Network Event Correlation Using Unsupervised Machine Learning Algorithms

Maxwell Dondo, Peter Mason, Nathalie Japkowicz and Reuben Smith

Defence R&D Canada – Ottawa
TECHNICAL MEMORANDUM
DRDC Ottawa TM 2006-193
November 2006
<table>
<thead>
<tr>
<th>1. REPORT DATE</th>
<th>2. REPORT TYPE</th>
<th>3. DATES COVERED</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOV 2006</td>
<td>N/A</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. TITLE AND SUBTITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Event Correlation Using Unsupervised Machine Learning Algorithms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5a. CONTRACT NUMBER</th>
<th>5b. GRANT NUMBER</th>
<th>5c. PROGRAM ELEMENT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5d. PROJECT NUMBER</th>
<th>5e. TASK NUMBER</th>
<th>5f. WORK UNIT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6. AUTHOR(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defence R&amp;D Canada - Ottawa Technical Memorandum DRDC Ottawa TM 2006-193 Canada</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defence R&amp;D Canada - Ottawa Technical Memorandum DRDC Ottawa TM 2006-193 Canada</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8. PERFORMING ORGANIZATION REPORT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10. SPONSOR/MONITOR'S ACRONYM(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11. SPONSOR/MONITOR’S REPORT NUMBER(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>12. DISTRIBUTION/AVAILABILITY STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved for public release, distribution unlimited</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>13. SUPPLEMENTARY NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>The original document contains color images.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>14. ABSTRACT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15. SUBJECT TERMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16. SECURITY CLASSIFICATION OF:</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORT</td>
</tr>
<tr>
<td>unclassified</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>17. LIMITATION OF ABSTRACT</th>
<th>18. NUMBER OF PAGES</th>
<th>19a. NAME OF RESPONSIBLE PERSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAR</td>
<td>108</td>
<td></td>
</tr>
</tbody>
</table>

Standard Form 298 (Rev. 8-98)  
Prescribed by ANSI Std Z39-18
Network Event Correlation Using Unsupervised Machine Learning Algorithms

Maxwell Dondo
Peter Mason
DRDC Ottawa

Nathalie Japkowicz
Reuben Smith
University of Ottawa

Defence R&D Canada – Ottawa
Technical Memorandum
DRDC Ottawa TM 2006-193
November 2006
Principal Author

*Original signed by Maxwell Dondo*

Maxwell Dondo

Approved by

*Original signed by J. Lefebvre*

J. Lefebvre
Head/NIO Section

Approved for release by

*Original signed by C. Boulet*

C. Boulet
Head/Document Review Panel

© Her Majesty the Queen in Right of Canada as represented by the Minister of National Defence, 2006

© Sa Majesté la Reine (en droit du Canada), telle que représentée par le ministre de la Défense nationale, 2006
Abstract

We have successfully implemented a two-stage event correlation model for intrusion detection system (IDS) alerts. The model is designed to automate alert and incidents management and reduce the workload on an IDS analyst. We achieve this correlation by clustering similar alerts together, thus allowing the analyst to only look at a few clusters instead of hundreds or thousands of alerts. The first stage of this model uses an artificial neural network (ANN)-based autoassociator. The autoassociator is trained to reproduce each alert at its output, and it uses the error metric between its input and output to cluster similar alerts together. The accuracy of the system is improved by adding another machine-learning stage which attempts to combine closely related clusters produced by the first stage into super-clusters. The second stage uses the Expectation–Maximisation (EM) clustering algorithm. The model and performance of this model are tested with intrusion alerts generated by a Snort IDS on DARPA’s 1999 IDS evaluation data as well as incidents.org alerts.

Résumé

Nous avons réussi l’implantation d’un modèle de corrélation d’événements à deux étapes pour les alertes IDS (système de détection d’intrusion). Ce modèle a été étudié afin d’automatiser la gestion des alertes et des incidents et réduire la charge de travail de l’analyste IDS. Nous réalisons cette corrélation en groupant les alertes semblables, ce qui permet à l’analyseur d’examiner des groupes au lieu de centaines, voire de milliers d’alertes. La première étape de ce modèle a recours à un auto-associateur basé sur ANN. Cet auto-associateur subit un apprentissage pour reproduire chaque alerte en sortie, et exploite une mesure d’erreur entre l’entrée et la sortie pour grouper les alertes semblables. La précision du système est rehaussée grâce à l’ajout d’une autre étape d’apprentissage machine dans laquelle on essaie de combiner les groupes étroitement reliés, obtenus dans la première étape, et former de supergroupes. La seconde étape fonctionne avec l’algorithme de groupage EM. Le modèle et sa performance sont testés au moyen d’alertes d’intrusion générées par un Snort IDS exploitant des données d’évaluation IDS 1999 du DARPA.
Executive summary

Network Event Correlation Using Unsupervised Machine Learning Algorithms


**Background:** This work is a follow-up of our earlier work, Autocorrel I [1], in which we attempted to make an intrusion detection system (IDS) analyst’s job easier by clustering similar alerts together, thereby reducing the number of alerts that an analyst has to deal with. In this work we tried to address the shortcomings of Autocorrel I, in particular the accuracy of the results.

In our conclusions to the Autocorrel I model, we noted that the results could be improved. We also noted that the most likely source of our errors was in the collapsing of a 42-tuple input pattern into a single threshold value which was used to perform the clustering. Therefore, in this work we added a second stage to provide a second layer of clustering. This second stage combines the threshold heuristic of the first stage (Autocorrel I) with raw attributes from Snort alerts.

We propose to use a neural network-based novelty detection approach and a machine-learning clustering algorithm to identify and cluster alerts into smaller attack categories. In this way, the analyst’s job is made significantly easier.

**Principal results:** We applied our approach to labelled DARPA alerts obtained from the 1999 DARPA IDS evaluation data set as well as to unlabelled incidents.org alerts. We successfully clustered sets of 500 alerts in each case with an accuracy of over 86%. The data was clustered into 21 and 18 clusters respectively. Thus, instead of sifting through 500 alerts, the analyst would only go through a maximum of 21 clusters for these particular data sets. This was a significant improvement from the results of Autocorrel I which produced an accuracy of 59% (76% with manual human intervention).

**Significance of results:** The accuracy of our results show that our original objectives were met. We were able to reduce the number of alerts that an analyst has to deal with on a daily basis, and we were able to significantly improve the accuracy of our results from the previous model. If developed further, this approach could produce a product that would be useful to IDS analysts like the Canadian Forces (CF) Computer Incident Response Team (CIRT) at the Canadian Forces Network Operations Centre (CFNOC). However, since the results were not perfect, more experimentation with different algorithms could improve the accuracy even more.
Future work: Although no future work is planned at this time, future work could focus on improving the accuracy of the system by using other clustering algorithms. An approach may also be required to help the analyst identify and manage the few instances of erroneous clustering.
Sommaire

Network Event Correlation Using Unsupervised Machine Learning Algorithms

Maxwell Dondo, Peter Mason, Nathalie Japkowicz, Reuben Smith; DRDC Ottawa TM 2006-193; R & D pour la défense Canada – Ottawa; novembre 2006.

Contexte: Ce travail est la poursuite d’un travail précédent, Autocorrel I [1], dans lequel on a essayé de faciliter le travail d’un analyste IDS en groupant des alertes similaires, afin de réduire le nombre d’alertes qu’un analyste devait examiner. Dans le travail actuel, nous avons essayé de corriger les lacunes de Autocorrel, tout particulièrement d’améliorer l’exactitude des résultats.

Dans nos conclusions sur le modèle Autocorrel I, nous avions souligné le fait que les résultats pouvaient tre améliorés. Nous avions aussi mentionné que la source la plus probable des erreurs découle de la réduction d’un modèle d’entrée à 42-tuples en un modèle à un seuil unique, nécessaire pour effectuer le groupage. C’est pourquoi, dans la recherche actuelle, nous avons ajouté une deuxième étape afin d’obtenir une seconde couche de groupage. Cette deuxième étape combine la solution d’une valeur seuil, celle de la première étape de Autocorrel I, aux attributs bruts des alertes Snort.

Nous proposons d’utiliser un réseau neuronal fonctionnant d’après une nouvelle approche de détection et un algorithme de groupage par apprentissage machine servant à l’identification et au groupage des alertes en petites attaques. Ainsi, le travail de l’analyste se trouve considérablement simplifier.

Principaux résultats: Nous avons mis en pratique notre approche sur des alertes DARPA étiquetées provenant des données d’évaluation IDS 1999 de DARPA, de même que sur des alertes non étiquetées incidents.org. Nous avons réussi à grouper des ensembles de 500 alertes dans chaque cas, avec une précision supérieure à 86%. Les données ont été réunies en groupes de 21 et 18 alertes respectivement. Ainsi, l’analyste au lieu de parcourir 500 alertes n’a plus à examiner que 21 groupes pour cet ensemble particulier de données. Il s’agit là d’une amélioration importante par rapport aux résultats de Autocorrel I, qui avait rendu une précision de 59% (76% dans le cas de l’intervention humaine).

Interprétation des résultats: L’exactitude de nos résultats démontre l’atteinte de nos objectifs de départ. Nous avons été en mesure de réduire le nombre d’alertes qu’un analyste doit traiter chaque jour, et nous sommes parvenus à améliorer sensiblement l’exactitude de nos résultats, par rapport au modèle précédent. Si nous poursuivons
cette recherche, nous allons pouvoir construire un produit utile aux analystes IDS, comme le CF CIRT est utile pour le CFNOC. Comme nos résultats ne sont pas parfaits, un surplus d’expérimentation avec des algorithmes différents devrait rehausser encore plus l’exactitude des résultats.

**Travaux à venir:** Bien que rien ne soit prévu pour le moment, les travaux à venir devront se concentrer sur l’exactitude du système via l’utilisation d’autres algorithmes de groupage. Une approche devrait aussi être envisagée afin d’aider l’analyste à déceler et à gérer les quelques cas de groupages erronés.
# Table of contents

Abstract ................................................................. i
Résumé ................................................................. i
Executive summary ...................................................... iii
Sommaire ............................................................... v
Table of contents ......................................................... vii
List of figures ............................................................ ix

1 Introduction ............................................................. 1
  1.1 Background ......................................................... 1
    1.1.1 Previous Work ................................................. 1
  1.2 Our Approach ...................................................... 3

2 Artificial Neural Networks (ANN)s and Clustering Algorithms ....... 5
  2.1 Artificial Neural Networks ........................................ 5
    2.1.1 The Neuron ................................................. 5
    2.1.2 The Activation Function ..................................... 6
    2.1.3 Multi-Layer ANNs .......................................... 7
    2.1.4 ANN Training ............................................... 9
    2.1.5 Training Rules .............................................. 9
    2.1.6 The Autoassociator ........................................ 11
  2.2 Clustering Algorithms ............................................ 11
    2.2.1 The Expectation–Maximisation (EM) Algorithm ........... 12
    2.2.2 Self-Organizing Maps (SOM) ................................ 12
List of figures

Figure 1: The basic neuron ................................................. 5
Figure 2: Activation functions ............................................ 6
Figure 3: Multi-layer perceptron ......................................... 7
Figure 4: Representation of an self-organising maps (SOM) .......... 13
Figure 5: AutoCorrel II model layout ................................... 14
Figure 6: AutoCorrel I model ................................................ 15
Figure 7: The performance of Autocorrel I system [2], varying the cluster barrier parameter .............................................. 23
Figure 8: Experiments to show how scaling method and clustering barriers were chosen .................................................... 24
Figure 9: Experiments to show how the Gaussian scaling method compares with the linear scaling methods .............................. 25
Figure 10: A Waikato Environment for Knowledge Analysis (Weka) screen shot showing the alert instance on the $y$-axis and the clusters on the $x$-axis ....................................................... 29
This page intentionally left blank.
1 Introduction

In our earlier work, Autocorrel I [1,2], we implemented an event correlation approach using an artificial neural network (ANN). The purpose of that work was to correlate network events in an effort to assist an intrusion detection system (IDS) analyst. In this work, we build on the success of Autocorrel I, and attempt to address its shortcomings. We also hope to be able to address the research recommendations made in Autocorrel I.

1.1 Background

Intrusion detection analysts can be overwhelmed by the multitude 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. It would be beneficial to the IDS analyst if similar alerts were clustered together. 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.

There are a number of IDSs that can perform correlation using different correlation techniques. However, most of the correlation tools use only limited attributes within the alert to perform the correlation, and are often unable to produce accurate results. As a result, there have been significant research efforts to design correlation tools that can improve on these existing approaches. In this work, we explore a machine-learning approach to network event correlation that builds and improves on our previous model, Autocorrel I.

In this section, we review existing approaches to event correlation. We then present our approach and show how it is related to the existing approaches.

1.1.1 Previous Work

The majority of the existing correlation tools use elementary approaches to correlate attacks. For example, Shadow [3] and ACID [4] use the internet protocol (IP) addresses to correlate attacks. However, it is known that IP addresses may be spoofed, therefore using IP addresses alone may not be sufficient to identify correlated alerts.

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 attribute at a time to correlate with other possible attacks. More sophisticated approaches use statistical methods on multiple alert attributes [6]. Statistical approaches usually make assumptions about the underlying statistical distributions of alert attributes. ANNs, on the other hand, do not make prior assumptions about the data they handle [7,8].
In their recent work, Hätälä et al. [9] also analyse various alert correlation efforts by different groups. They give the details of a number of correlation systems, most of which are not based on machine learning. They also discuss the Intrusion Detection Message Exchange Format (IDMEF) standard proposed by the Intrusion Detection Working Group (IDWG) [10] which hopes to encourage the development and interoperability of alert correlation in deployed intrusion detection systems.

Ning and Cui [8] recognise that many existing IDSs tend to detect low-level events without being able to relate these events to the broader plan of the attacker. Their goal was to create hierarchies of alerts using prerequisites and consequences for each type of alert. They attained this goal using a rule-based system (rather than a machine-learning system) that assigned prerequisites and consequences to each type of alert, which in turn allowed the analyst to see the possible consequences of the most mundane of alerts. Their system ran on the output of existing IDS systems. They demonstrated the usefulness of their system by showing how their system could significantly reduce the number of false alarms reported by an IDS while negligibly reducing the valid alarms. Their tool is most logically used as an off-line forensic tool for mining old stored alerts after a new vulnerability is found.

Debar and Wespi [11] presented an aggregation and correlation component (ACC) for an IDS. They used this correlation component to flag alerts as an original, a consequence, or a double alert. In this research, they presented rule-based methods for attaining this goal. They concerned themselves with implementation issues such as integrating their ACC with existing IDSs – namely, the Tivoli Enterprise Console – and they adopted the use of the IDWG [10] format for their work. They discussed other issues such as raising or lowering the priority of IDS alerts to reflect the inferences of their ACC.

Julisch et al. used a conceptual clustering algorithm in their model [12]. They resolved the problem of training their system by relying on conceptual clustering algorithms rather than classification algorithms which require training. However, they require to train the system as to which alerts can be handled automatically. 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 [13] also presented another machine-learning-based correlation system. In their paper, they 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. [9] criticise this work for its use of a simplistic data set. The Dain et al. research was only tested with a Defence Conference (DEFCON) data
The use of this data set 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. [13]).

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 [15] 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 with which to correlate.

Our previous work, Autocorrel I [1], attempted to address many of these shortcomings. We implemented an ANN-based event correlation engine that took most alert attributes into account. However, the results produced by the Novelty Detection algorithm [16] used in this work produced systematic errors. The main source of this error was attributed to mapping a 42-dimensional alert attribute space into a 1-dimensional clustering metric. It required manual human intervention to improve its alert clustering accuracy to 76%.

1.2 Our Approach

The objective of this work was to build on the success we achieved in Autocorrel I [1] and improve the performance of the model while trying to achieve our original correlation goal. We implemented a two-stage alert correlation approach involving an ANN autoassociator and another machine-learning model, the Expectation–Maximisation (EM) algorithm.

The first step takes a set of alerts, constructs the first step alert attributes, trains an autoassociator, and outputs alert clusters. Based on the reconstruction error determined by the autoassociator, closely matched alerts are grouped together to form this first set of clusters. The reconstruction error from the first stage and attributes from raw alerts in the first stage clusters are processed by the second stage using the EM algorithm. The result is a new set of clusters which include clusters from the first stage.

Adding a second stage of clustering effectively mimics Valdes and Skinner’s approach [6] of finding attack step correlations after the threading and multi-sensor correlations have been done. However, unlike approaches by Dain et al. [13] and Julisch et al [12], which used supervised data mining approaches, we used an unsupervised machine-learning approach.

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 are known to be fast and quite good 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 transmission control protocol (TCP) Quad format [17] and performing a multi-stage query on insufficient information, the user can input all the relevant information available and allow the neural network to organise and reduce this knowledge in an adaptive and goal-oriented fashion. We also explored a number of clustering algorithms to determine the best algorithm for the second stage. We concluded, through experimentation, that the EM algorithm would be the best for this work.

We demonstrate this approach by applying the system to the labelled 1999 DARPA IDS evaluation data set [18] as well as unlabelled www.incidents.org alerts. We use Snort [19] to extract alerts from the supplied raw TCPDump [20] format data.

The layout of our report starts with an overview of ANNs and clustering algorithms in Section 2. In Section 3, we present the complete model used, and the data modelling process. The experimentation and results are presented in Section 4. We follow this with our conclusions and discussions in Section 5.
2 Artificial Neural Networks (ANN)s and Clustering Algorithms

In this section we review the theory of ANNs. We also look at clustering algorithms, and explain how they were used in this work.

2.1 Artificial Neural Networks

ANN models attempt to emulate the human brain through the dense interconnection of simple computational elements called neurons [21]. 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 nonlinear optimisation algorithms.

2.1.1 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 a highly simplified model 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 processing node through the synaptic connections, which are represented by connection strengths called weights $w_i$. A bias term $\theta$ is also used at each node. 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.

$$o = f \left( \sum_{i=0}^{N-1} w_i u_i - \theta \right)$$

*Figure 1: The basic neuron*

Although ANNs are constructed using this fundamental building block, there are significant differences in the architectures and driving fundamentals behind each ANN model.
2.1.2 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:

\[
o = f \left( w^T u \right)
\]  

where \( w \) is the weight vector defined as

\[
w \equiv \left[ w_1 \ w_2 \ \cdots \ w_N \right]^T
\]

and the input vector \( u \) is defined as

\[
u \equiv \left[ u_1 \ u_2 \ \cdots \ u_N \right]^T
\]

It should be noted that in ANN implementations, \( u_N = -1 \) and \( \theta = w_N \).

There are many different types of activation functions \( f \) to choose from, depending on the application [21, 22, 23]. 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 smooth continuous activation function is the sigmoid.

\[
f_h(x) \quad f_t(x) \quad f_s(x)
\]

Hard Limiter  Threshold Logic  Sigmoid

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

\[\text{Figure 2: Activation functions}\]

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:

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

\[\text{(2)}\]
and the unipolar sigmoidal activation function is defined as

\[ f(x) \equiv \frac{1}{1 + \exp^{-\lambda x}} \quad (3) \]

where \( \lambda \) is a constant.

The basic building block of an ANN is a single node based on the neuron model shown in Figure 1. It is called a perceptron after the work of Rosenblatt [21]. A perceptron can consist of one or more neurons in one layer. 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 \( f(x) = ax + b \).

### 2.1.3 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 (MLP). 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.

![Multi-layer perceptron](image)

**Figure 3:** Multi-layer perceptron.

The objective of using an MLP 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 [23] 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 [23]:

- 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 (uni-directional), 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. The problem is divided and solved in more than one step.
- 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 connection. These take the following two forms:

- In fully- or partially-connected recurrent networks, neurons within a layer communicate their outputs to neurons within the same layer. This is done a number of times before they are allowed to send their outputs to another layer.
- In on-center/off-surround ANNs, a neuron in one layer has an excitatory connection to itself and its neighbors, and has inhibitory connections to other neurons in the same layer.

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

Similar to the brain, ANNs learn from experience. ANNs are trained by adjusting the connection weights $w$ during each iteration until the desired output is produced at $o$ (see Equation 1). The learning ability of an ANN is determined by its architecture and by the algorithm chosen for training. The training methods [21] 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. 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, 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 [23].

2.1.5 Training Rules

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

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

Equation 4 is the ANN general learning rule [23]. 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 $\Delta w^k$ at each iteration $k$. Some of the common training rules are as follows:

- In the Hebbian rule [23, 22], the connection weight update $\Delta w^k$ is proportional to the neuron’s output. This was the first ANN learning rule [21, 24].

- The perceptron rule [21] updates the weights based on the difference between the desired output $d$ and the actual neuron’s response $o$. 
• The delta learning rule \([21, 24]\) is based on the minimisation of the mean square error (MSE):

\[ w^{k+1} = w^k - \eta \nabla E \left( w^k \right) \quad (5) \]

where \( \eta \) is a learning constant, and \( \nabla E \) is the gradient of the error function \( E \), defined by:

\[ E_k = \frac{1}{2} (d^k - o^k)^2 \quad (6) \]

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

• The Widrow-Hoff \([22, 23]\) 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 \( \Delta w^k \) 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, and \( I \) inputs. The output layer weights are adjusted as follows:

\[ w_{kj} = w_{kj} + \eta \delta_{ok} y_j, \quad \text{for } k = 1, \ldots, K, \ j = 1, \ldots, J \quad (7) \]

where \( \eta \) is a learning constant and the output error \( \delta_{ok} \) is given by

\[ \delta_{ok} = \frac{1}{2} (d_k - o_k)(1 - o_k^2), \quad \text{for } k = 1, 2, \ldots, K \quad (8) \]

The weight update for the hidden layer is as follows:

\[ w_{ji} = w_{ji} + \eta \delta_{yj} u_i, \quad \text{for } k = 1, \ldots, K, \ i = 1, \ldots, I \quad (9) \]

where the output error \( \delta_{yj} \) is given by

\[ \delta_{yj} = \frac{1}{2} (1 - y_j^2) \sum_{k=1}^{K} \delta_{ok} w_{kj}, \quad \text{for } j = 1, 2, \ldots, J \quad (10) \]

The process is iteratively repeated until a preset threshold of the mean square error (MSE).

For an ANN with \( P \) training patterns, and \( K \) outputs, the root-mean square error is defined as:

\[ E_{rms} = \frac{1}{P K} \sqrt{\sum_{p=1}^{P} \sum_{k=1}^{K} (d_{pk} - o_{pk})^2} \quad (11) \]

In most ANN implementations, this value is usually used to terminate training when a satisfactory value is reached. Reaching the maximum number of training epochs can also be used to terminate training.
2.1.6 The Autoassociator

The autoassociator is a fully-connected, multi-layer, feedforward ANN. The principle of autoassociation is very simple: in an unsupervised ANN training, inputs are made the targets. An autoassociator ANN thus tries to learn the identity function.

The architecture is similar to that of a MLP shown in Figure 3. The potential of the autoassociator to internally represent data trends often lies in the number of hidden units and design of the hidden layer, so these parameters should be carefully tailored to the problem. The input and output layers are of equal size.

From Figure 3, the reconstruction error $\epsilon$ for a given autoassociator with $M$ inputs is given by the Euclidian distance between input and output:

$$
\epsilon = ||\mathbf{u} - \mathbf{o}|| = \sqrt{\sum_{i=1}^{M} [o_i - u_i]^2}
$$

(12)

where $\mathbf{u}$ is the input vector and $\mathbf{o}$ is the output vector. In our work, this represents the reconstruction error (RE).

The autoassociator uses the reconstruction error $\epsilon$ to classify different inputs. The theory holds that data items from input data categories which did not have representative training examples will have higher reconstruction errors. Data items which are not similar are expected to have distinguishable reconstruction errors [25].

In this work, we use the autoassociator as an extension to the work by Japkowicz et al. [16,26] and other researchers such as Sungzoon et al. [27] on Novelty Detection. In these works, 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.

2.2 Clustering Algorithms

Like the autoassociator, clustering algorithms are machine-learning approaches used to categorise similar objects based on the objects’ attributes. Since our work involves alert clustering, in this section, we review the theory of the two clustering algorithms used in our model; namely the Expectation–Maximisation (EM) algorithm (Section 2.2.1) and self-organising maps (SOM) (Section 2.2.2). We also use the autoassociator (Section 2.1.6) as will be explained later.
2.2.1 The Expectation–Maximisation (EM) Algorithm

The EM algorithm is one of the most well-known clustering algorithms [28, 29]. It helps to find maximum-likelihood estimators in problems with incomplete or missing data. Initially presented in Dempster et al. [30], it consists of two repeated steps: expectation and maximisation. The EM algorithm uses a statistical model called finite mixtures to achieve the goal of producing the most likely set of clusters given the number of clusters, \( k \), and a set of data [31].

The model consists of a set of \( k \) probability distributions, one to represent the data of each cluster. There are parameters that define each of the \( k \) distributions. The EM algorithm begins by making initial guesses for these parameters based on the input data, then, for each data instance, determines the probability that it belongs to a particular cluster using these parameter guesses. Based on the results, the distribution parameters are revised again and this process is repeated until the resulting clusters satisfy some predefined conditions or until a maximum number of algorithm iterations is reached.

In this work, we utilise the EM algorithm as implemented in the Waikato Environment for Knowledge Analysis (Weka). Weka [32] is a tool developed at the University of Waikato in New Zealand.

2.2.2 Self-Organizing Maps (SOM)

The SOM is an ANN algorithm developed by Teuvo Kohonen [33, 34], though some prefer to refer to it as a specific type of clustering algorithm [28]. It is an unsupervised method of mapping a high-dimensional input to a two-dimensional output such that similar inputs are mapped to close locations in the two-dimensional map (grid or lattice) output as illustrated in Figure 4 [21]. (Actually, other dimensionalities are available for output as well, but two-dimensional output is the most common.) This mapping is primarily used for the visualisation of high-dimensional data, but it can also be used for clustering.

Like other ANNs, the SOM has training parameters and is trained iteratively. The first step is to initialise the weights from \( N \) inputs to the \( M \) output nodes to small random numbers, and define an initial neighborhood radius for each node. The neighborhood \( d \) is the set of nodes which are similar to a given node. As the training progresses, the number of nodes in the neighborhood shrinks until only nodes representing similar inputs are left in the neighborhood. Thus, \( M > d(t_1) > d(t_2) \ldots \), where time \( t_1 < t_2 \ldots \).

As illustrated in Figure 4, each input \( x_i \) is connected to every output node \( j \) through a weight \( w_{ij} \). At each step, an input vector \( \mathbf{x} \) is presented to the SOM. The distance
between $\mathbf{x}$ and the SOM weight vectors is calculated, and the neuron whose weight vector $\mathbf{w}_c$ is closest to the input vector $\mathbf{x}$ is called the best-matching unit (BMU) as defined by the following equation:

$$||\mathbf{x} - \mathbf{w}_c|| = \min_i ||\mathbf{x} - \mathbf{w}_i|| \quad \text{for } i = 1, \ldots, d$$

(13)

where $||\cdot||$ is the Euclidian distance similar to Equation 12.

Whenever an example $\mathbf{x}$ is presented to the SOM, the closest representative weight vector $\mathbf{w}_j$ is found. The weight update at each iteration is as follows:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha [\mathbf{x} - \mathbf{w}_i] \quad \text{for all } i \text{ in the spatial neighbourhood of } j$$

(14)

The constant $\alpha$ and the neighbourhood are allowed to change. Vesanto et al. [35] also discuss other methods of clustering the SOM. In other related work, Vesanto et al. [36] developed the SOM Toolbox for use with MATLAB.

In our implementation the SOM is trained by initially giving the training data their approximate position on the grid and initialising the weights to small random numbers. Data is presented to the SOM and the corresponding output determined. For SOMs the output for an input data instance represents a point in the SOM. Similar outputs in the SOM should have similar representation in the higher dimensional input space. To cluster the SOM output we create one cluster for every point in the SOM output that has at least one data instance associated with it. In practice this produces acceptable clusters.
3 The Alert Correlation Model

Our model of finding correlations was implemented in two steps. The first step clustered input alerts. The second step was designed to take these clusters, and re-cluster them into super-clusters. The steps are summarised in Figure 5.

![Figure 5: AutoCorrel II model layout.](image)

The model takes alerts as input. Attributes are extracted from the alerts and fed into the first correlation step, Stage 1. This stage is very similar to Autocorrel I, in our previous work [1]. Stage 2 takes in the output from Stage 1 as well as original attributes extracted from the alerts. The processing in this stage produces the final alert clusters.

We will describe each of the stages in more detail. We will also give details of the data that was used in this work.

3.1 First Correlation Step (Stage 1)

In the first stage, our model collects alerts from input sensors. These sensors could be homogeneous or heterogeneous. However, we expect that all the sensors pass in their alerts as packets or sequences of packets that can be converted into Snort alerts. The alert packets are passed to Snort which then generates the relevant alerts. Using the same approach we used with Autocorrel I [1, 2], we extracted selected features from these alerts. This set of features was created to fully represent an IP packet flagged as an alert by an IDS, in our case by a Snort IDS. The feature set allows similar-looking IP packets to be clustered together by our algorithm.

In the first stage, we used the autoassociator for clustering. We used this ANN because we want to build on its performance from our previous work. We hope to improve the results of this model by adding a second stage.

As illustrated in Figure 6, we input the alerts data into an autoassociator. We then train it to reproduce the same alerts at the output. When training is complete, alerts are presented to the autoassociator. The reconstruction error (RE) between the input alert vector \( x_i \) and the autoassociator output alert vector \( y_i \) is determined for each
alert as follows:

\[ e_i = ||x_i - y_i|| \]  

(15)

Using a predetermined threshold value, alerts with similar REs are grouped together as clusters. These clusters are the final result of stage 1, and they become an input to stage 2. Equation 15 essentially reduced a 42-dimensional vector into a 1-dimensional threshold metric \( e(\text{RE}) \) used for clustering (this was identified as the source of errors in Autocorrel I.). The result of the clustering was multiple alert groups that each have similar attributes.

Unlike other clustering algorithms like the EM algorithm [30] or SOMs [33], the autoassociator does not require knowledge of the number of output clusters in advance. It determines that through a self-optimisation process in the training algorithm. This is the main reason why we used the autoassociator in the first stage.

### 3.1.1 Feature Extraction

We extracted variables’ information from the Snort alert. We did not use any information specific to Snort, so our selection of features can represent the alerts generated by any IDS. For instance, we did not use the Snort attack priority level or Snort’s labelling of the attack because this information cannot be found in the raw IP packet.

The features extracted consist 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. In the end, we had 42 features that we passed to the autoassociator for classification. These features are listed in Table 1. Some attributes of the alerts were dropped for various reasons; for 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 IP address attribute could dominate the alert clustering, when in fact we know that different alerts may originate from the same IP address. Early trials of our model also showed that the exclusion had no
impact on the results and the remaining attributes were able to provide the required classification. 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 alert based on the remaining input variables. However, it should be noted that we did use IP source and destination addresses in the second correlation step.

3.1.2 Scaling Algorithms

ANN nodes perform nonlinear calculations which would result in loss of accuracy when the training data ranges are of significantly different magnitudes. It is therefore desirable to scale the input and output patterns to some smaller range [37]. The scaling described here is applicable to both stages of this work. We scaled the data using the common method of determining a linear map from the training data, then applying it to both the training data and the testing and evaluation data [38].

Specifically, for every feature in the data-set we determined the range of values for that set. For a given feature \( x_i \), we called the highest value \( high \) and the lowest value \( low \). From these values we computed the linearly scaled value for \( x_i \) of a data item \( d_i \) as \((d_i - low)/(high - low)\), resulting in a linear scaling in the interval \([0, 1]\).

For IP addresses, for example, we know that the addresses are 32 bits by definition of the IPv4 header, so the values of the addresses are always in the range of encoded values \([0, 2^{32} - 1] \) [39]. We can assign \( low = 0 \) and \( high = 2^{32} - 1 \) and remove the step for determination of \( high \) and \( low \) altogether when running the scaling algorithm, leaving the system with more predictable sets of values produced.

An alternative to this linear scaling is to do a Gaussian scaling of the data. To
compute the Gaussian scaling for a given feature $x_i$ on a data item $d_i$, we first have to determine the mean $\mu$ and the standard deviation $\sqrt{\omega}$. We then use these values to compute the new value for $d_i$ as $(d_i - \mu)/\sqrt{\omega}$, where the formula includes the unscaled values for $d_i$. This gives the training data a new mean of 0 and a new standard deviation of 1. The values of $\mu$ and $\sqrt{\omega}$ from the training set are used to scale the evaluation and testing sets.

### 3.1.3 Training the Autoassociator

An input vector $x_i$, made up of alert TCP/IP information extracted as described in Section 3.1.1, is fed through an autoassociator through weights $w_i$. The objective is for the autoassociator to reproduce the vector $x_i$. However, the reproduction is not perfect, and a reconstruction error is a measure of this imperfection. Similar attack scenarios have similar attributes, and we expect their reconstruction errors to be very close. Similar alerts are grouped together based on the reconstruction error. This idea is an extension of the work by Japkowicz et al. [16, 26] 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.

Based on the Novelty Detection approach by Japkowicz et al [25], a three layer, fully connected feedforward ANN with $N$ input nodes, $N$ output nodes and $J$ hidden nodes was used. The output at node $i$ of each layer is given by:

$$ y_i = f(\sum_{k=1}^{K_i} w_{ik} y_k) $$  \hspace{1cm} (16)

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 at the output. This algorithm has been successfully used in other intrusion detection research [40], and this is also the training algorithm used by Japkowicz et al. [25] in their Novelty Detection work. The training objective is to minimise the error function $E$, as given by:

$$ E = \sum_{i=1}^{n} ||x_i - y_i||^2 $$  \hspace{1cm} (17)

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 RE $e_i$ given by:

$$ e_i = ||x_i - y_i|| $$  \hspace{1cm} (18)
for each input vector $\mathbf{x}_i$ during ANN recall. This is illustrated in Fig. 6. Equation 18 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 similar attributes.

This heuristic is formally known as a single-link clustering algorithm [41] and it is applied to one-dimensional data — namely the reconstruction errors of the data. The clustering algorithm is known as a single-link algorithm because if the distance between any two data in a set is less than the threshold then those two data will appear in the same cluster.

3.2 Second Correlation Step

For this second correlation step shown in Figure 5, we took the output clusters of the previous stage and represented them each as new data items, each encoded using a new set of features. To facilitate the learning process, we constructed our features such that a cluster of alerts representing a single step of an attack would be similar to another cluster of alerts from the same attack.

By having a second step in our correlation system we hoped to be able to cluster each of the clusters produced by the first step to form super-clusters, or clusters of clusters of alerts—the first stage clusters are sub-clusters of the super-clusters. We did this because we hoped to link clusters from the previous step that are related, but that did not form one discrete cluster. Effectively, we hoped to be able to link different steps of an attack together by representing each step such that it can be correctly clustered with other steps from the same attack.

Our feature extraction and construction for this stage closely resembles work by other researchers in the area, in particular Dain and Cunningham [13]. We used these features in unsupervised training of clustering algorithms. The features are as follows:

1. **ipSrcAddrCommonPart**: Attackers often perpetrate an attack from a single host, or from a single IP subnet. This feature indicates similarity between the source IP addresses of two alerts.

2. **ipDestAddrCommonPart**: Attackers often target a single host or subnet in their attacks [13]. Thus, we constructed a feature indicating the similarity of two destination IP addresses.

3. **ipSrcAddrCommonBits**: Similar to the work by Dain and Cunningham [13], we use a measure of closeness between two IP addresses as one of our attributes. This is applied to the source and destination IP addresses of alerts. The closeness was defined as the maximal number of most-significant bits shared between
two encoded IP addresses (called \( r \)). Thus, if the binary \textbf{and} operation is used on two IP addresses \( IP_1 \) and \( IP_2 \), and the result yields \( b_1 b_1 \cdots b_{32} \), then
\[
    r = \sum_{i=1}^{32} b_i
\]

(19)

According to Dain and Cunningham, “An attack scenario may contain components with spoofed source IP addresses while other components of the attack may use the attacker’s real source IP.”

4. \texttt{avgTimeSig} and \texttt{varTimeSig}: This features indicates the similarity in time between two alerts. In our model we have features to indicate the average time when a group of alerts was generated, and the standard deviation (named \texttt{varTimeSig}) for the times when a group of alerts were generated.

5. \texttt{avgReconsErr}: This feature indicates the average reconstruction error of a group of alerts. As we’ve established in the previous reports [1, 2], the reconstruction errors of alerts tend to correspond directly with their type. This value is drawn from the autoassociator in the first stage.

We also experimented with some other features that other researchers like Dain and Cunningham didn’t consider. These are as follows:

1. \texttt{modePortSrc} and \texttt{modePortDest}: We found that using the TCP source and destination ports was a strong indicator in determining the grouping of a set of alerts. We also found that when the ports were the dominant variables, there was almost always a particular source or destination port that was more common than the rest. Therefore, we created two features: one to encode the most common TCP source port in a cluster of alerts, and one to encode the most common destination port.

2. \texttt{avgSeqNumDiff}: We also found that TCP sequence and acknowledgement numbers are sometimes good predictors of the use of a hacker tool. Often there are obvious patterns in a group of sequence or acknowledgement numbers, and this pattern can usually be identified by taking the difference of the two numbers.

\subsection{3.3 The Data Sources}

Our data was obtained in two ways. We downloaded Snort [19] alerts directly from \texttt{incidents.org} [42]. We also used labelled alerts from the 1999 Defense Advanced Research Projects Agency (DARPA) [18] IDS evaluation data. Since the labelled DARPA data was in packet form, we ran the data through Snort using the commonly used configuration filters. We developed some Perl [43] scripts to read the text-based Snort alerts into numeric data for use with our machine-learning algorithms.
In these data sets of alerts, we reserved 10,000 alerts for training and a separate set of 1,000 alerts for evaluation and testing. These data sets offer the IP traffic as seen by single or multiple homogeneous sensors. Alerts from heterogeneous sensors need to be handled slightly differently. We chose not to go into this at this time.

It should also be noted that in our work, we paid more attention to the accuracy of our results with the incidents.org data set because this represented a real alert data set. Alerts from incidents.org were found “in the wild” and represent real attack patterns as opposed to the DARPA data set which was obtained from a simulated network. As will be seen later, more interesting and meaningful correlations were achieved with the incidents.org data set.
4 Results Analysis

In this section, we present the implementation of our model. We start by presenting the performance evaluation metrics. This is followed by the experiments we carried out to select parameters and algorithms. Finally, we present sample results from the implementation of our approach on two data sets.

4.1 Performance Evaluation

In this work, we use a number of methods and metrics to evaluate the system’s performance. Like in the work by Dain and Cunningham [13], we use a manually constructed data set, the *gold standard*, to evaluate our system. We also define experimentation effectiveness measures to show the performance of the system relative to the *gold standard*.

4.1.1 Gold Standard for Performance

We manually created a *gold standard* to dictate the type of results we want to see produced automatically by our system. When creating the gold standard, we tried to be consistent to the notion that one *super-cluster* — a cluster produced by the second correlation step of our system: a cluster of clusters — should represent one attack attempt against the network. The gold standard is only for testing system performance. It is not required for the system to work.

We draw the gold standard from the data sets described earlier, i.e. from the incidents.org and the 1999 DARPA data sets. The 500 alerts used in the gold standard are the first half of the subset of 1000 alerts reserved for evaluation and testing from the start of the data set. These 500 alerts are manually organised into clusters based on our knowledge and experience.

4.1.2 Experimentation Effectiveness Measures

Since this is an experiment, errors are inevitable. To understand and analyse the outcome of our experiments, we have used basic performance measures to characterise the effectiveness of the system.

We counted the number of errors we made by counting the number of alerts which were correctly clustered versus the number of alerts which were incorrectly clustered for a sample of test data. We use the gold standard from Section 4.1.1 to determine which alerts are errors. The performance measures we used in this work are as follows:
1. **separation errors (SE)** occur when a distinct group of alerts should belong to a larger group of related alerts, but isn’t clustered with the larger group. For instance, if we have a cluster $A$ of 10 *nmap scan* alerts and a separate cluster $B$ of 3 *nmap scan* alerts that are related to cluster $A$, then we count 3 separation errors because the 3 alerts in $B$ should have been grouped with the 10 alerts in $A$. This implies that the analyst will have more work to do.

2. **clustering errors (CE)** occur when two unrelated alerts are clustered together. For instance, if we have a mixed cluster of 10 *nmap* scans and 5 *Telnet Access* alerts where the alerts of the two types in the cluster are obviously not related, we count 5 clustering errors. This type of error indicates the problem of the system finding correlation where there is none.

3. **total number of errors (TE)** represent the total number of alerts erroneously placed into clusters where they do not belong.

\[
TE = CE + SE
\]  

Obviously CEs are more serious errors than SEs because they may cause an IDS analyst to miss important events.

### 4.2 Experiments

In this section, we describe how we carried out experiments to determine the best attribute scaling model and cluster barrier size (the predetermined threshold described in Section 3.1) for the first stage as well as the best algorithm for the second stage. We also experimented on the parameters to most effectively train this model.

#### 4.2.1 Data Scaling and Barrier Selection

In our previous work, with Autocorrel I [2], we used a linear scaling algorithm to scale features to the $[0, 1]$ range. Varying the cluster barriers produced different error levels. In Figure 7, we show the errors produced in the Autocorrel I system as we varied the cluster barrier. This shows that the best cluster barrier to use would be 0.0017. This is very close to the value of 0.025 used in the Autocorrel I system [2].

In this work, we experimented with different data scaling approaches. Our objective was to have a rough idea as to which scaling data approach to use. At the same time, we also determined the best cluster barrier to use in our model. Figure 8 shows a number of experiments conducted to determine the best cluster barrier to use with the autoassociator in the first stage for a given scaling range.

In the scaling process, we either used the training data set’s lowest and highest values or we used predetermined attribute ranges. The predetermined attribute ranges were
The value of the cluster barrier parameter obtained from the theoretical knowledge of the possible ranges of the attribute values. For example, the TCP port numbers can only range from 0 to 65535, so the input attribute range of [0, 65535] was used to scale the data. In contrast, the training data set may only have values ranging from 20 to 22000, for example. We set the output of the scaling to be either [0, 1] or [−1, 1].

From the two scaling ranges, we chose [−1, 1]. Experiments shown in Figures 8(c) and 8(d) have lower error values than experiments shown in Figures 8(a) and 8(b). It is also important to note that the range [−1, 1] gives the ANN more input option ranges than the [0, 1] scale.

Between the scaling methods shown in Figures 8(c) and 8(d), we chose the predefined scaling range shown in Figure 8(d). Although experiments shown in figures 8(c) and 8(d) are nearly the same, the experiment shown on Figure 8(d) has the lowest error value. Figure 8(d) also shows the best clustering barrier of 0.0027—which is very close to what was recommended in Figure 8(c). Choosing experiments of Figure 8(d) over 8(c) also removes the possibility of handling input data lying outside the training data set boundaries. This would produce unpredictable errors.

We also performed an additional experiment using the Gaussian scaling method. This is shown in Figure 9. The results are not as good as the ones from experiments reported in figures 8(c) and 8(d). The error level is worse than the other cases. It

Figure 7: The performance of Autocorrel I system [2], varying the cluster barrier parameter.
The value of the cluster barrier parameter
Percentage errors
The performance with [0,1] scale and training set induced feature ranges
Best at 0.0015 (total errors: 18%)
(a) Scale in [0, 1] using training set to determine ranges.

The value of the cluster barrier parameter
Percentage errors
The performance with [0,1] scale and pre-defined feature ranges
Best at 0.0016 (total errors: 22.6%)
(b) Scale in [0, 1] using predefined attribute ranges.

The value of the cluster barrier parameter
Percentage errors
The performance with [-1,1] scale and training set induced feature ranges
Best at 0.0028 (total errors: 15.4%)
(c) Scale in [-1, 1] using training set to determine ranges.

The value of the cluster barrier parameter
Percentage errors
The performance with [-1,1] scale and pre-defined feature ranges
Best at 0.0027 (total errors: 15.2%)
(d) Scale in [-1, 1] using predefined attribute ranges.

Figure 8: Experiments to show how scaling method and clustering barriers were chosen.
The value of the cluster barrier parameter seems to be important for minimizing errors. The performance using Gaussian models of training set feature values is best at 0.0026 (total errors: 20.8%).

**Figure 9:** Experiments to show how the Gaussian scaling method compares with the linear scaling methods.

Since we were most concerned with the accuracy of our model, we decided to go with the method that produces the smallest error rate. We therefore chose the linear scaling in the interval \([-1, 1]\] with a cluster barrier of 0.0027.

### 4.2.2 Parameter and Algorithm Exploration

In this set of experiments, we show how we arrived at some of the training parameters that were used in this work. We varied two autoassociator parameters described in Section 3.1: training epochs and number of hidden nodes. We also examined three candidate clustering algorithms to determine which gave the best performance.

To compare the performance of the various modifications of our system, we took 100-consecutive alert windows from both the 1999 DARPA data set [18] and the incidents.org data set [42]. For easy error comparison, we took these 100 alerts from the gold standard that we created earlier.

This parameter exploration was intended to give guidance as to the parameters that
would give the best results in the given experimentation environment. As a result, we compared the output of our system against the gold standard for predetermined numbers of hidden nodes, namely 8, 16, 32, and 64. Other numbers tried but not reported include 20, 28, 40, 42 and 50. These were not reported because they did not produce significantly different results from the other four.

We also decided to predetermine the number of training epochs. As ANNs become more trained, they become more capable of recognising the particular data on which they are being trained. One notable pitfall of the training process is that if the neural network is trained for too many epochs, then overfitting may occur [38]. First, we trained an ANN until we achieved acceptable MSE values (initially set to the original cluster barrier value of 0.0025). Then, based on this successful training, we examined three candidate epoch values, namely 500, 2500, and 5000. In the SOM, we used a lattice (grid) map of 4 × 6.

We separately compared the results of clustering the evaluation data sets of 100 alerts each using a trained autoassociator and three different clustering algorithms: the single-link heuristic algorithm we developed for Autocorrel I, the EM algorithm, and self-organizing maps. In tables 2 to 4, we list the TE, CE, and SE.

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Hidden Nodes</th>
<th>Epochs=500</th>
<th>Epochs=2500</th>
<th>Epochs=5000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TE</td>
<td>CE</td>
<td>SE</td>
<td>TE</td>
</tr>
<tr>
<td>incidents.org</td>
<td>8</td>
<td>29</td>
<td>8</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>47</td>
<td>8</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>42</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>40</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>DARPA</td>
<td>8</td>
<td>52</td>
<td>6</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>52</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>51</td>
<td>4</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>48</td>
<td>0</td>
<td>48</td>
</tr>
</tbody>
</table>

As previously noted, in our evaluation of this data, we were more concerned about good results for the incidents.org data than for the DARPA data. It will also be noticed in our results that the clustering accuracy for the DARPA data is lower. We attribute this to fact that this is simulated attack data which does not represent the same variability and correlations as with real attack data found “in the wild”.

The results from tables 2 to 4 show that the lowest TE achieved was 28 from the SOM with 32 hidden nodes trained for 2500 and 5000 epochs. Since we regard CE as more

---

1An overfitted ANN is a neural network that has been trained for more epochs than it should have for optimal performance, and is unable to generalise.
Table 3: Results of the autoassociator and the EM algorithm

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Hidden Nodes</th>
<th>Epochs=500</th>
<th></th>
<th>Epochs=2500</th>
<th></th>
<th>Epochs=5000</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TE</td>
<td>CE</td>
<td>SE</td>
<td>TE</td>
<td>CE</td>
<td>SE</td>
<td>TE</td>
</tr>
<tr>
<td>incidents.org</td>
<td>8</td>
<td>32</td>
<td>4</td>
<td>28</td>
<td>39</td>
<td>36</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>44</td>
<td>0</td>
<td>44</td>
<td>44</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>39</td>
<td>2</td>
<td>37</td>
<td>40</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>39</td>
<td>2</td>
<td>37</td>
<td>43</td>
<td>1</td>
<td>42</td>
</tr>
<tr>
<td>DARPA</td>
<td>8</td>
<td>44</td>
<td>3</td>
<td>41</td>
<td>61</td>
<td>0</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>63</td>
<td>4</td>
<td>59</td>
<td>43</td>
<td>3</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>44</td>
<td>8</td>
<td>36</td>
<td>51</td>
<td>4</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>65</td>
<td>3</td>
<td>62</td>
<td>45</td>
<td>3</td>
<td>42</td>
</tr>
</tbody>
</table>

Table 4: Results of the autoassociator and the SOM

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Hidden Nodes</th>
<th>Epochs=500</th>
<th></th>
<th>Epochs=2500</th>
<th></th>
<th>Epochs=5000</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TE</td>
<td>CE</td>
<td>SE</td>
<td>TE</td>
<td>CE</td>
<td>SE</td>
<td>TE</td>
</tr>
<tr>
<td>incidents.org</td>
<td>8</td>
<td>34</td>
<td>6</td>
<td>28</td>
<td>33</td>
<td>4</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>33</td>
<td>4</td>
<td>29</td>
<td>33</td>
<td>4</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>29</td>
<td>4</td>
<td>25</td>
<td>28</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>35</td>
<td>4</td>
<td>31</td>
<td>30</td>
<td>4</td>
<td>26</td>
</tr>
<tr>
<td>DARPA</td>
<td>8</td>
<td>62</td>
<td>4</td>
<td>58</td>
<td>61</td>
<td>4</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>61</td>
<td>3</td>
<td>58</td>
<td>61</td>
<td>2</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>58</td>
<td>6</td>
<td>52</td>
<td>60</td>
<td>9</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>60</td>
<td>9</td>
<td>51</td>
<td>58</td>
<td>6</td>
<td>52</td>
</tr>
</tbody>
</table>
harmful than SE, we concluded that the best system performance in this experiment
would be attained by the SOM maps with an autoassociator with 32 hidden nodes trained for 2 500 epochs.

Table 5: Clustering algorithm average errors

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Clustering Algorithm</th>
<th>Average Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TE</td>
</tr>
<tr>
<td>incidents.org</td>
<td>Single-link</td>
<td>40.9 (28.8)</td>
</tr>
<tr>
<td></td>
<td>EM</td>
<td>40.9 (14.8)</td>
</tr>
<tr>
<td></td>
<td>SOM</td>
<td>31.5 (6.1)</td>
</tr>
<tr>
<td>DARPA</td>
<td>Single-link</td>
<td>50.5 (2.1)</td>
</tr>
<tr>
<td></td>
<td>EM</td>
<td>53.8 (89.3)</td>
</tr>
<tr>
<td></td>
<td>SOM</td>
<td>59.5 (2.5)</td>
</tr>
</tbody>
</table>

To further analyse the effectiveness of the three clustering algorithms, we considered
the averages of TE and CE for tables 2, 3, and 4. These averages are shown in Table 5. The values in italics are the variance measures of the total errors. While the SOM has the lowest TE, the EM algorithm produces the lowest average CE. The single-link algorithm preformed much better for an autoassociator with 32 or 64 hidden units. Since the CE are more critical than SE, we therefore decided to use the single-link algorithm (which is the same model as AutoCorrel I) in the first stage and use the EM algorithm in the second stage.²

4.3 Sample Results

The model was tested with unlabelled alerts from www.incidents.org as well as labeled alerts from the 1999 DARPA IDS evaluation set. In each case, 10 000 alerts were used for training.

For the first stage, an autoassociator with 42 inputs, 32 hidden layers and 42 outputs was trained, using the error backpropagation algorithm with a training constant $\eta$ of 0.4. Training was initially fixed to 500 epochs, but could be changed if desired. The data was then passed onto the EM algorithm in the second stage, and clusters were produced.

We loaded 500 alerts from the incidents.org data set first. Using the EM algorithm implemented in Weka as the second stage clustering algorithm, we were able to cluster the data into 18 clusters. In Figure 10, we show a screen-shot of the Weka window that produced these clusters. The clusters can be distinguished by the different colours of

²It should be noted that the only evaluation done for the first stage was the cluster barrier, number of training epochs and scaling range (which is also applicable to the second stage).
Figure 10: A Weka screen shot showing the alert instance on the y-axis and the clusters on the x-axis.
the individual alerts and the key at the bottom. Some of the colours are very similar, and therefore difficult to distinguish. In addition, Weka seems to merge data points that are similar into one point. That explains why the number of alert instances seems to be less than the 500 alerts that were tested.

To carry out the analysis in more detail, we list a sample of the incidents.org clustered alerts in Table 6. The rest of the clustered alerts are listed in Annex C. The sample is from the first of the 18 clusters produced. Within this cluster there are three sub-clusters, each of which was output from stage 1. The printout from our

Table 6: A sample of cluster 1 alerts produced by this model.

<table>
<thead>
<tr>
<th>Supercluster 1/18</th>
<th>Cluster 1/3 (sc: 1)</th>
</tr>
</thead>
</table>

model shows Supercluster 1/18, meaning that this is cluster 1 of 18 super-clusters. The next line shows Cluster 1/3 (sc:1), meaning that within this super-cluster this is cluster 1 of 3; sc:1 reminds us that we are still in super-cluster 1.

The results on Table 6 are an example of the Proxy Port Scan. A comparison with the gold standard shows that all these alerts belong to the same cluster. In this

---

3Unfortunately, Weka does not have the option to reconfigure these screen shots.
cluster, there are no alerts that do not belong to the Proxy Port Scan group; thus $CE = 0\%$.

A summary of the results is shown in Table 7. Detailed results of all the alert clusters summarised here are listed in Annex C. We compared the output of our system with the gold standard clusters. It should be noted that the clusters produced by our system are not in the same order as the gold standard clusters.

In Figure 7, the numbers in boldface represent the number of machine produced alerts which correspond to the gold standard. The boxed numbers represent the alerts that have been erroneously separated from the main clusters, but they individually occupy sub-clusters of their own.

Although some clusters do not match the gold standard, they constituted clusters of their own. So, if there were no dissimilar alerts in these clusters, we associated a $CE$
of 0% to the clusters. However, the errors associated with these clusters are reflected in the corresponding SE, since they are clearly separated from the main clusters of their type. Examples of these are clusters 1, 5, 8, and 15 as shown in Table 7.

In all cases, whenever there were two or more clusters representing the same gold standard attack type, we chose the cluster with the highest number of alerts to represent that cluster group. Thus, we chose cluster 17 to represent the Proxy Port Scan cluster over cluster 1. We also “separated” some clusters which had two or more significant alert groupings in sub-clusters of their own. For example cluster 16 was split up into two as shown in Table 7. A sample of these alerts is shown in Table 8. The cluster contains two sub-clusters of two distinct attack types. Although

**Table 8: Sample of cluster 16 alerts.**

```
Cluster1/2 (sc: 16)


Cluster2/2 (sc: 16)


```

they are clearly related (they are all targeting the same 170.129.x.y addresses), they represent two different modes of attacks and separating them this way would be beneficial to the analyst. On the other hand, the sub-cluster 1/2 could be the first stages of a denial of service (DoS) land attack represented by sub-cluster 2/2, in which case, separating them would provide the analyst with less information. In cases like these, it would be best to leave such decisions to the analyst.

We therefore concluded that from the 18 system-produced clusters, 9 matched the gold standard. However, the system failed to classify Tiny Fragments, P2P Gnutella,
and Low source port alerts into their own individual clusters. Although the failure proportion is low (25%), this is the type of error that an analyst would want to avoid. The alerts themselves are hidden inside other clusters, and an analyst can easily miss them as a result.

We can see that the CEs for the system-produced clusters range from 0 – 50%. The 50% in cluster 12 is not necessarily statistically significant since there are only two alerts in this cluster. Only 18 (3.6%) alerts were placed completely erroneously. However, if we consider all the alerts that did not fall into any of the 9 matched clusters, we have a total of 56 erroneously placed alerts. This corresponds to a TE of 11.2%.

Similarly, the separation errors produced for this data set range from 0 – 61.7%. The worst case is Proxy Port Scan cluster. However, in each one of the separated clusters, our system was able to distinguish the alerts as unique and put them into their own cluster (as in cluster 15 and 17) or sub-cluster (as in clusters 5 and 8). This can easily be fixed manually.

We also carried out a similar simulation with DARPA data. The results of this simulation are summarised in Table 9. In this case the CE varied from 0 – 33%. In this data set, the system managed to produce 11 clusters out of a possible 12. The only exception was that it also produced as a sub-cluster of cluster 16. This is the only case of a complete CE in this data set. However, most of the errors are SEs. There were a total of 66 SEs. This corresponds to a TE of 13.2%; this is the same TE error range as for the incidents.org data set. It should also be noted that all the SEs in this data set were also individual sub-clusters. Although these sub-clusters were small, their existence as sub-clusters or clusters is an aid to any human intervention that may be required.

In comparison, our system performed better on the labelled DARPA alerts than it did on the incidents.org alerts. It is also important to note that absence of significant CEs in the DARPA data set results is a major achievement of this approach which we would like to emphasise. Our algorithm was able to achieve its goals with an SE percentage of less than 14%.

Clearly our system performs better than our earlier model, AutoCorrel I. The improvements came from expanding the model to two stages. Our new system also found a way around the 42-1 mapping limitation of AutoCorrel I by using the reconstruction error and other original alert attributes in the final stage of the correlation. However, separation errors are still significant. The only immediate solution to that at the moment is to have human intervention associate these “lost” clusters with one another.
**Table 9**: System-produced clusters for the DARPA data set.

<table>
<thead>
<tr>
<th>System-Produced Clusters</th>
<th>Cluster Description</th>
<th>Gold Standard Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>telnet access</td>
<td>Apache dir disclosure</td>
</tr>
<tr>
<td>2</td>
<td>Apache dir disclosure</td>
<td>74</td>
</tr>
<tr>
<td>3</td>
<td>telnet access</td>
<td>Tiny Fragments</td>
</tr>
<tr>
<td>4</td>
<td>telnet login incorrect</td>
<td>telnet access</td>
</tr>
<tr>
<td>5</td>
<td>telnet access</td>
<td>SNMP access</td>
</tr>
<tr>
<td>6</td>
<td>telnet access</td>
<td>SNMP access</td>
</tr>
<tr>
<td>7</td>
<td>SNMP access</td>
<td>SNMP access</td>
</tr>
<tr>
<td>8</td>
<td>Web 403 forbidden</td>
<td>Web 403 forbidden</td>
</tr>
<tr>
<td>9</td>
<td>telnet access</td>
<td>telnet access</td>
</tr>
<tr>
<td>10</td>
<td>Tiny Fragments</td>
<td>Tiny Fragments</td>
</tr>
<tr>
<td>11</td>
<td>SNMP access</td>
<td>SNMP access</td>
</tr>
<tr>
<td>12</td>
<td>SNMP access</td>
<td>SNMP access</td>
</tr>
<tr>
<td>13</td>
<td>SNMP access</td>
<td>SNMP access</td>
</tr>
<tr>
<td>14</td>
<td>Web search.dll</td>
<td>Web search.dll</td>
</tr>
<tr>
<td>15</td>
<td>Web /doc access</td>
<td>Web search.dll</td>
</tr>
<tr>
<td>16</td>
<td>IIS register.asp</td>
<td>IIS register.asp</td>
</tr>
<tr>
<td>17</td>
<td>Web 403 forbidden</td>
<td>Web 403 forbidden</td>
</tr>
<tr>
<td>18</td>
<td>CGI calendar</td>
<td>CGI calendar</td>
</tr>
<tr>
<td>19</td>
<td>Web 403 forbidden</td>
<td>Web 403 forbidden</td>
</tr>
<tr>
<td>20</td>
<td>CGI redirect</td>
<td>CGI redirect</td>
</tr>
<tr>
<td>21</td>
<td>telnet access</td>
<td>telnet access</td>
</tr>
</tbody>
</table>

**Error (SE)**: 0% 29.5% 0% 0% 45.4% 75% 0% 0% 0% 100% 0% 0% 10%
5 Discussion and Conclusions

In this work, we successfully designed and implemented a two-stage alert correlation model using an autoassociator and the Expectation–Maximisation (EM) algorithm. We demonstrated how our model was able to cluster similar alerts with better accuracy than what we achieved with our earlier model, Autocorrel I. With this model, the number of alerts that an analyst has to deal with is significantly reduced.

Like Autocorrel I, our approach capitalises on the ability of artificial neural network (ANN)s to learn through self-optimisation based on given training examples. To improve the performance and errors encountered with Autocorrel I, we added a stage that implements another machine learning clustering algorithm. We reported our investigations into the use of different clustering techniques. We showed that for this type of work, the EM algorithm performs better than the self-organising maps (SOM) and the single-link clustering algorithms. Our results show that this new model architecture achieved much better performance than Autocorrel I—with a clustering accuracy of over 86%.

We successfully tested this approach on unlabelled alerts from incidents.org. With an accuracy of over 86%, we were able to cluster 500 alerts into 18 clusters. We also successfully tested the approach with a labelled data-set from the 1999 DARPA IDS evaluation data-set. In this case, we were able to accurately cluster 500 alerts into 21 clusters. The accuracy of the results was impressive in both cases. We achieved an accuracy of over 86% in each case. Although the number of clustering errors (CE)s is very low (less than 5% in each of the tested case), these are errors which can result in important missed calls by an analyst, so it would be good to find a way to reduce these errors even further. The number of separation errors (SE)s is very small as well, and very manageable through human intervention to ensure that the straying clusters and sub-clusters are associated with the right clusters.

5.1 Future Work

While no future work is currently planned in this area, there are a few things that could be worked on to improve what we have achieved here.

The results show that the addition of the second correlation stage significantly reduced the number of CEs. However, the same cannot be said about SEs. Although the total number of errors have significantly gone down, some additional thought needs to be put in to further reduce the SEs. It may be worthwhile to investigate using this approach with other non-machine learning methods. Perhaps adding a supervised machine training algorithm could also improve the performance.

The number of SEs, though small, means that human intervention is required to
ensure completely accurate clustering. Future work in this area should also investigate the possibility of reading the output from this approach and reorganising it into error-free clusters that could be used in day-to-day intrusion detection system (IDS) analyses. Further work may also include the coupling of Autocorrel II with good visualisation techniques to provide an easy-to-use tool for clustering alerts.

The current design of this model is such 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 packets through Snort first. Future work could expand AutoCorrel II to directly handle inputs from multiple heterogeneous sensors.
References


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


[12] 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.


[32] Weka 3: Data Mining Software in Java (online), 


[37] Hsu, Chih-Wei, Chang, Chih-Chung, and Lin, Chih-Jen (2003), A Practical Guide to Support Vector Classification, *National Taiwan University*.


## Annex A: Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>aggregation and correlation component</td>
</tr>
<tr>
<td>BMU</td>
<td>best-matching unit</td>
</tr>
<tr>
<td>ANN</td>
<td>artificial neural network</td>
</tr>
<tr>
<td>CE</td>
<td>clustering errors</td>
</tr>
<tr>
<td>CF</td>
<td>Canadian Forces</td>
</tr>
<tr>
<td>CFNOC</td>
<td>Canadian Forces Network Operations Centre</td>
</tr>
<tr>
<td>CIRT</td>
<td>Computer Incident Response Team</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defense Advanced Research Projects Agency</td>
</tr>
<tr>
<td>DEFCON</td>
<td>Defence Conference</td>
</tr>
<tr>
<td>DoS</td>
<td>denial of service</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation–Maximisation</td>
</tr>
<tr>
<td>IDMEF</td>
<td>Intrusion Detection Message Exchange Format</td>
</tr>
<tr>
<td>IDS</td>
<td>intrusion detection system</td>
</tr>
<tr>
<td>IDWG</td>
<td>Intrusion Detection Working Group</td>
</tr>
<tr>
<td>IP</td>
<td>internet protocol</td>
</tr>
<tr>
<td>MSE</td>
<td>mean square error</td>
</tr>
<tr>
<td>MLP</td>
<td>multi-layer perceptron</td>
</tr>
<tr>
<td>RE</td>
<td>reconstruction error</td>
</tr>
<tr>
<td>SE</td>
<td>separation errors</td>
</tr>
<tr>
<td>SOM</td>
<td>self-organising maps</td>
</tr>
<tr>
<td>TE</td>
<td>total number of errors</td>
</tr>
<tr>
<td>TCP</td>
<td>transmission control protocol</td>
</tr>
<tr>
<td>Weka</td>
<td>Waikato Environment for Knowledge Analysis</td>
</tr>
</tbody>
</table>
Annex B: Gold Standard Output

This section contains an edited version of the gold standard super-clusters for the incidents.org dataset [42]. This section is edited because it was too long (about 50 pages) to include here otherwise. It has been edited to give an overview of how the gold standard was created.

Supercluster 1/25

Cluster 1/6 (sc:1)

[**]

TCP Data Offset is less than 5! [**]

11/09-20:51:11.676507 62.13.27.29:0->207.166.33.145:0
TCP TTL:234 TOS:0x0 Id:0 DgmLen:40
Seq:0x81F9750 Ack:0x81F9750 Win:0x0 TcpLen:0

Cluster 2/6 (sc:1)

TCP Data Offset is less than 5! [**]

11/10-01:12:17.866507 172.20.10.199:0->207.166.119.62:0
TCP TTL:235 TOS:0x0 Id:0 DgmLen:40
Seq:0x82D2D468 Ack:0x82D2D468 Win:0x0 TcpLen:16

Cluster 3/6 (sc:1)

TCP Data Offset is less than 5! [**]

11/10-01:28:34.556507 62.13.27.29:0->207.166.78.44:0
TCP TTL:234 TOS:0x0 Id:0 DgmLen:40 DF
Seq:0x91D8C02 Ack:0x91D8C02 Win:0x0 TcpLen:12

Supercluster 2/25

Cluster 1/6 (sc:2)

[**]

TCP Data Offset is less than 5! [**]

11/12-18:38:17.846507 203.80.239.162:0->207.166.182.137:0
TCP TTL:107 TOS:0x0 Id:35119 DgmLen:48 DF
Seq:0x7930005 Ack:0xD80A04D1 Win:0x64BA TcpLen:0 UrgPtr:0x800

Cluster 2/6 (sc:2)

TCP Data Offset is less than 5! [**]

11/14-09:54:38.316507 170.129.50.120:63362->159.153.199.24:80
TCP TTL:125 TOS:0x0 Id:38908 DgmLen:436 DF
Seq:0x99AD8FC7 Ack:0x732FBFCD Win:0x4230 TcpLen:20

Cluster 3/6 (sc:2)

TCP Data Offset is less than 5! [**]

11/15-02:46:28.446507 170.129.50.120:64749->216.130.211.11:80
TCP TTL:124 TOS:0x0 Id:40697 DgmLen:1332 DF
Seq:0x1601F507 Ack:0x9AB18054 Win:0x43E1 TcpLen:20

Cluster 1/5 (sc:3)

BACKDOOR Q access [**]

TCP TTL:15 TOS:0x0 Id:0 DgmLen:43
Seq:0x0 Ack:0x0 Win:0x0 TcpLen:20
cluster 2/5 (sc: 3)

---------------------

[**][1:184:4] BACKDOOR Q access [**]
tcp ttl:14 tos:0x0 id:0 iplen:20 dgmlen:43
***a*r** seq:0x0 ack:0x0 win:0x0 tcplen:20

[**][1:184:4] BACKDOOR Q access [**]
tcp ttl:14 tos:0x0 id:0 iplen:20 dgmlen:43
***a*r** seq:0x0 ack:0x0 win:0x0 tcplen:20

...
Cluster 2/4 (sc:8)

---

**[1:628:3] SCAN map TCP [**]
TCP TTL:43 TOS:0x0 Id:16388 IpLen:20 DgmLen:40

...**A**** Seq:0x260 Ack:0x0 Win:0x578 TcpLen:20

**[1:628:3] SCAN map TCP [**]
TCP TTL:49 TOS:0x0 Id:9868 IpLen:20 DgmLen:40

...**A**** Seq:0x1FC Ack:0x0 Win:0x578 TcpLen:20

**[1:628:3] SCAN map TCP [**]
TCP TTL:49 TOS:0x0 Id:8842 IpLen:20 DgmLen:40

...**A**** Seq:0x134 Ack:0x0 Win:0x578 TcpLen:20

Cluster 1/1 (sc:15)

---

**[1:556:5] P2P Outbound GNUTella client request [**]
TCP TTL:123 TOS:0x0 Id:22720 IpLen:20 DgmLen:158 DF

...**AP**** Seq:0x5A0CA92C Ack:0x53A65413 Win:0x4038 TcpLen:20

**[1:556:5] P2P Outbound GNUTella client request [**]
11/14-15:43:37.316507 170.129.50.120:61122->24.65.114.32:6003
TCP TTL:123 TOS:0x0 Id:22766 IpLen:20 DgmLen:62 DF

...**AP**** Seq:0x5A129557 Ack:0x53A7A3BA Win:0x4038 TcpLen:20

Supercluster 16/25

---

Cluster 1/2 (sc:16)

---

**[1:620:6] SCAN Proxy Port 8080 attempt [**]
11/14-16:00:56.996507 66.159.18.66:43517->170.129.50.120:8080
TCP TTL:53 TOS:0x0 Id:59575 IpLen:20 DgmLen:60 DF

...S* Seq:0xBED8745 Ack:0x0 Win:0x16D0 TcpLen:40

TCP Options (5) => MSS:1460 SackOK TS:48656370 0 NOP WS:0

**[1:618:5] SCAN Squid Proxy attempt [**]
11/14-16:00:56.996507 66.159.18.66:43518->170.129.50.120:3128
TCP TTL:53 TOS:0x0 Id:50174 IpLen:20 DgmLen:60 DF

...S* Seq:0xBF3AC0C Ack:0x0 Win:0x16D0 TcpLen:40

TCP Options (5) => MSS:1460 SackOK TS:48656370 0 NOP WS:0

Cluster 2/2 (sc:16)

---

**[1:618:5] SCAN Squid Proxy attempt [**]
11/14-23:32:20.316507 66.159.18.49:55991->170.129.50.120:3128
TCP TTL:53 TOS:0x0 Id:16363 IpLen:20 DgmLen:60 DF

...S* Seq:0xB4E1FC5B Ack:0x0 Win:0x16D0 TcpLen:40

TCP Options (5) => MSS:1460 SackOK TS:51364794 0 NOP WS:0

Supercluster 21/25

---

Cluster 1/2 (sc:21)

---

**[1:1390:4] SHELLCODE x86 inc ebx NOOP [**]
11/14-16:10:30.806507 129.118.2.10:57425->170.129.50.120:63414
TCP TTL:51 TOS:0x0 Id:66986 IpLen:20 DgmLen:1420 DF

...**A**** Seq:0x8217BFFC Ack:0x90CF9E29 Win:0x16D0 TcpLen:20

**[1:648:6] SHELLCODE x86 NOOP [**]
11/14-21:55:36.566507 129.118.2.10:57425->170.129.50.120:63414
TCP TTL:51 TOS:0x0 Id:46490 IpLen:20 DgmLen:1420 DF

...**A**** Seq:0xA0742424 Ack:0x90CF9E29 Win:0x16D0 TcpLen:20

**[1:648:6] SHELLCODE x86 NOOP [**]
11/14-21:55:36.576507 129.118.2.10:57425->170.129.50.120:63414
TCP TTL:51 TOS:0x0 Id:46491 IpLen:20 DgmLen:1420 DF

...**A**** Seq:0xA074E7A4 Ack:0x90CF9E29 Win:0x16D0 TcpLen:20

Cluster 2/2 (sc:21)

---
BAD-TRAFFIC tcp port 0 traffic
11/5-07:36:26.406507 211.47.255.24:41866->170.129.195.40:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq:0xD8010CF5 Ack:0x0 Win:0x16D0 TcpLen:32
TCP Options (6) => MSS:1460 NOP NOP SackOK NOP WS:0

BAD-TRAFFIC tcp port 0 traffic
11/5-07:36:29.296507 211.47.255.24:41866->170.129.195.40:0
TCP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
******S* Seq:0xD8010CF5 Ack:0x0 Win:0x16D0 TcpLen:32
TCP Options (6) => MSS:1460 NOP NOP SackOK NOP WS:0

BAD-TRAFFIC same SRC/DST
IGMP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:28

BAD-TRAFFIC same SRC/DST
IGMP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:28

BAD-TRAFFIC same SRC/DST
TCP TTL:242 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset:0x0064 Frag Size:0x0014

BAD-TRAFFIC ip reserved bit set
TCP TTL:242 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB
Frag Offset:0x0064 Frag Size:0x0014

BAD-TRAFFIC ip reserved bit set
TCP TTL:242 TOS:0x0 ID:0 IpLen:20 DgmLen:40 RB

Supercluster 22/25
-------------------
Cluster 1/13 (sc:22)
-------------------

Supercluster 23/25
-------------------
Cluster 1/3 (sc:23)
-------------------

Supercluster 24/25
-------------------
Cluster 1/3 (sc:24)
-------------------

Supercluster 25/25
-------------------
Cluster 1/3 (sc:25)
-------------------
This page intentionally left blank.
Annex C: Recall Results

Supercluster 1/18

Cluster 1/3 (sc: 1)

ID:31456 Iplen:20 DgmLen:48 DF
******S* Seq: 0x1439203 Ack: 0x0 Win: 0x2000 TcpLen: 28 TCP Options (4) => MSS: 536 NOP NOP SackOK

ID:33248 Iplen:20 DgmLen:48 DF
******S* Seq: 0x1439203 Ack: 0x0 Win: 0x2000 TcpLen: 28 TCP Options (4) => MSS: 536 NOP NOP SackOK

ID:26758 Iplen:20 DgmLen:48 DF
******S* Seq: 0x13D84F7 Ack: 0x0 Win: 0x2000 TcpLen: 28 TCP Options (4) => MSS: 536 NOP NOP SackOK

Cluster 2/3 (sc: 1)

[**
206.48.61.152:3109 -> 170.129.132.70:8080 TCP TTL:113 TOS:0x0
ID:63760 DgmLen:20 DgmLen:48 DF
Seq: 0x211675 Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK
Cluster 3/3 (sc: 1)
-------------------
[**
[119:4:1] (http_inspect) BARE BYTE UNICODE ENCODING [**]
11/14-09:54:38.316507 170.129.50.120:63362 -> 170.129.50.120:8080 TCP TTL:125 TOS:0x0 ID:38908
Seq: 0x99AD8FC7 Ack: 0x732FBFCD Win: 0x4230 TcpLen: 20
[**
[119:4:1] (http_inspect) BARE BYTE UNICODE ENCODING [**]
11/14-09:54:47.286507 170.129.50.120:63387 -> 159.153.199.24:80 TCP TTL:125 TOS:0x0 ID:38984
Seq: 0x99CB8003 Ack: 0x733D8EE2 Win: 0x4230 TcpLen: 20
Cluster 3/3 (sc: 2)
-------------------
[**
[119:4:1] (http_inspect) BARE BYTE UNICODE ENCODING [**]
11/15-10:00:30.986507 170.129.50.120:64645 -> 199.45.45.132:80 TCP TTL:125 TOS:0x0 ID:1655
Seq: 0x68E6FA3B Ack: 0x43F74153 Win: 0x4470 TcpLen: 20
[**
[119:12:1] (http_inspect) APACHE WHITESPACE (TAB) [**]
11/15-10:00:30.986507 170.129.50.120:64645 -> 199.45.45.132:80 TCP TTL:125 TOS:0x0 ID:1655
Seq: 0x68E6FA3B Ack: 0x43F74153 Win: 0x4470 TcpLen: 20
Cluster 3/3 (sc: 2)
-------------------
[**
[119:4:1] (http_inspect) BARE BYTE UNICODE ENCODING [**]
11/15-05:01:26.376507 170.129.50.120:63188 -> 207.68.164.250:80 TCP TTL:125 TOS:0x0 ID:5826
Seq: 0xC9F1EEBB Ack: 0xF019024A Win: 0xFA61 TcpLen: 20
[**
[119:12:1] (http_inspect) APACHE WHITESPACE (TAB) [**]
11/15-05:01:26.376507 170.129.50.120:63188 -> 207.68.164.250:80 TCP TTL:125 TOS:0x0 ID:5826
Seq: 0xC9F1EEBB Ack: 0xF019024A Win: 0xFA61 TcpLen: 20
Cluster 3/3 (sc: 2)
-------------------
[**
[119:4:1] (http_inspect) BARE BYTE UNICODE ENCODING [**]
11/14-09:29:29.826507 255.255.255.255:31337 -> 170.129.172.186:515 TCP TTL:15 TOS:0x0 ID:0
Seq: 0x0 Win: 0x0 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
Cluster 1/1 (sc: 3)
-------------------
[**
[119:4:1] (http_inspect) BARE BYTE UNICODE ENCODING [**]
11/14-09:32:53.016507 255.255.255.255:31337 -> 170.129.132.79:515 TCP TTL:15 TOS:0x0 ID:0
Seq: 0x0 Win: 0x0 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
[**
[119:4:1] (http_inspect) BARE BYTE UNICODE ENCODING [**]
11/14-09:49:26.166507 255.255.255.255:31337 -> 170.129.188.515 TCP TTL:15 TOS:0x0 ID:0
Seq: 0x0 Win: 0x0 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
<table>
<thead>
<tr>
<th>Time Stamp</th>
<th>Source IP Address</th>
<th>Destination IP Address</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>TTL</th>
<th>TOS</th>
<th>ID</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/14-10:10:22.596507</td>
<td>255.255.255.255</td>
<td>170.129.195.178</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-10:11:44.836507</td>
<td>255.255.255.255</td>
<td>170.129.30.34</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-11:44:56.156507</td>
<td>255.255.255.255</td>
<td>170.129.137.174</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-12:56:26.746507</td>
<td>255.255.255.255</td>
<td>170.129.89.87</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-14:01:20.986507</td>
<td>255.255.255.255</td>
<td>170.129.23.133</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-14:04:08.966507</td>
<td>255.255.255.255</td>
<td>170.129.190.188</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-16:14:32.966507</td>
<td>255.255.255.255</td>
<td>170.129.192.22</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-16:16:56.986507</td>
<td>255.255.255.255</td>
<td>170.129.134.5</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-16:17:12.036507</td>
<td>255.255.255.255</td>
<td>170.129.156.132</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-16:30:57.086507</td>
<td>255.255.255.255</td>
<td>170.129.146.62</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-16:47:18.156507</td>
<td>255.255.255.255</td>
<td>170.129.1.102</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-17:06:30.606507</td>
<td>255.255.255.255</td>
<td>170.129.41.171</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
<tr>
<td>11/14-17:18:56.666507</td>
<td>255.255.255.255</td>
<td>170.129.80.5</td>
<td>31337</td>
<td>515</td>
<td>TCP</td>
<td>15</td>
<td>0x0</td>
<td>0x0</td>
<td>20</td>
</tr>
</tbody>
</table>


Supercluster 4/18
------------------

Clustet 1/2 (ac: 4)
------------------


[**] [116:46:1] (snort_decoder) WARNING: TCP Data Offset is less than 5! [**] [Classification: Potential Corporate Privacy Violation] [Priority: 1] 11/10-01:28:34.556507 62.13.27.29:0 -> 207.166.78.44:0 TCP TTL:234 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF


Cluster 2/2 (ac: 4)
------------------


**WARNING: TCP Data Offset is less than 5!**
**Classification: Potential Corporate Privacy Violation**

**Priority: 1**


**Seq: 0x24A1C4C Ack: 0x24A1C4C Win: 0x0 TcpLen: 0**

Supercluster 5/18

------------------
Cluster 1/1 (sc: 5)

------------------
Cluster 1/1 (sc: 5)

**[Classification: Attempted Information Leak]**

11/15-18:54:34.916507 204.188.170.90:4175 -> 170.129.152.15:8080 TCP TTL:113 TOS:0x0 ID:54756 IpLen:20 DgmLen:40

**Seq: 0x1CF3682 Ack: 0x0 Win: 0x2000 TcpLen: 28**

TCP Options (4) => MSS: 536 NOP NOP SackOK

**[Classification: Attempted Information Leak]**


**Seq: 0xB4F108EA Ack: 0x0 Win: 0x16D0 TcpLen: 40**

TCP Options (5) => MSS: 1460 SackOK TS: 51364783 0 NOP WS: 0

Supercluster 6/18

------------------
Cluster 1/1 (sc: 6)

------------------
Cluster 1/1 (sc: 6)

**[Classification: Executable code was detected]**


**Seq: 0xA075E500 Ack: 0x90CF9E29 Win: 0x16D0 TcpLen: 20**

[Xref => http://www.whitehats.com/info/IDS181]

**[Classification: Executable code was detected]**


**Seq: 0xA076B1D8 Ack: 0x90CF9E29 Win: 0x16D0 TcpLen: 20**

[Xref => http://www.whitehats.com/info/IDS181]

**[Classification: Executable code was detected]**


**Seq: 0xA07AE7F8 Ack: 0x90CF9E29 Win: 0x16D0 TcpLen: 20**

[Xref => http://www.whitehats.com/info/IDS181]

**[Classification: Bare Byte Unicode Encoding]**


**Seq: 0x9F68995 Ack: 0x6193010 Win: 0xFAF0 TcpLen: 20**

**[Classification: Bare Byte Unicode Encoding]**


**Seq: 0x9FEC995 Ack: 0x6193010 Win: 0xFAF0 TcpLen: 20**

[Xref => http://www.whitehats.com/info/IDS181]

**[Classification: Bare Byte Unicode Encoding]**


**Seq: 0xA014995 Ack: 0x6193010 Win: 0xFAF0 TcpLen: 20**

[Xref => http://www.whitehats.com/info/IDS181]

**[Classification: Bare Byte Unicode Encoding]**


**Seq: 0xA024995 Ack: 0x6193010 Win: 0xFAF0 TcpLen: 20**

[Xref => http://www.whitehats.com/info/IDS181]

**[Classification: Bare Byte Unicode Encoding]**


**Seq: 0xA034995 Ack: 0x6193010 Win: 0xFAF0 TcpLen: 20**

[Xref => http://www.whitehats.com/info/IDS181]
Supercluster 7/18
------------------
Cluster 1/1 (sc: 7)
-------------------

[**] [1:2314:1] SHELLCODE x86 0x90 NOOP unicode [**]  [Classification: Executable code was detected]  [Priority: 1] 11/15-12:09:01.196507 216.183.64.22:43124 -> 170.129.50.3:20 TCP TTL:235 TOS:0x0 ID:42286 IpLen:20 DgmLen:1500 DF

[**] [1:2314:1] SHELLCODE x86 0x90 NOOP unicode [**]  [Classification: Executable code was detected]  [Priority: 1] 11/15-12:09:01.256507 216.183.64.22:43124 -> 170.129.50.3:20 TCP TTL:235 TOS:0x0 ID:42290 IpLen:20 DgmLen:1500 DF

Cluster 2/2 (sc: 8)
-------------------


Cluster 1/2 (ac: 8)
-------------------


Cluster 2/2 (ac: 8)
-------------------

BAD-TRAFFIC tcp port 0 traffic

**Classification:** Misc activity [Priority: 3] 11/15-15:08:37.996507
211.47.255.23:46819 → 170.129.23.96:0 TCP TTL:4 TOS:0x0 ID:0
Iplen=20 DgmLen=52 DF
Seq: 0x03432FEE Ack: 0x00 Win: 0x1600 TcpLen: 32
TCP Options (6) → MS: 1460 NOP SACK NOP NOP WS: 0

**Classification:** Misc activity [Priority: 3] 11/16-07:36:46.156507
211.47.255.20:47040 → 170.129.118.169:0 TCP TTL:46 TOS:0x0 ID:0
Iplen=20 DgmLen=52 DF
Seq: 0x1800FAD Ack: 0x00 Win: 0x1600 TcpLen: 32
TCP Options (6) → MS: 1460 NOP SACK NOP NOP WS: 0

**Classification:** Misc activity [Priority: 3] 11/16-07:36:49.176507
211.47.255.20:47040 → 170.129.118.169:0 TCP TTL:46 TOS:0x0 ID:0
Iplen=20 DgmLen=52 DF
Seq: 0x1800FAD Ack: 0x00 Win: 0x1600 TcpLen: 32
TCP Options (6) → MS: 1460 NOP SACK NOP NOP WS: 0

**Classification:** Misc activity [Priority: 3] 11/16-07:37:08.116507
211.47.255.20:47040 → 170.129.118.169:0 TCP TTL:46 TOS:0x0 ID:0
Iplen=20 DgmLen=52 DF
Seq: 0x1800FAD Ack: 0x00 Win: 0x1600 TcpLen: 32
TCP Options (6) → MS: 1460 NOP SACK NOP NOP WS: 0

**Classification:** Misc activity [Priority: 3] 11/16-07:36:14.166507
211.47.255.23:46671 → 170.129.118.169:0 TCP TTL:4 TOS:0x0 ID:0
Iplen=20 DgmLen=52 DF
Seq: 0x1661E114 Ack: 0x00 Win: 0x1600 TcpLen: 32
TCP Options (6) → MS: 1460 NOP SACK NOP NOP WS: 0

**Classification:** Misc activity [Priority: 3] 11/16-07:36:23.166507
211.47.255.23:46671 → 170.129.118.169:0 TCP TTL:46 TOS:0x0 ID:0
Iplen=20 DgmLen=52 DF
Seq: 0x1661E114 Ack: 0x00 Win: 0x1600 TcpLen: 32
TCP Options (6) → MS: 1460 NOP SACK NOP NOP WS: 0

**Classification:** Misc activity [Priority: 3] 11/16-07:36:35.146507
60 DRDC Ottawa TM 2006-193
<table>
<thead>
<tr>
<th>IP Address</th>
<th>Destination IP Address</th>
<th>TCP Flags</th>
<th>Sequence</th>
<th>Acknowledgment</th>
<th>Window</th>
<th>TCP Length</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>211.47.255.20</td>
<td>170.129.118.169</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
<tr>
<td>211.47.255.21</td>
<td>170.129.48.125</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
<tr>
<td>211.47.255.21</td>
<td>170.129.48.125</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
<tr>
<td>211.47.255.21</td>
<td>170.129.48.125</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
<tr>
<td>211.47.255.21</td>
<td>170.129.48.125</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
<tr>
<td>211.47.255.20</td>
<td>170.129.118.169</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
<tr>
<td>211.47.255.21</td>
<td>170.129.48.125</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
<tr>
<td>211.47.255.21</td>
<td>170.129.48.125</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
<tr>
<td>211.47.255.21</td>
<td>170.129.48.125</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
<tr>
<td>211.47.255.21</td>
<td>170.129.48.125</td>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>MSS: 1460 NOP NOP SackOK NOP WS: 0</td>
</tr>
</tbody>
</table>

DRDC Ottawa TM 2006-193
211.47.255.21:41618 -> 170.129.156.144:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0x89778A1D Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:55518 -> 170.129.25.163:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0x5B030D9A Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:55518 -> 170.129.25.163:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0x5B030D9A Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:55518 -> 170.129.25.163:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0x5B030D9A Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:41358 -> 170.129.195.40:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xD4C4A857 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:41358 -> 170.129.195.40:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xD4C4A857 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:41358 -> 170.129.195.40:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xD4C4A857 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:55769 -> 170.129.25.163:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0x5D09DDB1 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:55769 -> 170.129.25.163:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0x5D09DDB1 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:55769 -> 170.129.25.163:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0x5D09DDB1 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:55769 -> 170.129.25.163:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0x5D09DDB1 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

211.47.255.24:41604 -> 170.129.195.40:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
211.47.255.21:40582 -> 170.129.156.144:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xB5097658 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
211.47.255.21:40582 -> 170.129.156.144:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xB5097658 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
211.47.255.21:40582 -> 170.129.156.144:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xB5097658 Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
211.47.255.24:56519 -> 170.129.23.90:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xC1B34FCA Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
211.47.255.24:56519 -> 170.129.23.90:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xC1B34FCA Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
211.47.255.24:56519 -> 170.129.23.90:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xC1B34FCA Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
211.47.255.24:56519 -> 170.129.23.90:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xC1B34FCA Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
211.47.255.24:57025 -> 170.129.23.90:0 TCP TTL:46 TOS:0x0 ID:0
IpLen:20 DgmLen:52 DF
****** Seq: 0xC371A50B Ack: 0x0 Win: 0x16D0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0
BAD-TRAFFIC tcp port 0 traffic (**) [Classification: Misc activity] [Priority: 3] 11/16-10:15:06.056507
Ip.len: 20 Dgm.len: 52 DF

BAD-TRAFFIC tcp port 0 traffic (**) [Classification: Misc activity] [Priority: 3] 11/16-10:18.076507
Ip.len: 20 Dgm.len: 52 DF

SHELLCODE x86 NOOP (**) [Classification: Executable code was detected] [Priority: 1] 11/14-21:55:39.356507

SHELLCODE x86 NOOP (**) [Classification: Executable code was detected] [Priority: 1]

DRDC Ottawa TM 2006-193
Executable code was detected [Priority: 1] 11/14-21:56:03.896507
129.118.2.10:57425 -> 170.129.50.120:63414 TCP TTL:51 TOS:0xA0
ID:48301 IpLen:20 DgmLen:1420 DF

[**] [1:1390:4] SHELLCODE x86 inc ebx NOOP [**] [Classification: Executable code was detected] [Priority: 1] 11/14-21:56:03.916507
129.118.2.10:57425 -> 170.129.50.120:63414 TCP TTL:51 TOS:0xA0
ID:48303 IpLen:20 DgmLen:1420 DF

[**] [1:1390:4] SHELLCODE x86 inc ebx NOOP [**] [Classification: Executable code was detected] [Priority: 1] 11/14-21:56:03.936507
129.118.2.10:57425 -> 170.129.50.120:63414 TCP TTL:51 TOS:0xA0
ID:48305 IpLen:20 DgmLen:1420 DF


Supercluster 11/18

------------------
Cluster 1/3 (sc: 11)

64.152.70.68:80 -> 170.129.50.120:53 TCP TTL:51 TOS:0x0 ID:33050 IpLen:20 DgmLen:40

64.152.70.68:53 -> 170.129.50.120:53 TCP TTL:51 TOS:0x0 ID:33051 IpLen:20 DgmLen:40

12.111.47.194:80 -> 170.129.50.120:2564 TCP TTL:48 TOS:0x0 ID:24950 IpLen:20 DgmLen:40

141.155.200.194:80 -> 170.129.50.120:2564 TCP TTL:45 TOS:0x0 ID:24956 IpLen:20 DgmLen:40


**Attempted Information Leak**

**Classification:** Attempted Information Leak

**Priority:** 2

**Timestamp:**
- 11/14-10:10:03.816507
- 11/14-12:52:04.916507
- 11/15-11:24:29.666507
- 11/15-11:47:53.986507
- 11/16-01:39:05.426507

**Source:**
- 61.218.161.202
- 61.218.161.210
- 208.155.15.101
- 206.102.126.101
- 206.102.126.101
- 61.218.161.210
- 61.218.161.202
- 61.218.161.210
- 61.218.161.202
- 208.155.15.101
- 206.102.126.101
- 61.218.161.210
- 61.218.161.202
- 61.218.161.210
- 61.218.161.202

**Target:**
- 170.129.19.170
- 170.129.14.62
- 170.129.50.122
- 170.129.50.122
- 170.129.14.62
- 170.129.14.62
- 170.129.14.62
- 170.129.14.62
- 170.129.14.62
- 170.129.14.62
- 170.129.14.62
- 170.129.14.62
- 170.129.14.62
- 170.129.14.62

**TCP Details:**
- Port 80
- Sequence numbers
- Acknowledgment numbers
- Window sizes
- Packet lengths
- Time-to-Live (TTL)
- Type of Service (TOS)

**References:**
- [IDS28](http://www.whitehats.com/info/IDS28)
Scan nmap TCP [**] [Classification: Attempted
-> 170.129.31:15:20 TCP TTL:44 TOS:0x0 ID:36247 IpLen:20 DgmLen:40
***A**** Seq: 0xFE Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Xref => http://www.whitehats.com/info/IDS28

Scan nmap TCP [**] [Classification: Attempted
-> 170.129.31:15:20 TCP TTL:44 TOS:0x0 ID:36504 IpLen:20 DgmLen:40
***A**** Seq: 0x26C Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
-> 170.129.31:15:20 TCP TTL:44 TOS:0x0 ID:36802 IpLen:20 DgmLen:40
***A**** Seq: 0x2EA Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/14-10:10:23:856507
163.23.236:9:80 -> 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]

Scan nmap TCP [**] [Classification: Attempted
-> 170.129.31:15:20 TCP TTL:44 TOS:0x0 ID:37072 IpLen:20 DgmLen:40
***A**** Seq: 0x35F Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/16-01:39:00.386507
IpLen:20 DgmLen:40
***A**** Seq: 0x1D4 Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/16-01:38:55.506507
IpLen:20 DgmLen:40
***A**** Seq: 0x170 Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/16-01:38:50.396507
IpLen:20 DgmLen:40
***A**** Seq: 0x10A Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/14-12:52:09.906507
163.23.236:9:80 -> 170.129.19:170:80 TCP TTL:44 TOS:0x0 ID:29544
IpLen:20 DgmLen:40
***A**** Seq: 0x2FD Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/15-11:48:06.816507
163.23.236:9:80 -> 170.129.152:80 TCP TTL:43 TOS:0x0 ID:29034
IpLen:20 DgmLen:40
***A**** Seq: 0x35B Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/14-12:52:09.906507
163.23.236:9:80 -> 170.129.19:170:80 TCP TTL:44 TOS:0x0 ID:29544
IpLen:20 DgmLen:40
***A**** Seq: 0x2FD Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/15-11:48:06.816507
163.23.236:9:80 -> 170.129.152:80 TCP TTL:43 TOS:0x0 ID:29034
IpLen:20 DgmLen:40
***A**** Seq: 0x35B Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/14-12:52:09.906507
163.23.236:9:80 -> 170.129.19:170:80 TCP TTL:44 TOS:0x0 ID:29544
IpLen:20 DgmLen:40
***A**** Seq: 0x2FD Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/14-12:52:09.906507
163.23.236:9:80 -> 170.129.19:170:80 TCP TTL:44 TOS:0x0 ID:29544
IpLen:20 DgmLen:40
***A**** Seq: 0x2FD Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

Scan nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/15-11:48:06.816507
163.23.236:9:80 -> 170.129.152:80 TCP TTL:43 TOS:0x0 ID:29034
IpLen:20 DgmLen:40
***A**** Seq: 0x35B Ack: 0x0 Win: 0x578 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
Executable code was detected [Priority: 1] 11/14-21:56:03.856507
129.118.2.10:57425 -> 170.129.50.120:63414 TCP TTL:51 TOS:0x0
ID:48298 IpLen:20 DgmLen:1420 DF

***AP*** Seq: 0xA09AF480 Ack: 0x90CF9E29 Win: 0x16D0 TcpLen: 20

[*] [1:628:3] SCAN nmap TCP [**] [Classification: Attempted
-> 170.129.238.112:80 TCP TTL:45 TOS:0x0 ID:23584 IpLen:20 DgmLen:40
***A**** Seq: 0x3A6 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] 11/14-14:28:54.256507 202.29.28.1:80
-> 170.129.238.112:80 TCP TTL:45 TOS:0x0 ID:23204 IpLen:20 DgmLen:40
***A**** Seq: 0x2E6 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] 11/14-12:14:11.266507
61.222.14.98:80 -> 170.129.81.112:80 TCP TTL:49 TOS:0x0 ID:48192
IpLen:20 DgmLen:40

[*] [1:628:3] SCAN nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/14-12:14:16.266507
61.222.14.98:80 -> 170.129.81.112:80 TCP TTL:49 TOS:0x0 ID:48734
IpLen:20 DgmLen:40

[*] [1:628:3] SCAN nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/16-03:52:30.576507
163.22.229.253:80 -> 170.129.98.80:80 TCP TTL:44 TOS:0x0 ID:49726
IpLen:20 DgmLen:40

[*] [1:628:3] SCAN nmap TCP [**] [Classification: Attempted
61.222.192.98:80 -> 170.129.81.112:80 TCP TTL:49 TOS:0x0 ID:49268
IpLen:20 DgmLen:40

[*] [1:628:3] SCAN nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/16-03:52:35.565507
61.221.99.242:80 -> 170.129.98.80:80 TCP TTL:47 TOS:0x0 ID:49912
IpLen:20 DgmLen:40

[*] [1:628:3] SCAN nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/15-12:12:33.965507
61.222.192.98:80 -> 170.129.98.80:80 TCP TTL:49 TOS:0x0 ID:49820
IpLen:20 DgmLen:40

[*] [1:628:3] SCAN nmap TCP [**] [Classification: Attempted
Information Leak] [Priority: 2] 11/16-03:52:41.016507
163.22.229.253:80 -> 170.129.98.80:80 TCP TTL:44 TOS:0x0 ID:49726
IpLen:20 DgmLen:40


**[1]: 628:3 SCAN nmap TCP [**] [Classification: Attempted Information Leak] [Priority: 2] 11/14-20:33:54.676507 61.221.88.198:80 -> 170.129.69.49:80 TCP TTL:50 TOS:0x0 ID:10358 1pLayer:20 DgmLen:40


80

DRDC Ottawa TM 2006-193
Iplen:20 Dgmlen:40
***A**** Seq: 0x07 Ack: 0x0 Win: 0x578 Tcp len: 20
[Xref => http://www.whitehats.com/info/IDS28]

Iplen:20 Dgmlen:40
***A**** Seq: 0x08 Ack: 0x0 Win: 0x578 Tcp len: 20
[Xref => http://www.whitehats.com/info/IDS28]

Iplen:20 Dgmlen:40
***A**** Seq: 0x12D Ack: 0x0 Win: 0x578 Tcp len: 20
[Xref => http://www.whitehats.com/info/IDS28]

Iplen:20 Dgmlen:40
***A**** Seq: 0x18F Ack: 0x0 Win: 0x578 Tcp len: 20
[Xref => http://www.whitehats.com/info/IDS28]

Iplen:20 Dgmlen:40
***A**** Seq: 0x34 Ack: 0x0 Win: 0x578 Tcp len: 20
[Xref => http://www.whitehats.com/info/IDS28]

Iplen:20 Dgmlen:40
***A**** Seq: 0x39F Ack: 0x0 Win: 0x578 Tcp len: 20
[Xref => http://www.whitehats.com/info/IDS28]

Iplen:20 Dgmlen:40
***A**** Seq: 0xA1 Ack: 0x0 Win: 0x578 Tcp len: 20
[Xref => http://www.whitehats.com/info/IDS28]

Iplen:20 Dgmlen:40
***A**** Seq: 0xA6 Ack: 0x0 Win: 0x578 Tcp len: 20
[Xref => http://www.whitehats.com/info/IDS28]


Cluster 3/3 (sc: 11)

-----------
Seq: 0x17A Ack: 0x0 Win: 0x578 TcpLen: 20 [Xref => http://www.whitehats.com/info/IDS28]

Seq: 0x1EC Ack: 0x0 Win: 0x578 TcpLen: 20 [Xref => http://www.whitehats.com/info/IDS28]

Seq: 0x2FE Ack: 0x0 Win: 0x578 TcpLen: 20 [Xref => http://www.whitehats.com/info/IDS28]

Supercluster 12/18

------------------
Cluster 1/1 (sc: 12)

------------------
Seq: 0x5502CA7 Ack: 0x0 Win: 0x578 TcpLen: 20 [Xref => http://www.whitehats.com/info/IDS07]

Seq: 0x95300006 Ack: 0x950A04D1 Win: 0x64BA TcpLen: 0 UrgPtr: 0x800

Supercluster 13/18

------------------
Cluster 1/1 (sc: 13)

------------------
[**] [1:1390:4] SHELLCODE x86 inc ebx MOP [**] [Classification: Executable code was detected] [Priority: 1] 11/14-16:10:30.806507 129.118.2.10:57425 -> 170.129.50.120:63414 TCP TTL:51 TOS:0xA0 ID:56986 Iplen:20 DgmLen:1420 DF
Seq: 0x8217BFFC Ack: 0x90CF9E29 Win: 0x16D0 TcpLen: 20

Supercluster 14/18

------------------
Cluster 1/1 (sc: 14)

------------------
Seq: 0x2A02EB38 Ack: 0x2A02EB38 Win: 0x0 TcpLen: 12

Supercluster 15/18

------------------
Cluster 1/2 (sc: 15)

------------------
[**] [1:615:5] SCAN SDCCS Proxy attempt [**] [Classification: Attempted Information Leak] [Priority: 2] 11/14-16:00:06.996507 66.159.18.66:43620 -> 170.129.50.120:1080 TCP TTL:53 TOS:0x0 ID:4253 Iplen:20 DgmLen:60 DF
Seq: 0x9DA7F8F Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 48656370 0 NOP WS: 0 [Xref => http://help.undernet.org/proxyscan/]

Cluster 2/2 (sc: 15)

------------------
[**] [1:615:5] SCAN SDCCS Proxy attempt [**] [Classification: Attempted Information Leak] [Priority: 2] 11/14-16:00:06.996507 66.159.18.66:43621 -> 170.129.50.120:1080 TCP TTL:53 TOS:0x0 ID:2662 Iplen:20 DgmLen:60 DF
Seq: 0xBF46594 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 48656370 0 NOP WS: 0 [Xref => http://help.undernet.org/proxyscan/]

Cluster 1/2 (sc: 15)

------------------
[**] [1:615:5] SCAN SDCCS Proxy attempt [**] [Classification: Attempted Information Leak] [Priority: 2] 11/14-16:00:06.996507 66.159.18.66:43622 -> 170.129.50.120:1080 TCP TTL:53 TOS:0x0 ID:2662 Iplen:20 DgmLen:60 DF
Seq: 0xBF46594 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 48656370 0 NOP WS: 0 [Xref => http://help.undernet.org/proxyscan/]

Cluster 1/2 (sc: 15)
Supercluster 16/18

Cluster 1/2 (sc: 16)


Cluster 2/2 (sc: 16)


170.129.215.110 -> 170.129.215.110 IGMP TTL:47 TOS:0x0 ID:0 IpLen:20

170.129.215.120 -> 170.129.215.120 IGMP TTL:47 TOS:0x0 ID:0 IpLen:20

170.129.215.126 -> 170.129.215.126 IGMP TTL:47 TOS:0x0 ID:0 IpLen:20

170.129.215.131 -> 170.129.215.131 IGMP TTL:47 TOS:0x0 ID:0 IpLen:20

170.129.215.137 -> 170.129.215.137 IGMP TTL:47 TOS:0x0 ID:0 IpLen:20

170.129.215.142 -> 170.129.215.142 IGMP TTL:47 TOS:0x0 ID:0 IpLen:20

170.129.215.85 -> 170.129.215.85 IGMP TTL:47 TOS:0x0 ID:0 IpLen:20

170.129.71.37 -> 170.129.71.37 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

170.129.71.42 -> 170.129.71.42 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

170.129.71.53 -> 170.129.71.53 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

170.129.71.47 -> 170.129.71.47 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

170.129.71.69 -> 170.129.71.69 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20
170.129.71.74 -> 170.129.71.74 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

170.129.71.63 -> 170.129.71.63 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

170.129.71.20 -> 170.129.71.20 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

170.129.71.7 -> 170.129.71.7 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.24 -> 170.129.71.24 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.9 -> 170.129.71.9 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.14 -> 170.129.71.14 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.30 -> 170.129.71.30 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.58 -> 170.129.71.58 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.31 -> 170.129.71.31 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.25 -> 170.129.71.25 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.30 -> 170.129.71.30 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.58 -> 170.129.71.58 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.25 -> 170.129.71.25 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.30 -> 170.129.71.30 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.58 -> 170.129.71.58 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.25 -> 170.129.71.25 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.71.30 -> 170.129.71.30 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20
BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.164.41 -> 170.129.164.41 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:28
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.164.52 -> 170.129.164.52 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:28
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.163.252 -> 170.129.163.252 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:28
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.164.20 -> 170.129.164.20 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:28
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.164.36 -> 170.129.164.36 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:28
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016

BAD-TRAFFIC same SRC/DST [Classification: Potentially Bad Traffic] [Priority: 2] 11/16-11:33:01.856507
170.129.164.46 -> 170.129.164.46 IGMP TTL:46 TOS:0x0 ID:0 IpLen:20 DgmLen:28
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0016

---

Cluster 1/2 (sc: 17)
---

SCAN Proxy Port 8080 attempt [Classification: Attempted Information Leak] [Priority: 2] 11/15-06:59:47.096507
Seq: 0x4E70998D Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 17303918 0 NOP WS: 0

SCAN Proxy Port 8080 attempt [Classification: Attempted Information Leak] [Priority: 2] 11/15-06:59:50.116507
Seq: 0x4E70998D Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 17304218 0 NOP WS: 0

SCAN Proxy Port 8080 attempt [Classification: Attempted Information Leak] [Priority: 2] 11/15-06:59:56.086507
Seq: 0x4E70998D Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 17304818 0 NOP WS: 0

SCAN Proxy Port 8080 attempt [Classification: Attempted Information Leak] [Priority: 2] 11/16-00:41:08.616507
204.188.170.90:1921 -> 170.129.237.240:8080 TCP TTL:13 TOS:0x0 IP:54971 Dgm:48 DF
Seq: 0x30307E Ack: 0x0 Win: 0x2000 TcpLen: 28
TCP Options (4) => MSS: 536 NOP NOP SackOK

SCAN Squid Proxy attempt [Classification: Attempted Information Leak] [Priority: 2] 11/14-16:00:56.996507
66.159.18.66:43818 -> 170.129.50.120:8080 TCP TTL:13 TOS:0x0
Cluster 2/2 (sc: 17)

---

Cluster 2/2 (sc: 18)

---

Cluster 1/2 (sc: 18)

---

Supercluster 18/18

-------------------

DRDC Ottawa TM 2006-193
Distribution list

DRDC Ottawa TM 2006-193

Internal distribution

4 Author
1 Section Display
4 Library electronic
1 electronic Marc Grégoire

Total internal copies: 10

External distribution

1 Assistant Deputy Minister S&T (ADM S&T) (for distribution)
1 Director S&T for Command and Control Information Systems (DSTCCIS) electronic
1 electronic Director R&D Knowledge and Information Management (DRDKIM-2)
1 Maj. Kent Schramm (NO&SCFNOC), CFS Leitrim, 101 Colonel By Dr., Ottawa, ON K1A 0Z4
1 CFNOC CIRT Team Leader, CFS Leitrim, 101 Colonel By Dr., Ottawa, ON K1A 0Z4
1 Ted Bennett (CSE), Manager, Cyber Defence Futures, Edward Drake Building, 1500 Bronson Ave. Ottawa, ON K1G 3Z4
1 Luc Dandurand (CSE) Cyber Defence Futures, Edward Drake Building, 1500 Bronson Ave. Ottawa, ON K1G 3Z4
1 David Whyte (CSE) SIGINT, Sir Leonard Tilley Building, 719 Heron Road, Ottawa, ON K1A 0K2
1 Paul Van Oorschot, Canada Research Chair (Network and Software Security), Carleton University, 1125 Colonel By Drive Ottawa, ON K1S 5B6
1 Carlisle Adams, SITE, University of Ottawa, 800 King Edward Avenue Ottawa, ON K1N 6N5
Total external copies: 13

Total copies: 23
<table>
<thead>
<tr>
<th>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.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defence R&amp;D Canada – Ottawa</td>
</tr>
<tr>
<td>3701 Carling Avenue, Ottawa, Ontario, Canada</td>
</tr>
<tr>
<td>K1A 0Z4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNCLASSIFIED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>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)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Event Correlation Using Unsupervised Machine Learning Algorithms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. AUTHORS (last name, first name, middle initial)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dondo, M.; Mason, P.; Japkowicz, N.; Smith, R.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5. DATE OF PUBLICATION (month and year of publication of document)</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2006</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6b. NO. OF REFS (total cited in document)</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>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)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical Memorandum</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defence R&amp;D Canada – Ottawa</td>
</tr>
<tr>
<td>3701 Carling Avenue, Ottawa, Ontario, Canada K1A 0Z4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9a. PROJECT NO. (the applicable research and development project number under which the document was written. Specify whether project)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15BF29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9b. GRANT OR CONTRACT NO. (if appropriate, the applicable number under which the document was written)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10a. ORIGINATOR’S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRDC Ottawa TM 2006-193</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10b. OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X) Unlimited distribution</td>
</tr>
<tr>
<td>( ) Defence departments and defence contractors; further distribution only as approved</td>
</tr>
<tr>
<td>( ) Defence departments and Canadian defence contractors; further distribution only as approved</td>
</tr>
<tr>
<td>( ) Government departments and agencies; further distribution only as approved</td>
</tr>
<tr>
<td>( ) Defence departments; further distribution only as approved</td>
</tr>
<tr>
<td>( ) Other (please specify):</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>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)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
We have successfully implemented a two-stage event correlation model for IDS alerts. The model is designed to automate alert and incidents management and reduce the workload on an IDS analyst. We achieve this correlation by clustering similar alerts together, thus allowing the analyst to only look at a few clusters instead of hundreds or thousands of alerts. The first stage of this model uses an ANN-based autoassociator. The autoassociator is trained to reproduce each alert at its output, and it uses the error metric between its input and output to cluster similar alerts together. The accuracy of the system is improved by adding another machine-learning stage which attempts to combine closely related clusters produced by the first stage into super-clusters. The second stage uses the EM clustering algorithm. The model and performance of this model are tested with intrusion alerts generated by a Snort IDS on DARPA's 1999 IDS evaluation data as well as incidents.org alerts.

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