a user event would be averaged over long time intervals (in Figure 3).

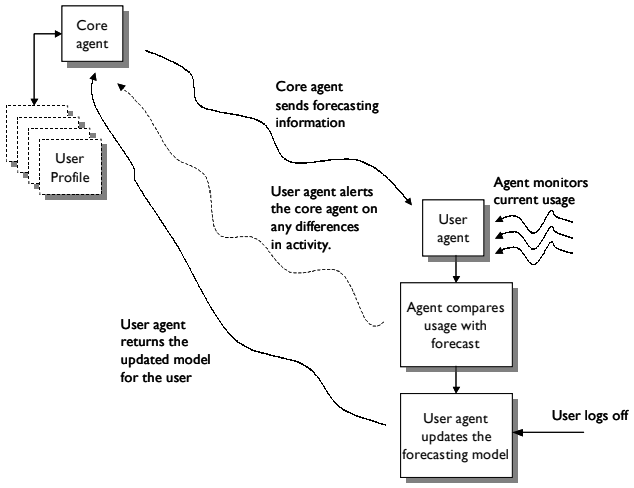

Figure 2: Agent forecasting model

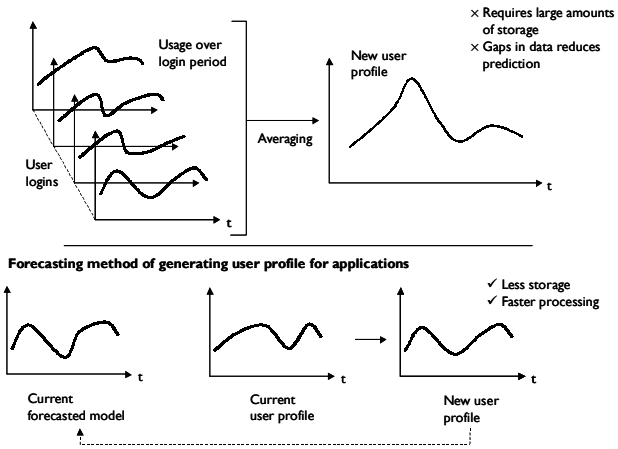

Figure 3: Traditional method of generating user profile for applications

5 Prediction Model

When our intrusion detection system is installed, the prediction part monitors the user behaviour for 15 times. After that, it evaluates itself for five times. After this it is ready to make an accurate prediction. Our model has three stages of operation. The stages are:

1. **Observation stage.** In this stage the model is monitoring the user and records its behaviour.
2. **Evaluation stage.** In this stage the model makes a prediction and also monitors the user actual movements and calculates the result. This stage is critical, because the model modifies itself according to the environment that it operates in.
3. **One-step prediction.** In this stage the model makes a single step prediction. For example, assume that the user is logged in for 15 times and the model is configured, and it is ready to start predicting user moves. Instead of making a five or ten step prediction, like other mathematical models, our model makes a prediction for the next step. When the user logs in and out of our model, it takes the actual behaviour of the user, compares it with the one step prediction that it has performed before and calculates the error. So the next time a prediction is made for this user it will include also the data of the last user behaviour. With this procedure we maximise the accuracy of the prediction system.

The proposed forecasting method improves this by requiring much less memory storage. Figure 4 shows a generic model for the predicting using parameters for a given window size (n), time units and prediction number (z).

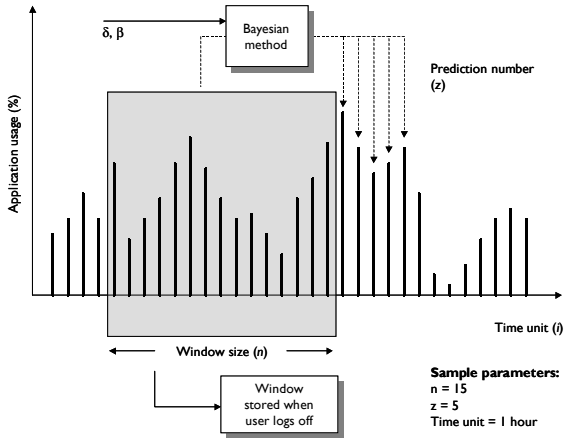

Figure 4: Forecasting calculation

The general multivariate model (DLM) is given by the next equations:

$$Y_t = F_t^{'}\theta_t + v_t, \qquad v_t \sim N[0,\Sigma] \qquad (1)$$

$$\theta_t = G_t\theta_{t-1} + \omega_t, \qquad \omega_t \sim N[0,W_t] \qquad (2)$$

We use multivariate models because we want to incorporate and forecast several variables simultaneously. Again note that the fact that the parameters θ_t change both deterministically (through t) and stochastically (through the variance W_t), and thus make the model dynamic. Also standard ARIMA (Auto-Regressive Integrated Moving Average) models are a special and restrictive case of the above model, when you set $F_t = F$, $G_t = G$ and $W_t = W$ (all these three components are constant over time). This is restrictive since all these components are likely to change over time because e.g. (1) changes over time and there are other external sources of variation (such as extra subjective information about a variable). Moreover, equation (2) is not observable. This means that we never are going to see any evolution or trend in a diagram or a graph. This is a hidden model that cannot assume W_t to be constant over time. There is another large problem that we cannot ignore in multivariate models. The variance matrix Σ will not be known. Often, in standard time series, it is assumed known and they easily jump to another problem. However, in practice, this is extremely difficult to set it as a known matrix. It is very difficult to propose what variance to use to a system where 20 applications are considered and only 20 or 30 vectors are collected as data.

So for all these reasons we need to consider the dynamic models. Also, the system could provide forecasting as much ahead as we like, proving very accurate according to the results. For this purpose we used a Bayesian framework, which virtually means that at time t we will have some kind of knowledge, that is a subjective belief, expressed in terms of a distribution. This is the prior distribution of $(\theta_t \mid D_{t-1})$ at time t. In other words, it is what we know before Y_t becomes available. Once this happens, we revise this prior belief, using the likelihood function, to find the posterior distribution $(\theta_t \mid D_t)$ or revised, which is better and more accurate. Then according to simple calculations, we find the prior of time $t-1$ and we calculate the posterior at $t+1$, only when information of the data Y_{t+1} comes in to the system (e.g. in our case is the real behaviour of the user). The model used becomes:

- **Autoregressive moving average model**. The general model introduced by Box and Jenkins (1976) includes autoregressive as well as moving average parameters, and explicitly includes differencing in the formulation of the model. Specifically, the three types of parameters in the model are: the autoregressive parameters (p), the number of differencing passes (d), and moving average parameters (q). In the notation introduced by Box and Jenkins, models are summarized as ARIMA (p, d and q); so, for example, a model described as (0, 1, 2) means that it contains 0 (zero) autoregressive (p) parameters and 2 moving average (q) parameters which were computed for the series after it was differenced once.
- **Identification**. As mentioned earlier, the input series for ARIMA needs to be stationary, that is, it should have a constant mean, variance, and autocorrelation through time. Therefore, usually the series first needs to be differenced until it is stationary (this also often requires log transforming the data to stabilize the variance). The number of times the series needs to be differenced to achieve stationary is reflected in the d parameter (see the previous paragraph). In order to determine the necessary level of differencing, one should examine the plot of the data and autocorrelogram. Significant changes in level (strong upward or downward changes) usually require first-order non-seasonal (lag=1) differencing; strong changes of slope usually require second order non-seasonal differencing. Seasonal patterns require respective seasonal differencing (see below). If the estimated autocorrelation coefficients decline slowly at longer lags, first-order differencing is usually needed. However, one should keep in mind that some time series may require little or no differencing, and that over differenced series produce less stable coefficient estimates.
  At this stage we also need to decide how many autoregressive (p) and moving average (q) parameters are necessary to yield an effective, but still efficient, model of the process (that is with the fewest parameters and greatest number of degrees of freedom among

all models that fit the data). In practice, the values of the *p* or *q* parameters are rarely greater than two (see below for more specific recommendations).

- **Estimation and Forecasting**. At the next step (estimation), the parameters are estimated (using function minimization procedures), so that the sum of squared residuals is minimised. The estimates of the parameters are used in the last stage (forecasting) to calculate new values of the series (beyond those included in the input data set) and confidence intervals for those predicted values. The estimation process is performed on transformed (differenced) data; before the forecasts are generated, the series needs to be integrated so that the forecasts are expressed in values compatible with the input data. This automatic integration feature is represented by the letter I in the name of the ARIMA methodology.

In addition to the standard autoregressive and moving average parameters, ARIMA models may also include a constant, as described above. The interpretation of a statistically significant constant depends on the model that is fit. Specifically:

> if there are no autoregressive parameters in the model, then the expected value of the constant is μ the mean of the series;
> if there are autoregressive parameters in the series, then the constant represents the intercept.

If the series is differenced, the constant represents the mean or intercept of the differenced series; For example, if the series is differenced once, and there are no autoregressive parameters in the model, the constant represents the mean of the differenced series, and therefore the linear trend slope of the un-differenced series.

ARIMA models are similar to our model. They use the existing data to calculate the parameters of the model. But if, for example, some external information is available. For example, we may know that it is the *x* user and although he does not have an illegal user profile, it is very probable that at a specific point of time he will perform a huge invasion to an important application. ARIMA will try to change the parameters to adjust the model, but even in this case, it is doubtful how well the model will do in all the applications. With our DLM it is not a problem. Simply we add to the prior information we have, the external information. This is named expert intervention, and the revised posterior takes into account the new knowledge. Our system is not assumed perfect when the model is fitted, and we let information, no matter what its sort, to make us learn and improve the system.

Now our model is slightly different than the one we use for illustration purposes. We find recurrence relationships, which are more natural to overall long formulae that ARIMA works out. We note that because ARIMA is quite complicated, many practitioners end up to a simple, very simple subclass of ARIMA model, not even sometimes stating the assumptions. This produces results that sometimes do not correspond to the real application. The only difficulty with the DLMs is the specification of the initial values, such that the algorithm may be put into practice. In general this requires to be solved by the experience of the individual practitioner.

In our case, we have to specify the following: $m_0, C_0, S_0, n_0, \beta, \delta, F_t * m_0$ is the mean of $(\theta_0 \mid D_0)$ and C_0 its variance. The choices made are:

- $m_0 = 0$. This is set when we expect that the prior distribution $(\theta_0 \mid D_0)$ (the distribution of the parameter Θ at time t given D_0 - any initial information which is explicitly known) will not give any drift to Y_1. The fact that we expect this to happen, but we are not sure, so there is here an uncertainty, which is expressed by the variance C_0. It is natural and common policy to assume $C_0 = I$, the identity matrix. But care must be taken when we are very uncertain about our choice we MUST increase the diagonal elements of C_t. Of course, this affects all the following results somehow, but the approach is more realistic. In general we will have more data vectors than 15, or 20 (our case), hence initial values will dominate the actual estimates in a decreasing rate.
- $C_0 = I$. This is motivated by our belief that m_0 is not important to the following values of Y_t, t=1, ... S_0 is typically, almost always set to I and it has not got any special meaning. The only one we can find is that it is chosen such that according to the formula that we have to calculate S_t, S_0 must lead to acceptable results (symmetric matrices). The n_0 can be set to 0 (a case which implies $n_1 = 1$, without great loss) or $n_0 = 1$ (a case which implies $n_1 = \beta + 1$). The choice of n_0 is not crucial since there is theorem that states that S_t converges to Σ as t goes to infinity and it does not depend on n_0. But it must take small values.
- δ . The δ choice is discussed with details in Ameen and Harrison (1982a) where it is shown that it must be $0.85 < \delta < 1$ and quite high. Thus we have set it 0.95.
- β . The β is a discount factor as well. In this document we state that it has to be smaller than δ, as, in general, S_t is not so much influenced by the data as it is m_t. Note that δ is in A_t and so it influences m_t.

The components are defined as:

m_0 : The mean of the influence of Θ_1, Y_1 from D_0, our initial info.

C_0 : Dispersion of the above influence.

S_0 : No meaning, and is an auxiliary quantity for S_t.

n_0 : No meaning, and is an auxiliary quantity for n_t.

β : Factor of the influence of the data to the estimate S_t.

δ : Factor of the influence of the data to the estimate m_t.

F_t : A basic quantity that expresses the linearity of the model and gives different trends to the several values of Y_t, both for time series analysis (what has happened in the past) and forecasting (what will happen in the future).

Finally, we make clear that when we say *factor* in the above explanation we do not mean any percentage or whatever. Factor means discount factor, which means that the estimates of m_t and S_t are discounted somehow and in different rate, since both are influenced by data.

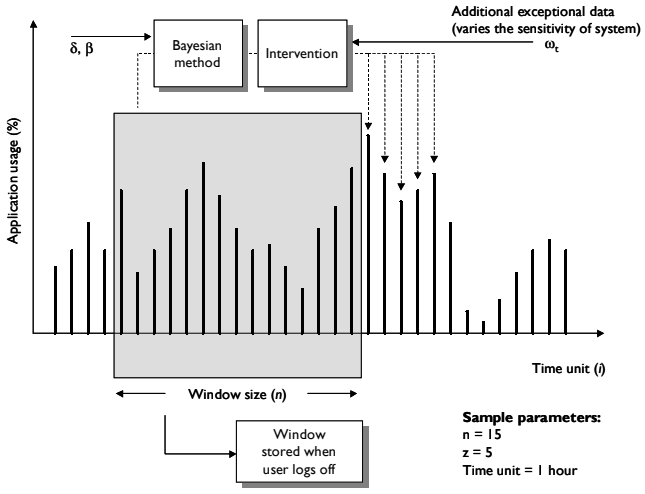

Figure 5: Forecasting calculation with intervention

Intervention

Intervention is a mechanism for improving the prediction accuracy. It is used when there is additional information about the future behaviour of the system, and can be added to the model prior the prediction. For example if there is some users that are keen on using illegal software or there are new users that there is not enough information about their behaviour, by applying the intervention mechanism, we increase the accuracy of the model and can make more accurate predictions (Figure 5).

In our model we can observe this by looking at Figure and Figure . In these we can observe that our model prediction is very close to the actual users behaviour for the application number one at the specific time *t=19*. We achieved this accuracy by applying the intervention technique. We can also observe that the ARIMA model did not make any prediction for this particular user behaviour.

Results and experiments

The first set of experiments are made in order to test our security environment to the extent that it works and to get some results from our proposed statistical model and compare it with other statistical models.

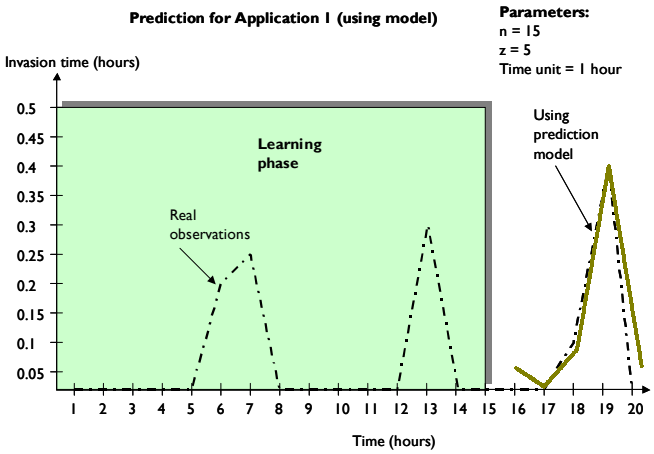

Figure 6: The Real Observations of the Model

Our environment is vastly improved with the use of the prediction mechanism. Our system is not using the real time data that its agents gather only for real time detection. Our addition of the prediction model in our environment, increases its functionality and its usability to the maximum.

Figure 8 shows one user that logged on to the system 20 times and had one hour sessions each time. We monitored all their moves and all the applications that he used. In our prediction model we had only three applications to predict. The intervals are from 0 to 1 and they denote an hour. So, for example, 0.3 means that the user used this program for 0.3 of the hour (18 minutes), in this specific hour of the system usage.

We used our prediction mechanism for the last five observations. As we can observe from the results, if we compare the graphs in Figure 6, which are the real observations for the three applications, and the graphs in Figure 7, we can see that the two figures are almost identical. We cannot say the same if we compare the real readings from the results of the ARIMA model. We can see that they are

less precise with the actual readings and they fail to predict the action of the user in application 1 at the time interval nineteen, in comparison with our model that predicted it with a very close figure.

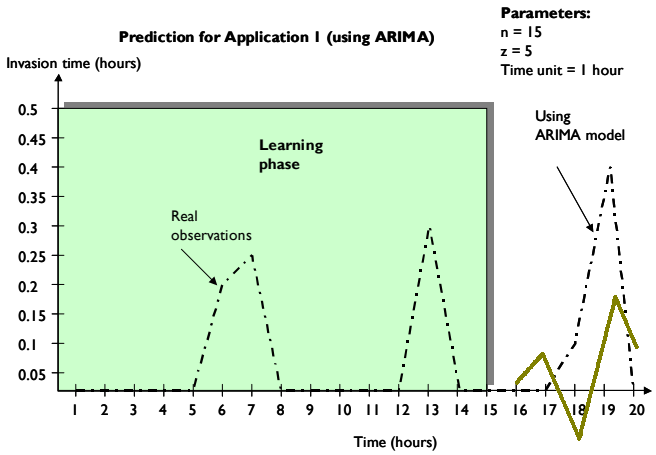

Figure 7: The proposed Model 5 step prediction

Evaluation

Our proposed model is a multivariate linear model that is a simple and fast adoptive model. It requires far less preparation than other models like, for example, the neural net weights that you have to decide before you build your neural net model. Our proposed environment reacted as expected to all the tests that were applied. The monitoring of the user behaviour was successful and the overhead on the system resources was minimum.

There was a 1 to 2% increase on the CPU usage, when the user agent was monitoring the user moves, and the prediction task only took two seconds to complete with the three applications and for a fully operational system with 20-25 applications, we estimate that it will take no more than five seconds.

Our proposed environment is collecting information about the user every five seconds. The prediction procedure is taking place at the end of each hour of a user's use of the system. If the user log off before one hour completes, the calculation of the prediction takes place whenever the user will finish log on to the system again and completes one hour.

Another difference of our model is that the statistical models that are in use now, work inside acceptable parameters, only because they make too may assumptions about the initial parameters, a factor that we believe makes them give results that does not represent actual situations.

Future Work

The experiments that are conducted up to now were setup to verify that the environment works and to show that our statistical proposed model gives better results than the models that are widely used up to now. In the next stage, we are planning to expand the number our experiments, and also the number of the applications that we use, and the number of users involved. We also planning to fully exercise our model by instructing users to have some extreme behaviour for sort periods and normal behaviour for long periods so we want to see if the model detects and predicts extreme user behaviour (Figure 8).

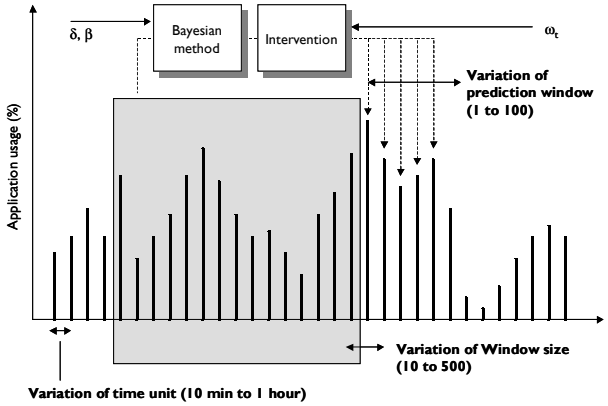

Figure 8: Experimental setup

8 References

[1] Carter and Catz, *Computer Crime: an emerging challenge for law enforcement*, FBI Law Enforcement Bulleting, pp 1-8, December 1996.

[2] Roger Blake, *Hackers in The Mist,* Northwestern University, December 2, 1994.

[3] National Institutes of Health. *Center for Information Technology,* http://www.alw.nih.gov/Security/securityprog.html#commercial, October 1998.

[4] W.J. Buchanan. Handbook of Data Communications and Networks, Kluwer, 1998.

[5] SamsNet, *A Hacker's Guide to Protecting Your Internet Site and Network*, URL:http://mx.nsu.ru/Max_Security/ch28/ch28.htm

[6] NetworkICE Corporation, *Packet Sniffing*, http: //www.networkice.com/ advice/ Underground/ Hacking/ Methods/ Technical/ Packet_sniffing/default.htm

[7] Alan Ramsbottom, *FAQ: NT Cryptographic Password Attacks & Defences*, 1997, http: //www.omikron.de/ ~ecr/ nthack/ samfaq.htm.

[8] Chris Herringshaw, *Detecting Attacks on Networks*, IEEE Computer Magazine, pp 16–17, Dec. 1997.

[9] Debra Anderson, *Detecting Unusual Program Behavior Using the NIDES Statistical Component*, IDS Report SRI Project 2596, Contract Number 910097C (Trusted Information Systems) under F30602-91-C-0067 (Rome Labs), 1995.

[10] T. Lunt, H. Javitz, A. Valdes, *et al. A Real-Time Intrusion Detection Expert System (IDES)*, SRI Project 6784,

Feb. 1992. SRI International Technical Report.

[11] J Pikoulas and K Triantafyllopoulos, *Bayesian Multivariate Regression for Predicting User Behaviour in a Software Agent Computer Security System*", 20th International Symposium on Forecasting, Lisbon, Portugal, June 21, 2000.

[12] Sandeep Kumar and Gene Spafford, *A Pattern Matching model for Misuse Intrusion Detection*, Proceedings of the 17th National Computer Security Conference, Oct. 1994.

[13] Mark Crosbie and Gene Spafford, *Active Defence of a Computer System using Autonomous Agents*, COAST Group, Dept. of Computer Science, Prudue University, Technical Report (95-008),2–3, Feb 1995.

[14] *The Computer Misuse Detection System*, http://www.cmds.net/, 1998.

[15] Pikoulas J, Mannion M and Buchanan W, *Software Agents and Computer Network Security*, the 7th IEEE International Conference on the Engineering of Computer Based Systems, pp 211 – 217, Apr. 2000.

[16] Jean O. Dickey, Christian L. Keppenne, and Steven L. Marcus, *FORECASTING REGIONAL CLIMATE CHANGE WITH ADVANCED STATISTICAL METHODS*, Jet Propulsion Laboratory, California Institute of Technology, Pasadena.

[17] Professor Hossein Arsham, *Statistical Data Analysis: Prove it with Data*, University of Baltimore, http://ubmail.ubalt.edu/~harsham/stat-data/opre330.htm.

[18] Carlin B. and T. Louis, *Bayes and Empirical Bayes Methods for Data Analysi*s, Chapman and Hall, 1996.

[19] Stanford University, *GENSCAN: A Powerful tool for Gene Prediction*, Vol. 8, N. 1, 1999.

[20] Steven L. Salzberg , Arthur L. Delcher , Simon Kasif and Owen White, *Microbial gene identification using interpolated Markov models,* pp. 544–548, Nucleic Acids Research, 1998, Vol. 26, No. 2, 1998 Oxford University Press.

[21] *The Great Lakes Forecasting System*, The Ohio State University (OSU) and the National Oceanic and Atmospheric Administration (NOAA) Great Lakes Environmental Research Laboratory (GLERL), http://superior.eng.ohio-state.edu/main/noframes/about.html

[22] Sandia National Laboratories, *A Smart, Agent based simulation model*, http://www-aspen.cs.sandia.gov/, Feb. 2000.

[23] J.R.M. Ameen and P.J. Harrison, *Normal discount Bayesian models*, Journal of Bayesian Statistics, 1985.

[24] Georges A. Darbellay and Marek Slama, *Forecasting the sort term demand for electricity. Do neural networks stand a better chance?*, International Journal of Forecasting, pp. 71-83, 2000