# The Application Of Neural Networks To UNIX Computer Security

Kymie Tan
The University of Melbourne
Department of Computer Science
Parkville 3052, Australia

**ABSTRACT**
Computer security can be divided into two distinct areas, preventive security and the detection of security violations. Of the two, a greater degree of research and emphasis has been applied to prevention, while detection has been relatively overlooked. This is a costly oversight as preventive measures are never infallible. To date the detection of intruder violation on computer systems is a field heavily dominated by expert systems. However, the major drawbacks attributed to these systems including their heavy demand on system resources and their poor handling of the dynamic nature of user behaviour [10, 11], have made their use infeasible. In practice, the effectiveness of intruder detection is heavily reliant upon the skills of the presiding system administrators and their knowledge of the behaviour of their users. The present study approaches the problem from a pattern recognition point of view, where a neural network is used to capture user behaviour patterns. It proposes that neural networks are not only capable of outperforming its heavier expert systems counterparts but in many ways better suits the demands and dynamic nature of the problem. In exploiting the strengths of neural networks in recognition, classification and generalisation this research illustrates the effectiveness of the neural network contribution to the application of intruder detection.

## 1. Introduction

Over the last decade, as computers have proliferated into almost every area of information processing, the amount of sensitive data placed on computers has generated an increasing need to certify the level of trustworthiness of computer systems. For example, the National Computer Security Centre in the U.S. has formulated an evaluation criteria (The Orange Book [3]) for establishing the levels of *trust* in the security of computer systems. The effect of this is fundamentally to relay a degree of assurance that evaluated systems are secure and can be trusted to the level that they have been certified. Security procedures on these systems must not only encompass the protection of information against unauthorised disclosure, modification or even destruction but also maintain the integrity of computing systems and address the ever increasing problem of computer misuse. Computer misuse manifests itself as anomalous behaviour, both of individual system users and of the system as a whole. Current computer security systems do not generally protect against [10]:

- intruders posing as legitimate system users
- a highly privileged user behaving destructively
- a legitimate user taking advantage of mistakes in the configuration of system security measures or other system vulnerabilities
- Trojan horses, or executable programs that have been altered to perform some new improper function

It has become increasingly clear that although systems can be certified to be secure, users may not. They can be cleared to use the system, but they can never be fully trusted. This situation is referred to as the *insider threat problem*. The user is often the weak link in an otherwise trusted system [11]. The electronic medium obscures physical recognition and, as such, a user on the system can hide the presence of hackers and impersonators or even give misbehaving legitimate users the ability to claim innocence.

Recently, the effort to address these problems has given rise to research aimed at producing viable automated intruder/anomaly detection systems, where the goal is to identify threats to the system by anomalous user behaviour or anomalous system behaviour. The observation that intruder detection is essentially a rule and experience based application, has led to research in moulding expert systems to this task. There are currently three examples of such systems at varying stages of production:

- The Information Security Officer's Assistant (ISOA), Planning Research Orgnisation, Government information Systems, R&D.
- Wisdom and Sense, Safeguards Systems Group, Los Alamos National Laboratory.
- Network Anomaly Detection and Intruder Report (Nadir), Computer Research & Applications Group, Los Alamos National Laboratory.

It is noted that all three of the examples above are expert systems. However, where expert systems can adequately handle the static rules that may alert to an intrusion, modeling the inexactness and dynamic nature of user behaviour has resulted in problems. This area is often dealt with incompletely, for example, Winkler and Page [11] noted that in addition to expert systems, ‘a much higher form of analysis will be needed to prevent aggregation attacks and to detect suspicious and undesirable trends in user behaviour’. They have acknowledged that the main difficulty in modeling user behaviour is in making the distinction between normal user behaviour shifts versus a subtle attack by an intruder and *updating* the expert systems user profiles in response.

Due to the necessity of churning through an ever increasing number of rules that attempt to address almost all the possible events that characterise a user, expert systems based anomalous user behaviour detection systems are prohibitively resource intensive. The system must also deal with the fact that the available information about a user may represent only a small fraction of all the possible information about a user. Since it is not feasible to keep a full historical account of each user the system must be able to generalise from the available information smoothly and efficiently [10].

It is also important that the detection system run as close to real-time as possible to keep up with the speed of user events or system reporting facilities e.g. auditing, in order to catch the clues and information that may only be present for a short time. The expert system based intruder detection applications, e.g. Wisdom & Sense [10], require quite a significant percentage of a system’s resources (including CPU, memory, disk I/O and disk space). While real-time processing with W&S is possible, it is not cheap.

Not only does an ideal intruder detection system need be able to generalise efficiently and to run as close to real time as possible but it also has to perform one of the most important tasks that a human systems programmer has to perform. It has to be able to acquire knowledge on the behavioural patterns characteristic of each individual user on the system in order to target possible intruder presence. The dynamic quality of user behavioural patterns is one of the greatest source of ambiguous ‘rules’ in intruder detection. The challenge is to implement a system that adapts to the changing nature of user behaviour, which is the motivation for applying neural networks to this problem. By exploiting the generalisation qualities of neural networks, we are able to move one step further in this field by producing a detection system that is able to sensibly extrapolate or generalise small deviations in behaviour that it may not have seen before, thereby incorporating the dynamic behaviour of a user elegantly and efficiently.

## 2. Adaption of the MLFF Network for Intruder Detection

The performance of a neural network in an application is generally measured during recall mode. The recall processing phase of a neural network corresponds to the decoding of the stored content of the network, which in the case of the Multilayered Feed Forward Network (MLFF Network), would have been encoded in the network during the training phase. In order to perform the classification, recognition and generalisation processing tasks, the MLFF Network is employed as a universal approximator where it is capable of finding an approximation of a multivariable function $h(\mathbf{x})$ through the supervised training of an input-output mapping from a set of training examples. The classical framework for this problem is given by the approximation theory [12].

The goal of this work is to produce a network that is able to encapsulate the behaviour of a user and yet is flexible enough to slowly and continually adapt to the changing characteristics often exhibited by the users on a computer system. It was found that although the MLFF Network was able to contribute a number of desirable attributes to the application, there were also a number of inherent limitations presented by the traditional theory and learning concepts of these networks. As a result, these problems were circumvented by implementing a number of changes to the classical theory of the feed forward network and the backpropagation algorithm.

The first limitation presented by the original MLFF Network, is the requirement that all training set data be available at the time of training. This is clearly impractical since the inputs to be accepted by the network will occur as a stream of continuous data generated by a user’s activity on the system. Therefore, the decision was made to proceduralise the training stage to the point where a single input could be ‘learnt’ at a time. This allows patterns to arrive at any time and in any order.

The second limitation is the notion of distinct training and classification/recall stages. Traditionally a feed forward network becomes trusted to perform the classification of a given input vector only after overall network error has been reduced to under a specified termination error. Achieving this condition marks the beginning of the classification stage. However, this methodology produces two distinct phases, training and recall, causing opposition to the continual adaptation required of the network. Furthermore, since the input data occurs as a continuous stream generated by user activity, the difficulty will be in the ability of the network to make a clear distinction as to what point in the data stream will the network consider itself suitably trained, and considering the dynamic nature of user behaviour, there is no guarantee that the section of the data stream that the network has trained on truly typifies the user. This problem is compounded by the fact that user behaviour will most likely continue to change. Therefore the modified network must be able to both learn and classify an input at the same time. However like a traditional network, the output of the network cannot be trusted during the early stages of learning its input set as it may not have encapsulated the founding or initial behavioural patterns of a user yet.

It was decided that the network would indicate when it had reached a 'trusted' error (trusted error meaning that the average error of the training set is used to measure against the termination error for cessation of network training) after which point the stable classification of user inputs could take place. This combined with the proceduralisation of the training stage produced a network that could continue to both learn and classify indefinitely.

We also need to address the issue of having incomplete training sets. The network requires that all the input patterns that it is required to learn and classify be presented to it in a training set. This means that the network only recognises and classifies whatever is fed to it in the form of its inputs. For example, for the network to recognise the activity and inactivity times of a user, it must be given both the activity and inactivity times. It cannot recognise the inactivity times of the user given only the activity times. This becomes a problem in this application since the network can only train with the information available on the system, therefore only one set of patterns can be presented to it. The network must then recognise patterns that have *not* been given to it from the patterns that have. This problem was addressed by introducing random inputs during the training of a user input. These random inputs are considered to be the 'negative' of the user's input and have the effect of introducing error and noise into network training as well as serve to help the network distinguish the patterns that have not been provided to it from the ones that have.

If a negative input hits a user input pattern which the network has already learnt, it will reduce the network response for that user input pattern by a fraction, and this simulates the slow 'forgetting' of any pattern. This ability is particularly desirable in this application when considering the dynamic and sometimes volatile nature of system users. The network must be pliant enough to mould itself to new behavioural changes while slowly forgetting those patterns that it had already learnt should they no longer typify a user.

The rate at which these negative inputs should be presented to the network varies depending of the type of application or pattern set being represented. For pattern sets where the number of possible patterns is small or easily distinguishable from each other (*linearly separable*), negative inputs should be presented rarely. If there are large pattern sets to learn, and the input patterns are *linearly separable*, then quite large levels of negative pattern presentation may be required for correct training and classification. In this application the rate of negative presentation for each network was chosen appropriately for each pattern set to be learnt and classified.

## 3. Detection of Anomalous User Behaviour

Our goal is to investigate the effectiveness of neural networks at modeling user behavioural patterns so they can distinguish between normal and abnormal behaviour. In order to model user behaviour we identified and isolated the system logs that were required as sources of information for the networks. These logs being /etc/utmp, /etc/wtmp, /usr/adm/pacct, and /usr/adm/sulog, provided the required user activity information from where we derived the following behavioural characteristics which typifies users on the system :

- User activity times - The time at which a user is normally active.
- User login hosts - The set of hosts from which a user normally logs in from.
- User foreign hosts - The set of hosts which a user normally accesses via commands on the system (eg. FTP hosts).
- Command set - The set of commands which a user normally uses.
- CPU usage - The typical CPU usage patterns of a user.

Figure 1 illustrates how a complete system for the detection of user behavioural anomalies is structured. The coordination process is responsible for channeling system information into the neural networks. Each of the behavioural characteristics are both modeled by a modified feed forward network, as well as checked by a limited static rule filter for easy breaches of security. The processing performed by the networks then produced results for each user characteristic to be fed as input into a unifying correlation network. The desired response of the correlation network is a value close to $-1$ for a user who is not deviating from their normal behaviour, and a value of 1 for a user who is deviating in the most extreme sense. An arbitrary threshold value of 0.5 was used to determine when to raise an alarm for anomalous user activity. If the output of the correlation network is above this threshold then an alarm is raised. It is also important to note that a fundamental requirement of this detection mechanism is that the threshold value be adjustable by the system programmer in order to allow adaptation to the required sensitivity needed in the computer system.

## 4. Results

Results were obtained on the departmental student machine (mundil.cs.mu.OZ.AU) which is a Silicon Graphics 4D/340S server. This machine is a medium to large multi-user system with 128Mb of memory, 4 CPUs. It has

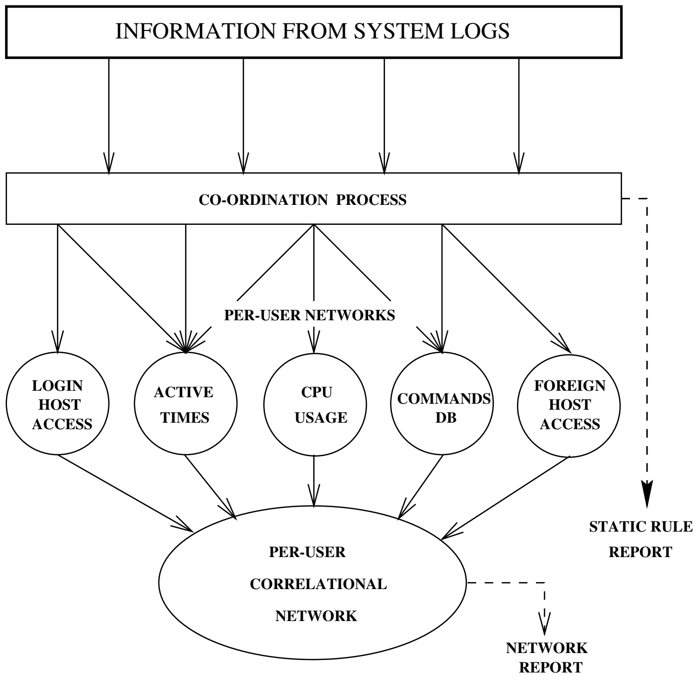


Fig. 1: Structure Of An Automated User Behaviour Anomaly Detection System

approximately 35 users on the system on average and approximately 157 users maximum. There is also an average of about 200 logins per day, peaking at over 1000.

In order to test the effectiveness of this system, it was allowed to run for a period of two weeks. Then the entire neural network detection system was tested by introducing a sample of anomalous behaviour profiles obtained from various system administrators. In summary these anomalous events included:

- Using `fsp`, a file transfer protocol which crackers favour as it is hard to trace.
- Being active at early morning hours (between 12 midnight and 3am).
- Creating connections to various known hacker sites (e.g. Swinburne, ANU, etc).
- Using IRC (Internet Relay Chat)
- Running various system commands looking to see if any well known bugs still exist that are not yet fixed.
- Using emacs

The goal of this exercise is to determine if the output of the per-user correlation networks indicated anomalous activity in the accounts to which anomalous activity was introduced.

The response of the correlation network for each user was recorded and graphed. A summary of the resulting graphs are displayed in Figure 2. They illustrate the response of the network when processing a normal user with non-varying behaviour, a normal user with changing behaviour, and and account under attack.

The following is a table that illustrates the percentage of CPU, memory and I/O load our system placed on the tested student system to maintain the neural networks. These results are of particular importance considering that one of the major reasons why expert systems versions of these detection systems are not viable is the heavy demands that they place on system resources. For practical viability, the detection system must use between 1 to 2 percent of the available system resources.

| System Load | % Of Total CPU Resources Required | % Of Total Memory Resources Required | % Of Total Disk IO Resources Required |
|---|---|---|---|
| Average | 0.84% | 0.4% | 0.02% |
| Maximum | 3.5% | 1.7% | 0.1% |

Table 1 : Summary of the Percentage of System Resources Required by the Neural Network Detection System

Our tested system under maximum load had 157 active users at once, while on average it had 35 users. Under maximum load the memory requirements are approximately 2.3Mb. Under normal load this comes down to approximately 0.5Mb.

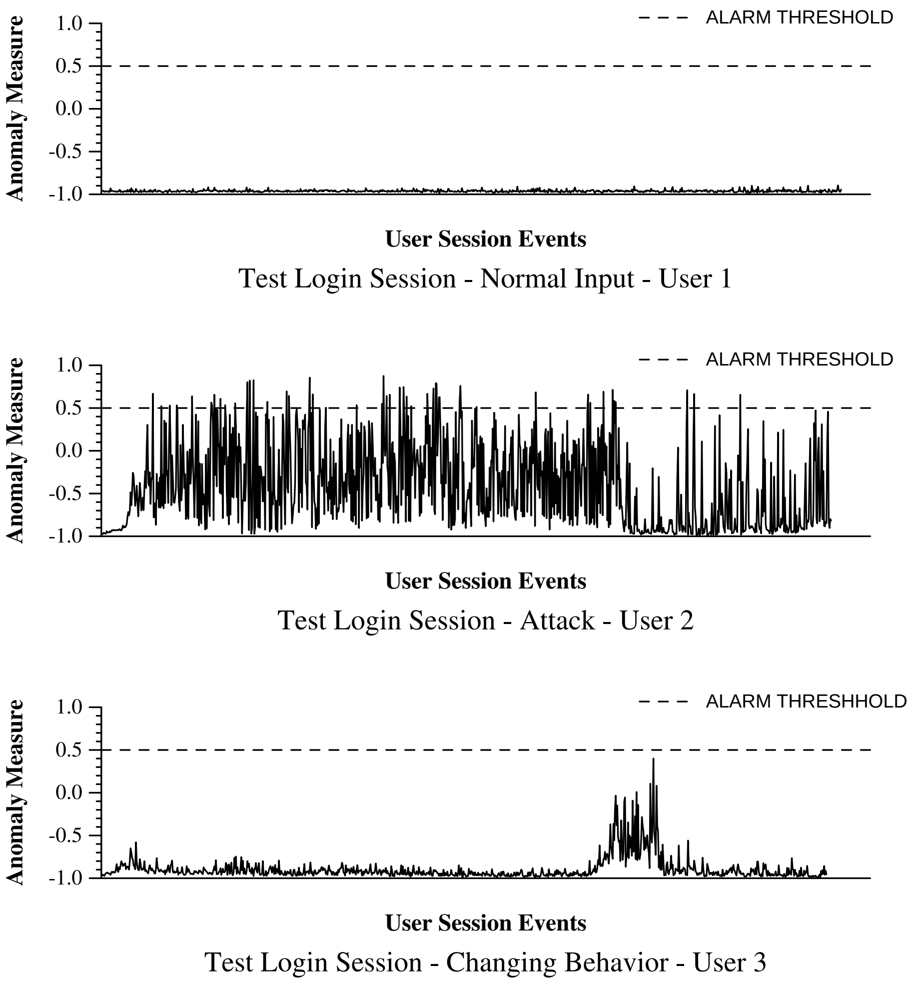


Fig. 2: Summary Graphs of Correlation Network for Three Accounts Under Testing

The total disk space requirements of our proposed system for a student machine with 1000 accounts would be 14.6Mb. On the machine we tested this would account for 1.4% of the machine's disk space.

I/O resources are consumed during the reading and writing back of the state of each user's neural network. The state of each active network in memory is written out to disk every 1/2 an hour to prevent loss of information.

## 5. Conclusion

The introduction of the computer has brought about, what is popularly termed, the "Information Revolution". Never before has such a wealth of data, both public and private, been so accessible and obtainable. With this revolution has come certain undesirable elements, being the underworld of computer fraud and crackers, who quite often achieve their aims by breaking into computer systems and impersonating legitimate system users.

Many methods preventing intrusion into computer systems have been implemented, but these are far from perfect. When preventive measures fail, many sites rely on the local expertise of their systems administrators to apprehend the intruders. This task is both time consuming and error prone, requiring a human to sift through copious quantities of data looking for deviations in user behaviour.

An automated system for detecting anomalous user behaviour will help alleviate a, sometimes unrealistic, burden on systems administrators. Some attempts have been made into this area but quite often they only concentrate on changes in system activity rather than user behaviour. Other systems which do model user behaviour are generally expert rule based and require large amounts of data processing, memory and disk space. These requirements may counter the incentive to use them.

Neural networks provide a robust approach to pattern learning and recognition that can model, generalise about, and classify many different types of input data where the exact nature of the input data is not known. The traditional multilayered feed forward networks used in this study presented some serious hurdles for an automatic intruder detection system. We have overcome the limitations of the traditional neural network by designing a network that has the ability to quickly learn new patterns, and to slowly forget them, as well as continue to adapt to changing patterns in the input stream. We have achieved this primarily through the introduction of a random presentation of input patterns. This way the network will also slowly forget any patterns it has learnt unless they are re-enforced through repeated presentation to the network.

A prototype for an automatic user behaviour anomaly detection system has been implemented and successfully tested. We have established the memory and CPU resource requirements of such a detection system based on real data and found that such a system would impact our tested UNIX system resources by less than 1% under average load and no more than 4% under the heaviest loads. Such an implementation would also require about 15Kb of disk space per user to maintain the state of the entire system.

## Acknowledgements

I would like to thank the cooperation and support of :

The Staff and System Administrators at the Departments of Computer Science and Electrical Engineering, University of Melbourne.

The Staff at the Computer Crime Division, Australian Federal Police

The Staff at the Computer Crime Division, Victorian State Police

## References

[1] D. A. Curry, "UNIX System Security, A Guide for Users and System Administrators", *Addison-Wesley,* 1992.

[2] V. Faber, "LANL Network Security: Intrusion Detection", *Network Security Symposium,* February 6, 1994.

[3] U.S. Department of Defense, "Department of Defense Trusted Computer System Evaluation Criteria", DOD 5200.28-STD, *U.S. Government Printing Office*, December 1985.

[4] D. B. Parker, "Crime By Computer", *Charles Scribner's Sons, New York,* 1976.

[5] S. Garfinkel and G. Spafford, "Practical UNIX Security", *O'Reilly & Associates, Inc.,* June 1994.

[6] O. Bowcott and S. Hamilton, "Beating The System", *Bllomsbury Publishing Ltd.,* 1992.

[7] Site Security Policy Handbook Working Group, "RFC-1244 Site Security Policy Handbook", *Internet Engineering Task Force,* June 1991.

[8] C. Stoll, "Stalking the Wily Hacker", *Communications of the ACM,* May 1988, pp. 484-497.

[9] R.C. Summer, "An overview of Computer Security", *IEEE Tutorial - Computer and Network Security,* 1984, pp. 9 - 33.

[10] H.S. Vaccaro and G.E. Liepins, "Detection of Anomalous Computer Session Activity", *1989 IEEE Computer Society Symposium on Security and Privacy,* May 1989, pp. 280-289.

[11] J.R. Winkler and W.J. Page, "Intrusion and Anomaly Detection in Trusted Systems", *Fifth Annual Computer Security Applications Conference,* Dec 1989, pp. 39-45.

[12] J. M. Zurada, "Introduction to Artificial Neural Systems", *WEST Publishing Company,* 1992.

[13] T. Poggio and F. Girosi, "Networks for Approximation and Learning ", *IEEE 78(9): 1481-1497,* 1990.

[14] Y. H. Pao, "Adaptive Pattern Recognition and Neural Networks", *Addison-Wesley,* 1989.