Defence Research and Development Canada
Recherche et développement pour la défense Canada

AutoCorrel I: A Neural Network Event Correlation Approach

Maxwell Dondo, Nathalie Japkowicz and Reuben Smith

Defence R&D Canada – Ottawa
TECHNICAL MEMORANDUM
DRDC Ottawa TM 2005-193
October 2005

Canada

AutoCorrel I: A Neural Network Event Correlation Approach

Maxwell Dondo

Nathalie Japkowicz
University of Ottawa

Reuben Smith
University of Ottawa

Defence R&D Canada – Ottawa

Technical Memorandum

DRDC Ottawa TM 2005-193

October 2005

Author

"original signed by"

M. Dondo

Approved by

"original signed by"

Dr. J. Lefebvre
Head/IO Section

Approved for release by

"original signed by"

Dr. A. Ashley
Chief Scientist

Abstract

Intrusion detection analysts are often swamped by multitudes of alerts originating from installed intrusion detection systems (IDS) as well as logs from routers and firewalls on the networks. Properly managing these alerts and correlating them to previously seen threats is critical in the ability to effectively protect a network from attacks. Manually correlating events can be a slow tedious task prone to human error. We present a two-stage alert correlation approach involving an artificial neural network (ANN) autoassociator and a single parameter decision threshold-setting unit. By clustering closely matched alerts together, this approach would be beneficial to the analyst. In this approach, alert attributes are extracted from each alert content and used to train an autoassociator. Based on the reconstruction error determined by the autoassociator, closely matched alerts are grouped together. Whenever a new alert is received, it is automatically categorised into one of the alert clusters which identify the type of attack and its severity level as previously known by the analyst. If the attack is entirely new and there is no match to the existing clusters, this would be appropriately reflected to the analyst. There are several advantages to using an ANN based approach. First, ANNs acquire knowledge straight from the data without the need for a human expert to build sets of domain rules and facts. Second, once trained, ANNs can be very fast, accurate and have high precision for near real-time applications. Finally, while learning, ANNs perform a type of dimensionality reduction allowing a user to input large amounts of information without fearing an efficiency bottleneck. Thus, rather than storing the data in TCP Quad format (which stores only seven event attributes) and performing a multi-stage query on reduced information, the user can input all the relevant information available and instead allow the neural network to organise and reduce this knowledge in an adaptive and goal-oriented fashion.

Résumé

Les analystes de la détection d'intrusion sont souvent inondés par une multitude d'alertes qui émanent des systèmes de détection d'intrusion (IDS pour intrusion detection systems) installés ainsi que des journaux des routeurs et des pare feu du réseau. La gestion adéquate de ces alertes et leur corrélation avec des menaces détectées antérieurement est essentielle pour pouvoir protéger efficacement le réseau contre les attaques. L'établissement manuel d'une corrélation entre les événements peut s'avérer un travail lent et fastidieux, sujet à des erreurs humaines. Nous présentons ici une approche de corrélation des alertes en deux étapes qui fait appel à un autoassociateur basé sur un réseau neuronal artificiel et une unité d'établissement d'un seuil de décision paramétrique. En regroupant ensemble les alertes étroitement appariées, cette approche s'avérera bénéfique pour l'analyste. Dans cette approche, les attributs de l'alerte sont extraits du contenu de chaque alerte et utilisés pour réaliser l'apprentissage d'un autoassociateur. En fonction de l'erreur de reconstruction déterminée par ce dernier, les alertes étroitement appariées sont groupées ensemble. Lorsqu'une nouvelle alerte est reue, elle est automatiquement classée dans un des groupes d'alertes qui identifient le type d'attaque et son niveau de gravité, déterminés antérieurement par l'analyste. Lorsque l'attaque est entièrement nouvelle et qu'elle ne correspond à aucun groupe existant, cela est également indiqué à l'analyste. L'utilisation d'une approche basée sur un réseau neuronal artificiel comporte plusieurs avantages. Tout d'abord, ces types de réseaux acquièrent les connaissances directement à partir des données, sans qu'un expert humain doive établir l'ensemble des faits et des règles du domaine. En second lieu, après la période d'apprentissage, les réseaux neuronaux artificiels s'avèrent très rapides et présentent une précision élevée pour des applications en temps quasi réel. Enfin, pendant leur période d'apprentissage, ils réalisent un type de réduction de la dimensionnalité qui permet à l'utilisateur de saisir d'importants volumes d'informations sans devoir craindre que cela cause un goulet d'étranglement nuisant à l'efficacité. Ainsi, plutt que de stocker les données en format TCP quadruple (qui stocke uniquement sept attributs d'événement) et de réaliser une interrogation en plusieurs étapes sur les informations réduites, l'utilisateur peut entrer toutes les informations pertinentes disponibles et permettre plutt au réseau neuronal d'organiser et de réduire ses connaissances d'une manière adaptative et orientée but.

Executive summary

AutoCorrel I: A Neural Network Event Correlation Approach

Maxwell Dondo, Nathalie Japkowicz, Reuben Smith; DRDC Ottawa TM 2005-193; Defence R&D Canada – Ottawa; October 2005.

Background

In their daily work, intrusion detection analysts often analyse multitudes of alerts generated by installed intrusion detection systems (IDS) as well as firewall and router logs on their networks [1, 2]. Properly managing these alerts and correlating them to previously seen threats is critical in their ability to effectively protect a network from attacks. Manually correlating events can be a slow tedious task prone to human error.

In this work, we propose to use a fast and accurate method of correlating network events. We propose to use a neural network-based novel detection approach to identify and cluster alerts into smaller attack categories. In that way, the analyst's job is made significantly easier.

Principal results

In this work, we were able to apply our neural network method to labelled DARPA alerts obtained from the 1999 DARPA IDS evaluation data set. As expected, we successfully clustered 48 alerts into six clusters with a minimum of 93% accuracy. With this success, we tested this approach on unlabelled `incidents.org` alerts. The approach was also successful, but with less accuracy compared with the DARPA data.

We therefore carried out an analysis to determine the source of the reduced accuracy. We found out that varying the separating metric (reconstruction error) affected the results. We carried out two additional experiments with fixed and variable reconstruction errors. We found out that the variable clustering approach significantly improved the accuracy of the results from a minimum of 59% to 76% . With this approach we managed to cluster 514 alerts into 15 clusters.

Significance of results

The results are significant in two ways. First, the approach was successful in meeting its original objective of reducing the number of alerts that the analyst has to deal

with. Further developments on this approach could produce a product that would be handy to IDS analysts like the Canadian forces' CIRT team at CFNOC. Secondly, the further experimentation that we carried out showed that there is room for improving the accuracy of the approach.

Future work

Future work will focus on improving the accuracy of the approach. One way would be to revisit the reconstruction error metric and determine if there is a better way to perform the clustering. It would also be worth investigating the use of different tools or a combination of neural networks and other methods such as machine learning or statistical methods.

Sommaire

AutoCorrel I : A Neural Network Event Correlation Approach

Maxwell Dondo, Nathalie Japkowicz, Reuben Smith; DRDC Ottawa TM 2005-193; R & D pour la défense Canada – Ottawa; octobre 2005.

Contexte

Dans leur travail au jour le jour, les analystes de la détection d'intrusion doivent souvent analyser une multitude d'alertes générées par les systèmes de détection d'intrusion (IDS pour intrusion detection systems) installés ainsi que par les journaux des routeurs et des pare feu dans leurs réseaux [1, 2]. La gestion adéquate de ces alertes et leur corrélation avec les menaces détectées antérieurement est essentielle pour pouvoir protéger efficacement le réseau contre les attaques. L'établissement manuel d'une corrélation entre les événements est un travail lent et fastidieux, sujet à l'erreur humaine.

Dans ce travail, nous proposons d'utiliser une méthode rapide et précise pour corréler entre eux les événements réseau. Nous proposons d'employer une nouvelle méthode de détection basée sur un réseau neuronal pour identifier et grouper les alertes en catégories d'attaques réduites. De cette faon, le travail de l'analyste est rendu significativement plus facile.

Principaux résultats

Dans ce travail, nous avons pu appliquer notre méthode, basée sur un réseau neuronal, à des alertes répertoriées par la DARPA, provenant de l'ensemble de données d'évaluation des IDS de cet organisme (1999). Comme prévu, nous sommes parvenus à grouper 48 alertes en six groupes, avec une précision minimale de 93%. À la suite de cette réussite, nous avons testé cette approche sur des alertes non répertoriées `incidents.org`. Notre approche a connu du succès là également, mais avec une précision moindre que celle que nous avons obtenue avec les données de la DARPA.

Nous avons par conséquent effectué une analyse pour déterminer la cause de cette réduction dans la précision. Nous avons constaté que nous obtenions des résultats différents en faisant varier la métrique de séparation (erreur de reconstruction). Nous avons mené deux expériences additionnelles, avec des erreurs de reconstruction fixe et variable. Nous avons constaté que la méthode de groupage variable permettait d'améliorer significativement la précision des résultats, et de faire passer celle ci d'une

valeur minimale de 59% à 76%. Grce à cette approche, nous sommes parvenus à grouper 514 alertes en 15 groupes.

Importance des résultats

Les résultats sont importants pour deux raisons. Tout d'abord, grce à cette approche, nous sommes parvenus à réaliser l'objectif initial, qui consistait à réduire le nombre d'alertes que l'analyste devait traiter. Des raffinements de cette approche pourraient conduire à un produit qui s'avérerait fort commode pour les analystes de la détection d'intrusion, comme ceux de l'EIIC des Forces canadiennes au CORFC. En second lieu, les expériences supplémentaires que nous avons menées ont démontré qu'il était possible d'améliorer la précision de la méthode.

Travaux ultérieurs

Les travaux ultérieurs seront axés sur l'amélioration de la précision de la méthode. On pourrait par exemple réexaminer la métrique de l'erreur de reconstruction et déterminer s'il n'y aurait pas un meilleur moyen de réaliser le groupage. Il serait également intéressant d'étudier l'emploi d'outils différents, ou une combinaison de réseaux neuronaux et d'autres méthodes comme l'apprentissage automatisé ou les méthodes statistiques.

Table of contents

List of figures

List of tables

Acknowledgements

We would like to thank Dr Peter Mason for his advice and assistance in preparing this document.

1 Introduction

Intrusion detection analysts can be overwhelmed by the multitudes of alerts that they receive on a daily basis. The majority of these alerts are not new, but the analyst must go through each one of them to determine the threat they each pose. In some cases, analysts use multiple IDS rules to classify alerts or use some of the generic correlation tools that come bundled together with IDSs.

The majority of the existing correlation tools use elementary approaches to correlate attacks. For example, Shadow [3] and ACID [4], use the IP addresses to correlate attacks. However, it is known that IP addresses may be spoofed, therefore using IP addresses alone may not provide a sufficient measure to classify the threat possed by an alert.

Currently, there is significant interest in alert correlation research. Recent work reported by Haines *et al.* [5], details some of the common correlation tools and approaches. The majority of these approaches take one alert metric at a time to correlate with other possible attacks. More sophisticated approaches use statistical methods on multiple alert metrics [1]. Statistical approaches usually depend on the underlying behavioural distributions such as Gaussian distribution; ANNs do not make prior assumptions about the data they handle [6, 7].

Julisch *et al.* used a conceptual clustering algorithm in their model [8]. They resolved the problem of training their system by relying on clustering algorithms rather than classification algorithms. They advocated having the administrator hand-modify the correlation system once per month to reflect the changes in the network environment. Their system also required that the correlation system have information specific to the local network before initial operation of the system. The approach requires a seasoned analyst to perform this initial configuration.

Dain and Cunningham [9] also presented another machine learning-based correlation system. In their paper, Dain *et al.* trained machine learning algorithms such as neural networks and decision trees to recognise attack scenarios based on a novel list of features. Hätälä *et al.* [10] criticise this work for it's use of a simplistic dataset. The Dain *et al.* research was only tested with a DEFCON conference dataset [11]. The use of this dataset simplified the problem of alert correlation because attackers were motivated by points in the competition and no points were awarded for stealthy attacks. As such, many of the attacks originated from a single IP address (Dain *et al.* [9]).

Other traditional correlation techniques involve the storage of historical data for future correlations. Due to the huge amounts of data that needs to be stored, a format called the *TCP Quad format* [12] is usually used to store reduced data content. It only

stores six event attributes, namely `date, time, Source IP, Source Port, Dest IP, Dest Port, Protocol`. However, by reducing the data, there is less information available to correlate with.

Our work undertakes to design an ANN-based event correlation engine that takes into account the majority of the alert attributes. We have incorporated a neural network whose speed is excellent for near real-time applications. A reduced dimension clustering algorithm ensures that very little additional computational effort is expended in performing the final decision making process. Neural networks are also adaptive and do not lose information as with classical correlation techniques that use data stored in TCP Quad format [12] and perform a multi-stage query on this reduced information or data.

We propose an approach that utilises most of the information available in a given alert. In our approach, most previously seen or collected attack data is used to train an ANN without prior analysis by experts. Once trained, this information is retained in the ANN as connection weights and does not need as much storage as the original alert data. In addition, this approach is fast enough to provide an analyst with close to real-time information.

We demonstrate this approach by applying the system to the labelled 1999 DARPA IDS evaluation data set [13] as well as unlabelled `www.incidents.org` alerts. We use Snort [14] to extract alerts from the supplied raw TCPDump [15] format data.

We begin by presenting a brief background on network event correlation in Section 2. Then we we present the background theory on ANNs in Section 3. At the end of this section we discuss the ANN model (the autoassociator) that is used in this work. To measure the performance of our model, we define and present experimental effectiveness measures in Section 3.3. This is followed by the results in Section 4. Finally, the conclusions are presented in Section 5.

2 Network Event Correlation

IDS researchers and developers are actively working on new and better methods to implement automated alert correlation from both homogeneous and heterogeneous sensors. Current work on IDS reporting standards [16, 1, 5] will facilitate the sharing of alert information within the IDS community.

We "tailor" our approach as closely as possible towards a model that would be capable of receiving and processing alert content from different homogeneous and heterogeneous IDS sensors. However, in this first generation model, we extract alert features that are common to many IDS sensors, while using Snort-based alerts [14]. Future work will harden this approach by looking at a broader spectrum of IDS sensors as opposed to considering just one. This includes any standards that IETF [16] might have developed at that time.

Our work is closely related to approaches by other correlation researchers like Vlades *et al.* [1] and Dain *et al.* [9]. Although we do not carry out analysis on the alert attributes themselves, we extract them and use them to determine the final correlation model. Our method of attribute extraction mostly consisted of reading attribute data from the text-based alert rather than creating new ones.

In this work, we extract variables information from the alert content. This consists of the header information associated with the alert. To maximise correlation and to ensure that all the important factors of a possible attack are taken into consideration, as much information as possible is extracted from the alert content. These features are listed in Table 1. Some attributes of the alerts were dropped for various reasons; for

Table 1: *Feature extraction for network event correlation.*

Feature	Feature	Feature	Feature
timeStamp	ipSrcAddress	ipDestAddress	ipVer
portSrc	portDest	ipIsIcmpProtocol	ipIsIgmpProtocol
ipIsTcpProtocol	ipIsUdpProtocol	ipLen	ipDgmLen
ipId	ipTos	ipTtl	ipOptLsrr
ipPacketDefrag	ipReserveBit	ipMiniFrag	ipFragOffset
ipFragSize	icmpCode	icmpId	icmpSeq
icmpType	tcpFlag1	tcpFlag2	tcpFlagUrg
tcpFlagAck	tcpFlagPsh	tcpFlagRst	tcpFlagSyn
tcpFlagFin	tcpLen	tcpWinNum	tcpUrgPtr
tcpOptMss	tcpOptNopCount	tcpOptSackOk	tcpOptTs1
tcpOptTs2	tcpOptWs	tcpHeaderTrunc	udpLen

example, the source IP address was dropped because this may give some misleading

clusters in cases where the IP address is spoofed; like in DoS attacks (for example). We also felt that the variable IP addresses on different alerts could be used as a strong determining factor in alert clustering, when we know that different alerts may originate from the same IP address. Early trails of our model also showed that the exclusion had no impact on the results and the remaining attributes were able to provide the necessary classification required. In-fact, the IP addresses were used to verify that the clustering was good. So it was our contention that the remaining variables should be able to identify and correlate any attack based on the remaining input variables.

The alert variable input vector $\mathbf{x}$, extracted from the TCP/IP content of the alert, is independent of the sensor used for the IDS. Since the majority of IDS sensors and logs provide the packet header content, the same information may be equally extracted. In the case of sensors that do not provide all this information, but log the offending packets, we propose using Snort to generate alerts with the offending packets and then extracting the alert attributes from there.

3 AutoCorrel I: An ANN Event Correlation Model

In this section, we present the basic theory of the ANN and how the algorithms are used to build the event correlation model used in this work.

3.1 Artificial Neural Networks

ANN models attempt to emulate the human brain through the dense interconnection of simple computational elements called neurons [17]. Each neuron is linked to some of its neighbours through synaptic connections of varying strengths. Learning is accomplished by continuously adjusting these connection strengths (weights) until the overall network outputs the desired results. These weight adjustments are based on mathematical algorithms used in solving nonlinear optimisation functions.

3.1.1 The Neuron

Similar to the biological nervous system, the basic computational element of an ANN is called the neuron or processing node. The neuron model is based on highly simplified considerations of the biological neuron. A simple node is shown in Figure 1, where N inputs are summed at the node. Each input u_i is connected to the pro-

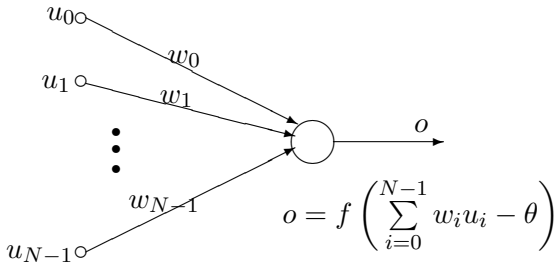

Figure 1: *The basic neuron*

cessing node through the synaptic connections, which are represented by connection strengths called weights w_i. A bias term θ is also used at each node. The bias term represents the $y-$intercept of the separation plane in cases where the separation plane does not pass through the origin. The sum is fed through a transfer function f, called the activation function, to generate the output o. The signal flow is considered unidirectional as indicated by the arrows.

Although ANNs are constructed using this fundamental building block, there are significant differences in the architectures and driving fundamentals behind each ANN model.

3.1.2 The Activation Function

The activation function f plays a pivotal role in the functioning of the neuron. It determines the node output. As in Figure 1, the neuron output signal is given by:

$$o = f\left(\mathbf{w}^T\mathbf{u}\right) \tag{1}$$

where $\mathbf{w}$ is the weight vector defined as

$$\mathbf{w} \equiv \left[\begin{array}{cccc} w_1 & w_2 & \cdots & w_N \end{array}\right]^T$$

and the input vector $\mathbf{u}$ is defined as

$$\mathbf{u} \equiv \left[\begin{array}{cccc} u_1 & u_2 & \cdots & u_N \end{array}\right]^T$$

There are many different types of activation functions f to choose from, depending on the application [17, 18, 19]. Some of the commonly used activation functions are shown in Figure 2. These activation functions are the *hard-limiter*, the *threshold logic*, and the *sigmoid*. Since real applications are usually modeled as continuous functions, the most commonly used continuous activation function is the sigmoid.

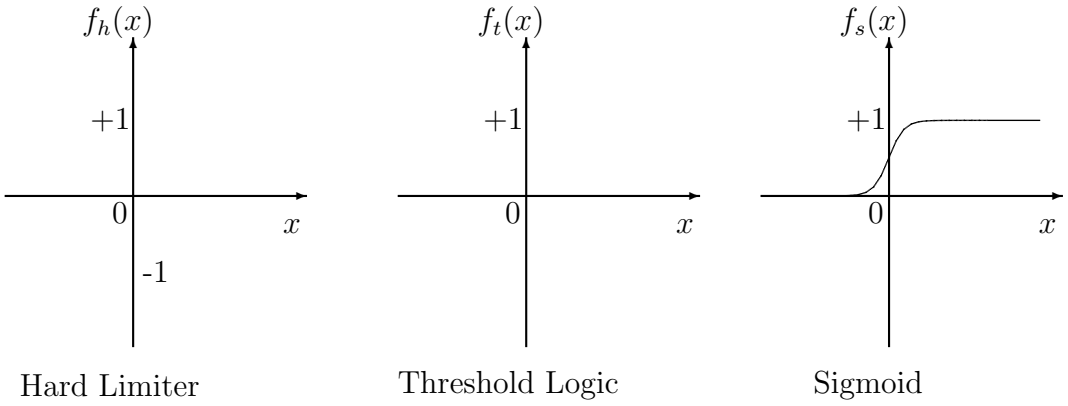

Figure 2: *Activation functions*

Activation functions may be either unipolar, for positive output, or bipolar for output that may be positive or negative. For example, the bipolar sigmoidal activation

function is defined as:

$$f(x) \equiv \frac{2}{1 + \exp^{-\lambda x}} - 1 \tag{2}$$

and the unipolar sigmoidal activation function is defined as

$$f(x) \equiv \frac{1}{1 + \exp^{-\lambda x}} \tag{3}$$

where λ is a constant.

A special case of an ANN is a single node based on the neuron model shown in Figure 1 and is called a *perceptron* after the work of Rosenblatt [17]. A perceptron consists of one or more neurons. If a continuous activation function is used, the neuron model is known as a *continuous perceptron*. A continuous perceptron is capable of classifying *linearly separable* classes of data of the form $a\mathbf{x} + b$. Multiple nodes in this format form a single layer multi-node ANN capable of classifying linearly separable data patterns.

3.1.3 Multi-Layer ANNs

To emulate massively interconnected biological systems, ANNs have to be similarly interconnected. ANNs are the simple clustering of primitive artificial neurons. This clustering occurs by creating layers of neurons which are connected to one another. Figure 3 shows a multi-layer perceptron. An input layer interfaces with the outside world to receive inputs and an output layer provides the outside world with the network's outputs. The rest of the neurons are hidden from view, and are called *hidden layers*.

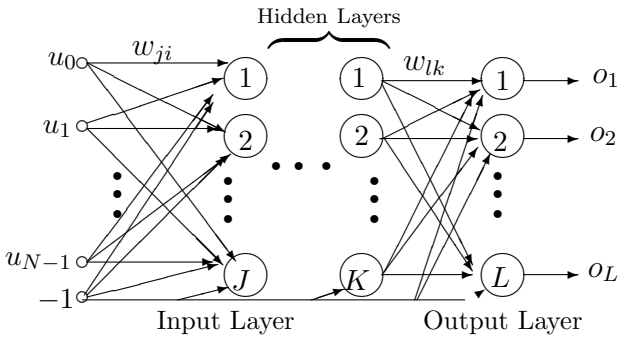

Figure 3: *Multi-layer perceptron*

The objective of using a multi-layer perceptron is to be able to classify patterns that linear classifiers (single layer ANNs) are incapable of classifying. The most

important attribute of multi-layer ANNs is that they can learn to classify a problem of any complexity. The biggest challenge is usually in deciding the number of hidden layers in an ANN.

Zurada [19] gives an extensive discussion on the design of the number and size of hidden layers in a given architecture; nevertheless, trial and error methods have been widely used. If the number of hidden layers is too large, the ANN architecture will have problems generalizing; it will simply memorize the training set, making it useless for use with new data sets.

Inter-layer connections within an ANN architecture can take the following forms [19]:

- In a *fully-connected* ANN, each neuron on one layer is connected to every neuron on the next layer.
- In a *partially-connected* ANN, a neuron on one layer does not have to be connected to all neurons on the next layer.

If signal flow direction is taken into consideration, these two architectures can be further refined:

- In a *feedforward* ANN, the neurons on one layer send their output to the neurons in the next layer, but they do not receive any input back from the neurons in the next layer.
- In a *bi-directional* ANN, the neurons on one layer may send their output to the next layer or the preceding layer, and the subsequent layers may also do the same.
- In a *hierarchical* ANN connection, the neurons of a lower layer may only communicate with neurons on the next level of layers.
- In a *resonance-connected* ANN, the layers have bi-directional connections, and they can continue sending messages across the connections a number of times until previously defined conditions are achieved.

In more sophisticated ANN structures the neurons communicate among themselves within a layer, this is known as intra-layer connections. These take the following two forms:

- In fully- or partially-connected *recurrent* networks, neurons within a layer communicate their outputs to neurons within the layer. This is done a number of times before they are allowed to send their outputs on to another layer.

- In *on-center/off-surround* ANNs, a neuron within a layer has excitatory connections to itself and its neighbors, and has inhibitory connections to other neurons. The neurons exchange their output signals a number of times until a winner is found. The winner is allowed to update its and its members' weights.

The overall architecture of an ANN depends on the mappings required, the type of input patterns, and the learning rules to be used.

3.1.4 ANN Training

Similar to the brain, ANNs learn from experience by changing the ANN's connection weights until a solution is found through the meeting of a previously defined objective function (see Section 3.1.5). The learning ability of an ANN is determined by its architecture and by the algorithm chosen for training. The training methods [17] fall into broad categories:

- In *unsupervised training*, hidden neurons find an optimum operating point by themselves, without external influence.
- *Supervised training* requires that the network be given sample input and output patterns to learn. It is guided through the learning process until a satisfactory optimum operating point or a predefined threshold is reached. The most common training termination criteria is by setting a training threshold.

Backpropagation training is a form of supervised learning that has proven highly successful in training multi-layered ANNs [20, 19]. Information about errors is filtered back through the system and is used to adjust the connections between the layers, thus improving performance.

ANNs can be trained *on-line* or *off-line*. In off-line training algorithms, the weights do not change after the successful completion of the initial training. This is the most common training approach; especially in supervised training. In on-line or real time learning, weights continuously change when the system is in operation [19].

3.1.5 Training Rules

There are a wide variety of learning rules used with ANNs. Error minimisation algorithms are used to determine the convergence levels when updating weights. In general, all ANN learning involves the iterative updating of the connection weights until the desired convergence is achieved. Most training algorithms start by initializing the weights to 0 or very small random numbers. This weight update is given by:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \Delta\mathbf{w}^k \tag{4}$$

Equation 4 is the ANN *general learning rule* [19]. The numerous learning rules, which are variations of this rule, only differ by the mathematical algorithms used to update the connection weights, or more specifically to calculate the value of $\Delta \mathbf{w}^k$ at each iteration k. Some of the common training rules are as follows:

- In the *Hebbian* rule [19, 18], the connection weight update $\Delta \mathbf{w}^k$ is proportional to the neuron's output. This was the first ANN learning rule [17, 21].
- The *perceptron* rule [17] updates the weights based on the difference between the desired output d and the actual neuron's response o.
- The *delta* learning rule [17, 21] is based on the minimisation of the mean square error (MSE) as represented by the error function E, as shown in Equation 5.

  $$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta \nabla E\left(\mathbf{w}^k\right) \tag{5}$$

  where η is a learning constant, and ∇E is the gradient of the error function E, defined by:

  $$E_k = \frac{1}{2}\left(d^k - o^k\right)^2 \tag{6}$$

  The objective is to iterate Equation 5 until the error E approaches zero (or a preset threshold value).

  For an ANN with P training patterns, and K outputs, the *root-mean square error* (also known as the MSE [19]) is defined as:

  $$E_{rms} = \frac{1}{PK}\sqrt{\sum_{p=1}^{P}\sum_{k=1}^{K}(d_{pk} - o_{pk})^2} \tag{7}$$

- The *Widrow-Hoff* [18, 19] learning rule (sometimes called the *Least Mean Square* learning rule) is considered a special case of the delta learning rule in that the neuron output o is independent of the activation function f.
- The most widely used supervised training approach which is derived from the Widrow-Hoff algorithm is the *error backpropagation training algorithm*. As the name implies, the error $\Delta \mathbf{w}^k$ is propagated back into the previous layers. This is done one layer at a time, until the first layer is reached.

  Consider an ANN with one hidden layer, K outputs, J hidden nodes, I inputs, and P training patterns. The output layer weights are adjusted as follows:

  $$w_{kj} = w_{kj} + \eta \delta_{ok} y_j, \quad \text{for k} = 1, \cdots, \text{K}, \ \text{j} = 1, \cdots, \text{J} \tag{8}$$

  where η is a learning constant and the output error $\delta_{\mathbf{ok}}$ is given by

  $$\delta_{ok} = \frac{1}{2}(d_k - o_k)(1 - o_k^2), \quad \text{for k} = 1, 2, \cdots, \text{K} \tag{9}$$

The weight update for the hidden layer is as follows:

$$w_{ji} = w_{ji} + \eta\delta_{yj}u_i, \quad \text{for k} = 1, \cdots, \text{K}, \ \text{i} = 1, \cdots, \text{I} \tag{10}$$

where the output error $\delta_{\mathbf{yj}}$ is given by

$$\delta_{yj} = \frac{1}{2}(1 - y_j^2)\sum_{k=1}^{K}\delta_{ok}w_{kj}, \quad \text{for j} = 1, 2, \cdots, \text{J} \tag{11}$$

The process is iteratively repeated until a preset threshold of the MSE (Equation 7) is achieved.

3.2 The Correlation Model

In this work we use an ANN functioning as an autoassociator [19]. An input vector $\mathbf{x}_i$, made up of alert TCP/IP information, is fed through an autoassociator through weights w_i. The objective is for the autoassociator to reproduce the vector $\mathbf{x}_i$. However, the reproduction is not perfect, and a reconstruction error is a measure of this imperfection. Similar attack scenarios have very similar attributes, and we expect their reconstruction errors to be very close. Similar or closely similar alerts are grouped together based on the reconstruction error. This idea is an extension of the work by Japkowicz *et al.* [22, 23] on Novelty Detection. In that work, it was seen that the autoassociator was very good at grouping previously seen events together, discriminating them from novel events. That kind of clustering, however, was coarse-grained. Our purpose here, is to use the autoassociator in a similar capacity, but as a finer-grained clustering system.

Based on the Novelty Detection approach by Japkowicz *et al* [24], three layer fully connected feedforward ANN with N input nodes, N output nodes and $J < N$ hidden nodes (unoptimised) was used. The output at node i of each layer is given by:

$$y_i = f(\sum_{k=1}^{K_i} w_{ik}y_k) \tag{12}$$

where y_i is the output of neuron i after receiving K_i signals from the neurons of the preceding layer, and $y_i = x_i$ for the input layer.

The network is trained using the error-backpropagation algorithm with the objective of reconstructing the input space at the output. This algorithm has been successfully used in other intrusion detection research [25], and this is also the training algorithm used by Japkowicz *et al.* [24] in their Novelty Detection work. The training objective is to minimise the square errors E, as given by:

$$E = \sum_{i=1}^{n} ||x_i - y_i||^2 \tag{13}$$

The ANN weights are iteratively updated until convergence is achieved Once training has been completed, a threshold level is used to compare with the resultant reconstruction error e_i given by:

$$e_i = ||\mathbf{x}_i - \mathbf{y}_i|| \tag{14}$$

for each input vector $\mathbf{x}_i$ during ANN recall. This is illustrated in Fig. 4. Equation 14 essentially reduces a 42-dimensional vector into a 1-dimensional threshold metric e used for clustering. The result of the clustering is multiple alert groups that have

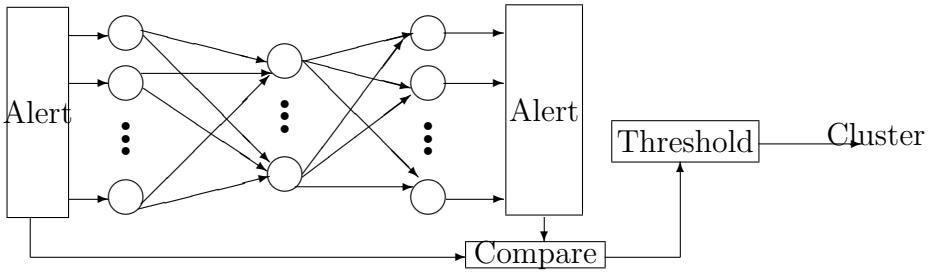

Figure 4: *AutoCorrel I model.*

similar or closely similar attributes.

However, as we will see later, the 42 to 1 mapping between the 42 tuple input and output vectors $\mathbf{x}$ and $\mathbf{y}$, and the scalar threshold value e, is a possible source of errors in this work. These errors are discussed in more detail in Section 4.3.

3.3 Experimentation Effectiveness Metrics

To analyse the success of our approach, we have developed measures of success criteria. Our approach will take raw alert data, and use part of it for training. Another part of the data is used for testing the trained autoassociator. We expect to get crisp clusters of similar, closely similar and identical alerts. To understand and analyse the outcome of our experiments, we have used basic performance measures to characterise the effectiveness of the system.

If our approach puts **B** alerts into a cluster, and we determine that **A** alerts do not belong to this cluster, then the alert placement accuracy (APA) for this cluster is $\frac{\mathbf{B-A}}{\mathbf{B}}$. In a similar manner, if **C** is the total number of alerts to be clustered, and **D** is the number of alerts that were misclustered, then the overall alert placement accuracy (OAPA) would be $\frac{\mathbf{C-D}}{\mathbf{C}}$.

If our test data has **E** alert types, we expect to get **E** crisp clusters for each alert type. Now, if **F** alert types don't have clusters of their own, then the cluster placement accuracy (CPA) is $\frac{\mathbf{E-F}}{\mathbf{E}}$.

4 Results

The model was tested in two steps. First, the model was tested with labeled DARPA [13] alerts. This portion served as a way to validate the ANN model. The main part of the model testing were carried out with unknown alerts from `www.incidents.org`. In each case, 10 000 alerts were used for training. An autoassociator with 42 inputs, 10 hidden layers and 42 outputs was trained, using the error backpropagation algorithm with a training constant η of 0.4. Training was initially fixed to 5 000 epochs, but could be changed if desired.

4.1 Labeled DARPA Alerts

The ANN was trained in 90 000 epochs (an MSE of 0.003) using 20 000 alerts generated from *week 5* of the 1999 DARPA IDS Evaluation data set [13]. Forty-eight labeled alerts not used in the training were used to test the ANN. The reconstruction errors for the individual alerts are plotted for each alert as shown in Figure 5.

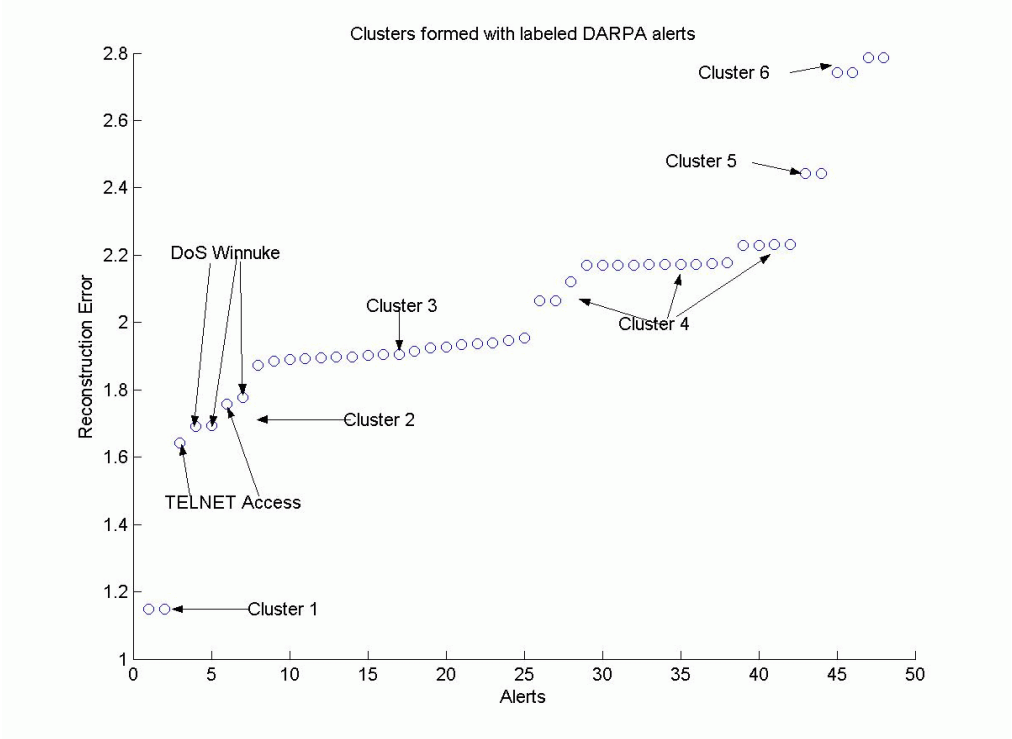

Figure 5: *Clusters generated by labeled DARPA alerts.*

The test alerts represented six attack categories. We expected to get six crisp clusters of these 48 alerts; namely *Telnet Access, DOS Winnuke, FIN Scan, Tiny Fragments,*

Splice attack, and DOS Land attack. Our experiment produced six (CPA = 100%) clusters as indicated in Figure 5. However, our model correctly clustered 45 of these alerts into six clusters (OAPA = 93.75%). Three alerts were placed into a cluster (cluster 2) where they shouldn't be (APA = 30%). This translated to 1 erroneous cluster, and three misplaced alerts. As shown in Figure 5, three *DOS Winnuke* alerts were wrongly placed in cluster 2 (*Telnet Access*).

To explain the three stray *DOS Winnuke* alerts in cluster 2 instead of cluster 4 where the other 17 appeared, we compared the values of the input attributes to see where there are significant differences with alerts in cluster 4. Unlike alerts in cluster 2, all alerts in cluster 4 have the TCP FIN flag set. In addition, the alerts in each cluster use different values of the TCP urgent pointer; `0x31` for cluster 2 and `0xf5` for cluster 3. Despite these minor attribute differences, we expected the ANN to be able to "see through" these differences and still cluster the alerts in one cluster. After all, the *Telnet Access* attributes are different from the *DOS Winnuke* alerts. So, we can't really explain this discrepancy at this time.

The clustering results are summarised in Table 2. The table shows the six clusters

Table 2: *DARPA alert clusters.*

Alert	Alerts in cluster					
	1	2	3	4	5	6
Splice attack	2	0	0	0	0	0
Telnet Access	0	2	0	0	0	0
FIN Scan	0	0	18	0	0	0
DOS Winnuke	0	3	0	17	0	0
DOS Land attack	0	0	0	0	2	0
Tiny Fragments	0	0	0	0	0	4
Clustering Error (1-APA)	0	60%	0	0	0	0

formed and the individual alerts that fall into each cluster. It also shows the clustering errors for each of the six clusters. The table also shows the number of alert types that were erroneously clustered, e.g. the *DOS Winnuke* alert that was clustered into clusters 2 and 4.

The labeled DARPA data were clustered with very good results; a 100% CPA and a 93.75% OAPA. We conclude that the ANN clustering was successful in placing alerts into relevant clusters. Only one cluster contained erroneously placed alerts. These statistics were very encouraging and we decided to take a close look at unlabeled alerts that are available in the wild (an unsimulated environment).

Table 3: *A sample of cluster 4 alerts.*

```
[**][1:628:3] SCAN nmap TCP [**]
11/14-10:10:23.856507 163.23.238.9:80 -> 170.129.19.170:80
TCP TTL:44 TOS:0x0 ID:31290 IpLen:20 DgmLen:40
**A**** Seq:  0x300 Ack:  0x0 Win:  0x578 TcpLen:  20
[Xref=> http://www.whitehats.com/info/IDS28]

[**][1:628:3] SCAN nmap TCP [**]
11/14-10:10:18.856507 61.218.161.210:80 -> 170.129.19.170:80
TCP TTL:48 TOS:0x0 ID:30943 IpLen:20 DgmLen:40
**A**** Seq:  0x278 Ack:  0x0 Win:  0x578 TcpLen:  20
[Xref=> http://www.whitehats.com/info/IDS28]

[**][1:628:3] SCAN nmap TCP [**]
11/14-10:10:13.826507 61.218.161.210:80 -> 170.129.19.170:80
TCP TTL:48 TOS:0x0 ID:30662 IpLen:20 DgmLen:40
**A**** Seq:  0x20A Ack:  0x0 Win:  0x578 TcpLen:  20
[Xref=> http://www.whitehats.com/info/IDS28]

[**][1:628:3] SCAN nmap TCP [**]
[Classification:Attempted Information Leak] [Priority:  2]
11/14-10:10:08.786507 61.218.161.202:80 -> 170.129.19.170:80
TCP TTL:48 TOS:0x0 ID:30366 IpLen:20 DgmLen:40
**A**** Seq:  0x198 Ack:  0x0 Win:  0x578 TcpLen:  20
[Xref=> http://www.whitehats.com/info/IDS28]
```

4.2 Unlabeled `incidents.org` Alerts

The approach was tested with a sample of 514 Snort alerts from `www.incidents.org`. A total of 20 000 alerts were used for training in 80 000 epochs until the MSE was 0.05. Similarly matching alerts were grouped into the same clusters. In Table 3, we show an example of *nmap scan* alert clustering produced by our approach. These closely related alerts belong to one cluster. As expected, the clustered alerts have closely matching attributes as defined earlier. We also note that the clustered alerts display other similar attributes that were not used in the clustering algorithm as variables; namely source and destination IP addresses.

The reconstruction errors plot of the individual alerts are shown in Figure 6. From Figure 6, we note that there are no clear cluster boundaries in some cases. It is evident that there are some cluster overlaps. We decided to carry out the analysis of these results in two stages; namely regular clustering and variable clustering by inspection.

4.2.1 Regular Clustering

This approach uses a regular threshold spacing between different clusters. The clusters are separated by a fixed change of 0.025 in reconstruction errors. This is illus-

Table 4: *Alert placement matrix for regular clustering.*

Cluster	Alerts																	
	NULL SCAN	TCP Port 0 Traffic	Squid Proxy SCAN	BACKDOOR Q Access	TCP nmap SCAN	Proxy Port SCAN	Short UDP packet	ICMP ISS Pinger	TCP Data Offset	ICMP Superscan	SOCKS Proxy SCAN	Land Attack	GNUTella Outbound	HTTP INSPECT	SHELLCODE x86	IP Reserved bit	TCP Header length	Error (1-APA)
1	**52**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	**14**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	**50**	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6%
5	0	0	0	**85**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	**75**	1	4	0	0	0	0	0	0	0	0	0	0	6%
7	0	0	0	0	**14**	0	1	3	0	0	0	0	0	0	0	0	0	22%
8	0	0	0	0	**12**	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	1	0	0	0	0	0	**5**	0	0	0	0	0	0	0	0	16%
10	0	0	0	0	0	0	0	0	**9**	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0	0	0	0
12	0	0	0	0	1	1	0	0	5	16	14	17	6	3	0	0	0	ND
13	0	0	0	0	0	0	0	0	0	0	**5**	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	2	0	0	0	**6**	0	0	0	25%
15	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	**5**	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0
18	0	0	0	0	0	0	0	0	**9**	0	0	0	0	4	0	0	0	30%
19	0	0	0	0	0	0	0	0	1	0	0	0	0	**2**	0	0	0	33%
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**7**	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**18**	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**40**	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**8**	0	1	11%
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0

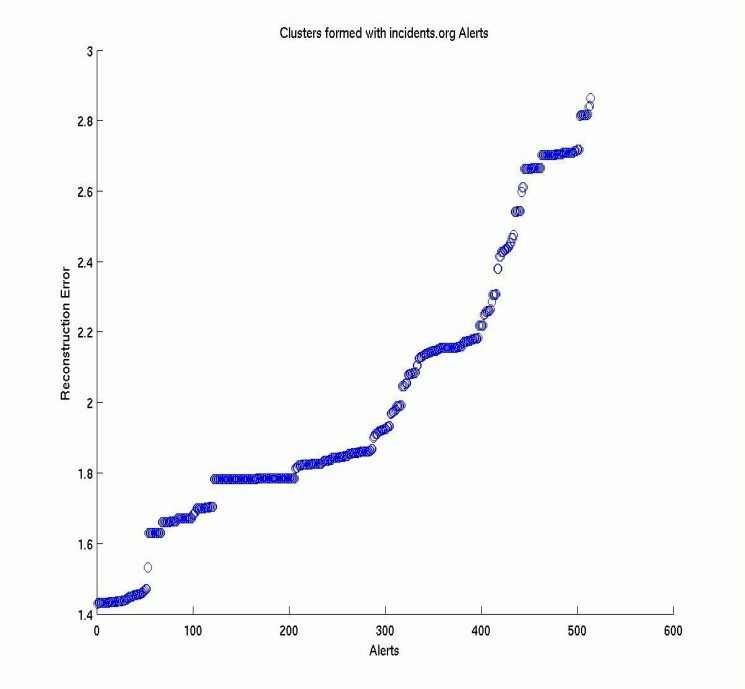

Figure 6: *Clusters generated by unlabelled alerts from* `incidents.org`.

trated in Figure 7 which shows the clustering overlaps on the clusters of Figure 6.The results are summarised in Table 4, where the ND (last column) represents an error value that cannot be determined because the clustering is not very clear. The bold figures represents the alert (column) that owns the cluster (row). The error value in the last column represents the error associated with placing alerts into each cluster (1− APA).

Using the regular clustering approach, we clustered the 514 test alerts into 27 clusters. Eighteen clusters accurately placed alerts into relevant clusters without errors (APA = 100%); although some alert groups were clustered into two or more clusters. Seven alert groups did not have clusters of their own. Twenty-one clusters had an APA of 85% or better. The OAPA for this approach was 82% and the CPA was 59%.

From these results and from Figure 6, some clusters have overlaps that cannot be separated by a fixed threshold value. The worst affected cluster is 12. Other overlaps are characterised by low APA values as in clusters 6, 14, 18, and 19. We also have seven alerts that span two or more clusters, and seven alerts that don't have clusters of their own. The possible sources of these errors will be presented in Section 4.3.

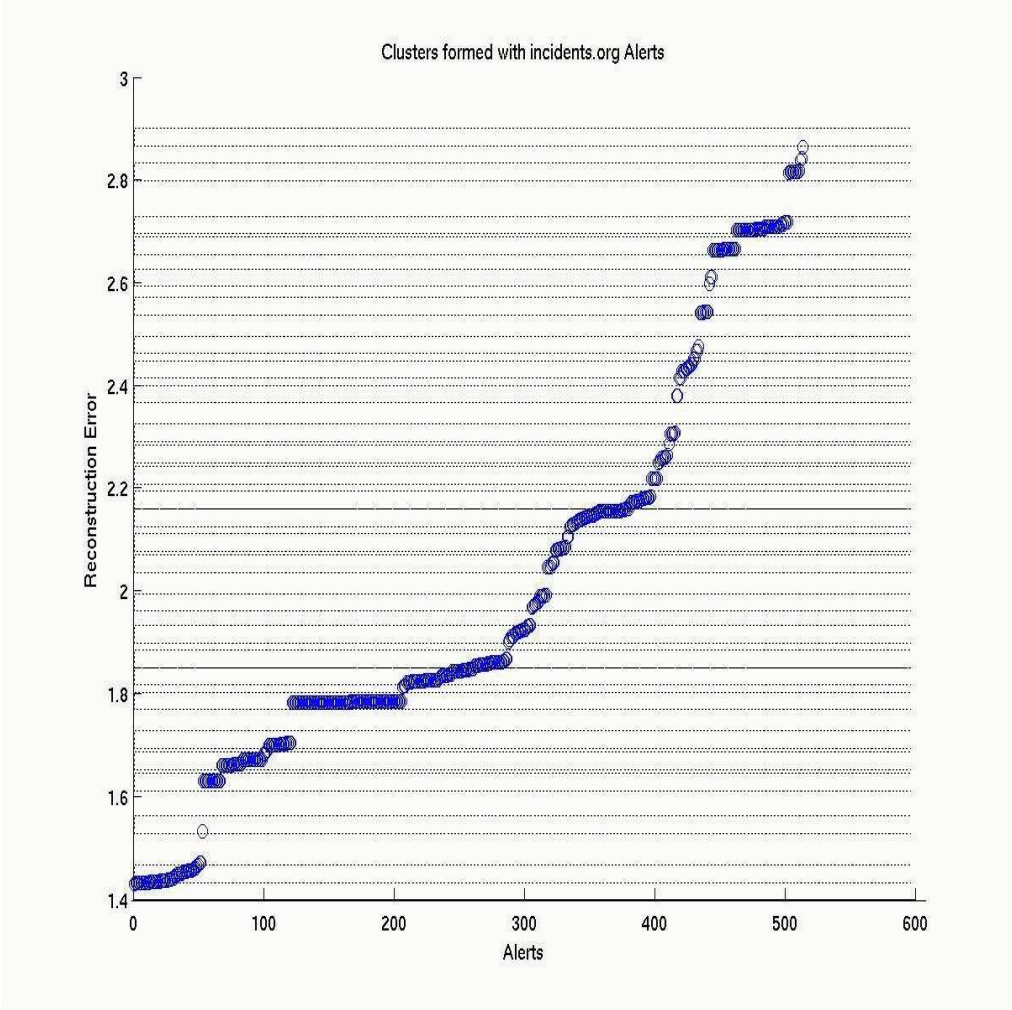

Figure 7: *Clusters generated by unlabelled alerts from* `incidents.org` *illustrating regular clustering.*

Table 5: *Alert placement matrix for variable clustering.*

Cluster	Alerts																		
	NULL SCAN	TCP Port 0 Traffic	Squid Proxy SCAN	BACKDOOR Q Access	TCP nmap SCAN	Proxy Port SCAN	Short UDP packet	ICMP ISS Pinger	TCP Data Offset	ICMP Superscan	SOCKS Proxy SCAN	Land Attack	GNUTella Outbound	HTTP INSPECT	SHELLCODE x86	IP Reserved bit	TCP Header length	Error (1-APA)
1	**53**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	**64**	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5%
3	0	0	0	**85**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**101**	1	5	3	0	0	0	0	0	0	0	0	0	8%
5	0	0	1	0	0	0	0	0	**16**	0	0	0	0	0	0	0	0	5%
6	0	0	0	0	1	1	0	0	4	**16**	1	0	0	0	0	0	0	30%
7	0	0	0	0	0	0	0	0	0	0	0	**17**	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	**7**	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	**18**	0	0	3	0	0	0	18%
10	0	0	0	0	0	0	0	0	1	2	0	0	0	**13**	0	0	0	20%
11	0	0	0	0	0	0	0	0	**10**	0	0	0	0	6	0	0	0	37%
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**8**	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	**18**	0	10%
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**48**	0	1	2%
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0
Error(1-APA)	0	0	ND	0	1%	ND	ND	ND	18%	11%	5%	0	0	45%	0	0	25%	

4.2.2 Variable Clustering by Inspection

Looking at Figure 6, we were also able to cluster the alerts through inspection. This resulted in clusters with no fixed threshold spacing. The threshold value varied form cluster to cluster. However, there were still some overlaps in the clusters. This is illustrated in Figure 8.

In Table 5, we show that we were able to cluster the 514 alerts into fifteen clusters. The clusters had varying clustering successes. We were able to place 477 alerts into thirteen clusters with an APA of 80% or better. This translates to an OAPA 92%. Clusters 6, 10 and 11 had low APA values. Four of the seventeen alert groups did not have clusters of their own (CPA = 76%). There were only two alert groups that fell into two different clusters, namely *TCP data offset* (clusters 5 and 11) and *Shellcode x86*(clusters 12 and 14). Table 6 shows the summary of the two methods. The variable clustering method is a marked improvement from the fixed threshold clustering. With varying success, AutoCorrel I was able to differentiate between subtle differences in different alerts and cluster them accordingly.

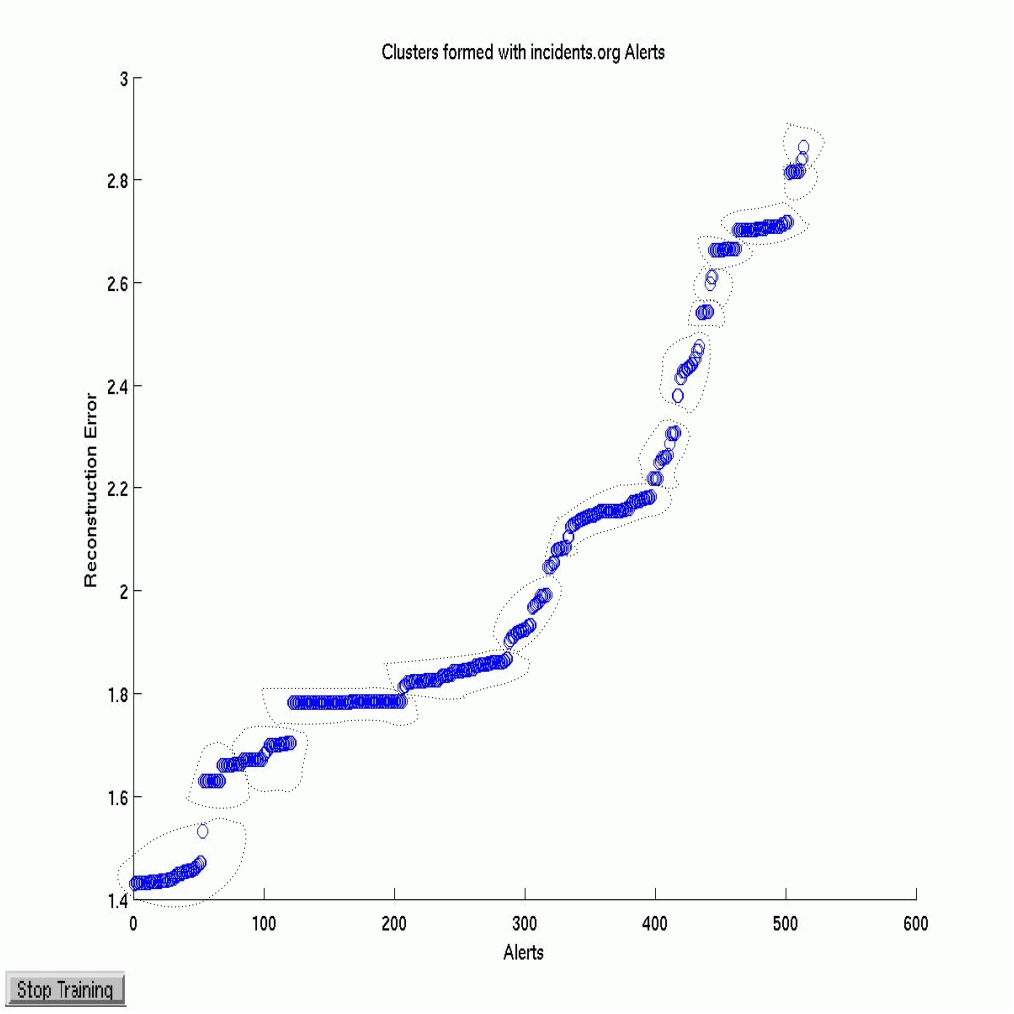

Figure 8: *Clusters generated by unlabelled alerts from* `incidents.org` *illustrating variable clustering.*

Table 6: *Clustering effectiveness for the incidents.org alerts*

Clustering Method	Group overlaps	OAPA	CPA
Regular clustering	8	82%	59%
Variable Clustering	2	92%	76%

4.2.3 Cluster Grouping

The results also show that some alerts can be grouped together because of the similarities in their signatures. To further assist an analyst with making crucial decisions, it is possible to group the clusters based on their similarities and the analyst's preference. For example, the six *scan* alerts can be grouped into one. This would reduce the number of clusters to eleven.

In a similar manner, all alerts spanning more than one cluster can be grouped together to form one *super-cluster*. For example, in Table 7, we list some of the clusters that could be grouped together. Depending on the analyst's preference, this grouping may be different, thus we suggest that a configurable decision visualisation engine be coupled to Autocorrel I. This reduces the number of alert categories the analyst has

Table 7: *Cluster Groups.*

"Super-Cluster"	Clusters
A	1, 4, 6, 8
B	5,11
C	12,14

to deal with.

4.3 Practical Impact of Clustering Errors

While the clustering process was successful overall, there were some clustering errors encountered. We will use the variable clustering results of Figure 5 to explain the varying error levels.

Closely following the six perfect clusters (1, 3, 7, 8, 12, and 15), we had the following finer grained cluster divisions:

1. The emergence of two or more distinct clusters for a given alert group as exemplified by the *Shellcode x86* alert group (clusters 12 and 14), and the *TCP data offset* alert group(clusters 5 and 11).

2. The "spilling over" of some of the alerts from the main clusters to the neighbouring clusters. This is exemplified in the *HTTP inspect* alert group (clusters 9–11) and *TCP data offset* (clusters 5–6 and 9–11).

Both errors 1 and 2 are very minor errors resulting from cluster overlaps. Although very close, the TCP/IP attributes for both alert categories were not "strong" enough to produce crisp clusters of their own. Some of the *TCP data offset* alerts are shown

in Table 8. The TCP/IP attributes of these three alerts closely match those in cluster 5, but for reasons stated above, they were placed in cluster 6.

Table 8: *Sample of* TCP data offset *alerts.*

```
[**][116:46:1] WARNING: TCP Data Offset is less than 5!
11/12-20:25:11.826507217.209.183.235:0 ->
207.166.252.249:0TCP TTL:236 TOS:0x0 ID:0 IpLen:20 DgmLen:40
*****R**Seq:  0x24A1C4C Ack:  0x24A1C4C Win:  0x0 TcpLen:  0

[**][116:46:1] WARNING: TCP Data Offset is less than 5!
09/05-07:34:31.984488 172.20.10.199:0 ->
138.97.150.9:0 TCP TTL:237 TOS:0x0 ID:0 IpLen:20 DgmLen:40
*****R**Seq:  0x58092C9A Ack:  0x58092C9A Win:  0x0 TcpLen:  12

[**][116:46:1] WARNING: TCP Data Offset is less than 5!
10/10-15:12:45.92650780.128.206.166:0 ->
32.245.166.119:0 TCP TTL:117 TOS:0x0 ID:25125 IpLen:20 DgmLen:40
DF*******F Seq:  0x720049 Ack:  0xD655185A Win:  0x5010 TcpLen:  0
```

We also had three cases of "stray" alerts in *TCP nmap Scan*, *Socks Proxy Scan* and *ICMP Superscan*. There isn't any significant difference between the clustered alerts and these stray alerts. These are likely the result of the cumulative effect of the ANN errors and the minor TCP/IP differences in these alerts and the clustered alerts.

The main errors encountered were in the four cases in which the alert groups failed to form individual clusters. The alerts for three of them are distributed inside other clusters (clusters 2 and 4). The overall reconstruction error for these alerts was indistinguishable with that of the clusters they are embedded in. The main cause of this type of error is attributable to the initial assumption of using a one-dimensional reconstruction error as the clustering attribute.

Some of these errors suggest the use of a supervised ANN to force the autoassociator to classify certain alerts into certain clusters. While the performance of AutoCorrel I is very good, the few errors encountered could possibly be reduced by conducting further research into the clustering algorithm of the autoassociator. For example, the threshold value dividing the different cluster boundaries was shown to produce better results if made variable rather than fixed. In addition, collapsing a 42 dimensional TCP/IP input vector into a one-dimensional (Section 3.2) decision-metric (reconstruction error) may also introduce some errors in the final results.

5 Concluding Remarks

A first generation system of an autoassociator correlation model was successfully designed and implemented. Its ability to cluster similar alerts with good accuracy could be a useful tool for the intrusion detection analyst. Our results showed that AutoCorrel I was often sensitive enough to small differences to set slightly different alerts apart; something that a human analyst could have missed (especially when swamped with the many alerts that are typical of networks today).

The processing speed produced by the ANN classifier makes this approach a good candidate for implementation in real time. Applications that would make use of this processing speed would utilise this model in conjunction with real-time operating systems on network devices or firewalls where the need for processing speeds and accuracy are important. The model could also be used as a correlation engine for an IDS that collects events from other Snort–based sensors.

Another advantage of using this neural network model is that it does not have a limitation on the amount of data it can handle. Its performance is also not affected by the amount of data it has to process. This is where our approach has advantages over other existing approaches in this area. The approach does not require frequent retraining, unless the number of new alerts significantly increases and the new alerts are constantly being misclassified.

The approach was successfully tested with a labelled data-set from the 1999 DARPA IDS evaluation data-set. We were able to accurately cluster 48 alerts into six clusters. The approach was also successfully tested on unlabelled alerts form the incident.org. The accuracy of the results were not as impressive as in the DARPA data, but with the variable form of clustering tested, the results were significantly improved; 514 alerts were grouped into only 15 clusters.

One current limitation of this model is that it can only directly handle Snort-based alerts. Alerts generated by other sensors have to be transformed to be compatible with this model. In fact, if the other sensor does not present all the attributes used by our model, the results are unpredictable, if not outright wrong. For sensors that log the actual offending packets, but don't provide enough attribute data in their logs, this approach would still be applicable by running the alerts through Snort first. Future work enhancements could expand AutoCorrel I to directly handle inputs from multiple heterogeneous sensors.

The errors associated with this approach may be attributable to the 42 to 1 mapping of the input variables to one dimensional decision metric–the reconstruction error. It is obvious that a lot of information is lost in the process. This calls for revisiting the initial definition of the clustering metric or the autoassociator used in the model;

this is currently being investigated. This may also require expanding the output of AutoCorrel I to cluster the alerts based on weighted values of the 42-tuple input and output metrics. This would also provide a possible ranking system for the alerts using the analyst's prior knowledge of the different alerts. Another possibility would be separating rare events from the common ones, by passing the unclustered alerts into another autocorrelator for reclassification.

Future work will also involve investigating the use of different clustering techniques. Self organising maps and EM algorithms have been used in other works, and we hope that they will be applicable to our work. Further work may also include the coupling of this model with good visualisation techniques to provide a possible product easy to use in analysing the clustered alerts.

References

[1] Valdes, Alfonso and Skinner, Keith (2001). Probabilistic Alert Correlation. In *RAID 2001, LNCS 2212*, pp. 54–68. Springer-Verlag.

[2] Carey, Nathan, Mohay, George, and Clark, Andrew (2003). Attack Signature Matching and DIscovery in SYstems Employing Heterogeneous IDS. In *Applied Computer Security Associates Conference*.

[3] SANS Institute (2001). Shadow. *SANS Institute*. Available on-line at http://www.nswc.navy.mil/ISSEC/CID/.

[4] Southcott Patrick (2002). Snort & Acid. *Online*.

[5] Haines, Joshua, Ryder, Dorene K., Tinnel, Laura, and Taylor, Stephen (2003). Intrusion Alert Correlation: Validation of Sensor alert Correlations. *IEEE Security & Privacy*, pp. 46–56.

[6] Lunt, T. F. (1993). A survey of intrusion detection techniques. *Computers & Security*, **4**(12), 405–418.

[7] Ning, Peng and Cui, Yun (2002). An Intrusion Alert Correlator Based on Prerequisites of Intrusions. *Department of Computer Science, North Carolina State University*.

[8] Julisch, Klaus and Dacier, Marc (2002). Mining Intrusion Detection Alarms for Actionable Knowledge. In *Proceedings of SIGKDD '02, the 8th International Conference on Knowledge Discovery and Data Mining*, pp. 366–375. Edmonton, Alberta, Canada: ACM.

[9] Dain, Oliver and Cunningham, Robert K. (2001). Fusing a Heterogeneous Alert Stream into Scenarios. In *Proceedings of the 2001 ACM Workshop on Data Mining for Security Applications*, pp. 1–13. Philadelphia, PA.

[10] Antti Hätälä, Camillo Särs, Addams-Moring, Ronja, and Virtanen, Teemupekka (2004). Event Data Exchange and Intrusion Alert Correlation in Heterogeneous Networks. In *Proceedings of the 8th Colloquium for Information Systems Security Education (CISSE)*, pp. 84–92. Westpoint, NY: CISSE.

[11] The DEFCON Data Set (Online). defcon.org. http://www.defcon.org/ (Not yet checked).

[12] Northcutt, Stephen and Novak, Judy (2000). Network Intrusion Detection : An Analyst's Handbook, 2 ed. Indianapolis, Indiana: New Riders.

[13] Lippmann, Robert, Haines, Joshua W., Fried, David J., Korba, Jonathan, and Das, Kumar (2000). The 1999 DARPA Off-Line Intrusion Detection Evaluation. *Computer Networks*, **34**(4), 579–595.

[14] Roesch, Martin (1999). Snort—Lightweight Intrusion Detection for Networks. In *Proceedings of LISA '99: 13th Systems Administration Conference*, pp. 229–238. Seattle, Washington: The USENIX Association.

[15] The TCPDump Program (Online). tcpdump.org. `http://www.tcpdump.org/` (Jan. 20, 2004).

[16] Erlinger Michael and Staniford-Chen Stuart (2003). Intrusion Detection Exchange Format (IDWG). *IETF*.

[17] Lippman, R.P. (1987). An Introduction to Computing with Neural Nets. In *IEEE ASSP Magazine*, pp. 4–22.

[18] Demuth, Howard and Beale, Mark (2001). Neural Network Toolbox, MathWorks Version 4.

[19] Zurada, J. M. (1992). Introduction to Artificial Neural Systems, New York NY: West Publishing Company.

[20] Ripley, B.D. (1996). Pattern Recognition and Neural Networks, 1 ed. Cambridge: Cambridge University Press.

[21] Widrow, B. and Lehr, M.A. (1990). 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. In *IEEE Proceedings*, Vol. 78, pp. 1415–1442.

[22] Japkowicz, N. (2001). Supervised versus Unsupervised Binary-Learning by Feedforward Neural Networks. *Machine Learning*, **42**(1/2), 97–122.

[23] Japkowicz, N., Myers, C., and Gluck, M (1995). A Novelty Detection Approach to Classification. In *The Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 518–523.

[24] Japkowicz, Nathalie (2001). Supervised Versus Unsupervised Binary-Learning by Feedforward Neural Networks. *Mach. Learn.*, **42**(1-2), 97–122.

[25] Ghosh, Anup K. and Schwartzbard, Aaron (1999). A Study in Using Neural Networks for Anomaly and Misuse Detection. In *Proceedings of the Eighth USENIX Security Symposium*.

Annex A: Acronyms and Abbreviations

ACL	Access Control List
APA	alert placement accuracy
ANN	Artificial Neural Network
CGI	Common Gateway Interface
CPA	cluster placement accuracy
DARPA	Defense Advanced Research Projects Agency
DDOS	Distributed Denial of Service
DOS	Denial of Service
DNS	Domain Name System
IDS	Intrusion Detection System
IDWG	Intrusion Detection Working Group
IETF	Internet Engineering Task Force
IP	Internet Protocol
MSE	Mean Square Error
OAPA	overall alert placement accuracy
OS	Operating System
SVM	Support Vector Machine
TCP	Transmission Control Protocol
URL	Uniform Resource Locator

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence R&D Canada – Ottawa
3701 Carling Avenue, Ottawa, Ontario, Canada K1A 0Z4

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable).

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title).

AutoCorrel I: A Neural Network Event Correlation Approach

4. AUTHORS (last name, first name, middle initial)

Dondo, Maxwell ; Japkowicz, Nathalie ; Smith, Reuben

5. DATE OF PUBLICATION (month and year of publication of document)

October 2005

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc).

40

6b. NO. OF REFS (total cited in document)

25

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered).

Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include address).

Defence R&D Canada – Ottawa
3701 Carling Avenue, Ottawa, Ontario, Canada K1A 0Z4

9a. PROJECT NO. (the applicable research and development project number under which the document was written. Specify whether project).

15BF29

9b. GRANT OR CONTRACT NO. (if appropriate, the applicable number under which the document was written).

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique.)

DRDC Ottawa TM 2005-193

10b. OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution beyond the audience specified in (11) is possible, a wider announcement audience may be selected).

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

Intrusion detection analysts are often swamped by multitudes of alerts originating from installed intrusion detection systems (IDS) as well as logs from routers and firewalls on the networks. Properly managing these alerts and correlating them to previously seen threats is critical in the ability to effectively protect a network from attacks. Manually correlating events can be a slow tedious task prone to human error. We present a two-stage alert correlation approach involving an artificial neural network (ANN) autoassociator and a single parameter decision threshold-setting unit. By clustering closely matched alerts together, this approach would be beneficial to the analyst. In this approach, alert attributes are extracted from each alert content and used to train an autoassociator. Based on the reconstruction error determined by the autoassociator, closely matched alerts are grouped together. Whenever a new alert is received, it is automatically categorised into one of the alert clusters which identify the type of attack and its severity level as previously known by the analyst. If the attack is entirely new and there is no match to the existing clusters, this would be appropriately reflected to the analyst. There are several advantages to using an ANN based approach. First, ANNs acquire knowledge straight from the data without the need for a human expert to build sets of domain rules and facts. Second, once trained, ANNs can be very fast, accurate and have high precision for near real-time applications. Finally, while learning, ANNs perform a type of dimensionality reduction allowing a user to input large amounts of information without fearing an efficiency bottleneck. Thus, rather than storing the data in TCP Quad format (which stores only seven event attributes) and performing a multi-stage query on reduced information, the user can input all the relevant information available and instead allow the neural network to organise and reduce this knowledge in an adaptive and goal-oriented fashion.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

Neural Network, Intrusion Detection System, Network Event Correlation, Alert Correlation, Autoassociator

Defence R&D Canada

Canada's leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca