A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguous and Discontiguous System Call Patterns

Gideon Creech, *Student Member, IEEE* and Jiankun Hu, *Member, IEEE*

Abstract—Host-based anomaly intrusion detection system design is very challenging due to the notoriously high false alarm rate. This paper introduces a new host-based anomaly intrusion detection methodology using discontiguous system call patterns, in an attempt to increase detection rates whilst reducing false alarm rates. The key concept is to apply a semantic structure to kernel level system calls in order to reflect intrinsic activities hidden in high-level programming languages, which can help understand program anomaly behaviour. Excellent results were demonstrated using a variety of decision engines, evaluating the KDD98 and UNM data sets, and a new, modern data set. The ADFA Linux data set was created as part of this research using a modern operating system and contemporary hacking methods, and is now publicly available. Furthermore, the new semantic method possesses an inherent resilience to mimicry attacks, and demonstrated a high level of portability between different operating system versions.

Index Terms—Intrusion detection, anomaly detection, computer security, system calls, host-based IDS, ADFA-LD

1 Introduction

Intrusion detection systems (IDS) represent an increasingly important component of computer security. The last 10 years have seen a marked increase in the number of intrusion events across the world, ranging from relatively benign one-off minor hacking activities through to international events such as the Stuxnet virus [1], [2], Flame epidemic [3] and 2011 Sony PlayStation Network data theft [4].

Hardening computer systems to resist attacks such as these is a complex process, with a multitude of advice and suggestions available in academic literature, open source documents and popular writing. Throughout all of these sources, it is rare to find cyber security advice that does not include a recommendation for some form of intrusion detection system. Indeed, government agencies charged with providing advice on computer security often make specific mention of IDSs, with the Australian Defence Signals Directorate listing a host-based IDS (HIDS) as fifth in its list of the top 35 recommended strategies to mitigate cyber intrusions [5].

This paper introduces a new semantic based algorithm, which performs significantly better than the existing methods. Two key research focusses fuelled this development, namely the search for increased core performance and the need to produce a robust and portable system capable of withstanding changes to the system baseline whilst degrading gracefully as the systemic noise levels increase. Three different data sets were used to evaluate this new algorithm; core performance was tested using the KDD98 data set and the new ADFA Linux data set (ADFA-LD), with portability and robustness testing conducted using the UNM data set.

Whilst increasingly criticised by articles such as [6], [7], [8], [9], [10], [11], the KDD98 is still used as a common benchmark for performance, though with a warranted level of scepticism about its applicability to the modern environment. Recent publications such as [12], [13], [14] are examples of the continued use of this publicly available data set, largely due to the absence of a readily accessible alternative. In an attempt to address the issues raised with the KDD98 data set, the new ADFA-LD was developed, using a modern operating system and contemporary attacks, for further evaluation of the semantic algorithm. This new data set is available for public use without restriction, and can be obtained by emailing j.hu@adfa.edu.au.

Numerous different decision engines (DEs) were trialled in various configurations throughout the experiments detailed in this paper, with a clear superiority shown by those DEs which used the new semantic feature as the basis for their classification. Whilst the main innovation of this paper is the semantic feature itself, a new type of artificial neural network, the extreme learning machine (ELM) [15], demonstrated superior performance against all data sets, and is discussed further in the body of the paper.

The rest of this paper is organised as follows: Section 2 presents a brief background and literature review of the field. Section 3 outlines central algorithms and theories which underpin the innovations presented in this paper. Section 4 discusses the key semantic concepts used to produce this paper's algorithm, and presents the core theory as a mathematical equation. Section 5 details the experimental methodology used in this research, while Section 6 contains

- *G. Creech is with the Cyber Security Research Group, University of New South Wales, Canberra Campus, Northcott Dr, Canberra, ACT 2600, Australia. E-mail: g.creech@adfa.edu.au.*
- *J. Hu is with Cyber Security Research Group, University of New South Wales, Canberra Campus, Northcott Dr, Canberra, ACT 2600, Australia. E-mail: j.hu@adfa.edu.au.*

Manuscript received 5 July 2012; revised 4 Dec. 2012; accepted 7 Jan. 2013; date of publication 24 Jan. 2013; date of current version 5 Mar. 2014.
Recommended for acceptance by M. Guo.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2013.13

the results from these experiments. Sections 7 and 8 provide a discussion of the experimental results and concluding remarks respectively.

2 Background and Literature Review

There are two broad classifiers which can be applied to an IDS. First, an IDS can be considered as either a host-based or network-based system. Host-based systems are designed to protect a single computer, and to prevent the execution of malicious code on that single system. Network-based IDS (NIDS) provide a different form of protection by examining network traffic in an attempt to detect intrusions.

Intrusion detection systems can be further classified as anomaly centric or signature-based. An anomaly IDS operates by first establishing a robust baseline of normal behaviour for the protected system. Theoretically, if this baseline is sufficiently accurate and extensive then any divergence from the baseline would be considered as an intrusion alarm. Signature-based systems, on the other hand, compare observed behaviour against a database of templates representing identifiable features from previously seen attacks. If behaviour matches a template then an alert is raised, but otherwise all activity is deemed legitimate.

Anomaly centric IDS suffer from high false alarm rates due to the difficulty of creating a robust and pervasive baseline. Computer activity is very dynamic within modern systems, and the high level of variation that results from this dynamism greatly increases the difficulty of distilling an effective and accurate baseline from which to measure anomalous divergences. The great advantage of the anomaly-based approach, however, is that no prior knowledge of the attack is necessary to raise an alarm; in other words, anomaly centric systems are capable of detecting completely new 'zero-day' attacks. Signature-based systems generally have lower false alarm rates and better detection rates for attacks which match a template within the database, but have no capability whatsoever to detect a previously unseen attack. This represents a critical weakness in the signature-based approach, and consequently increases the attractiveness of a robust anomaly centric system, provided that the traditionally high false alarm rates can be reduced to manageable levels.

Host-based IDS are usually implemented by selecting a metric present in the host and using this metric as the input to a decision engine. One family of techniques relies on information from the various log files present in a computer to provide this feature, with research such as [16], [17] presenting examples of solutions using this approach. The use of log files, however, suffers from several weaknesses. First, and most fundamentally, log files represent interpreted data. Log files are generated by daemon programs monitoring system activity, and inherently and irrevocably provide a diluted data source. Second, as discussed in [18], [19], the production of log files is not a seamless process, with a large amount of potentially irrelevant data given equivalent priority to critical data, accompanied by the mechanical problems of managing and creating log files.

The alternative to a log file-based approach is to use system calls as the raw data from which to formulate a decision feature. This approach was first suggested by Forrest [20], whose seminal work resulted in numerous subsequent research. The main advantage of a system call approach is that it allows the IDS access to raw data, providing insight into the interactions between programs and the kernel itself. This, in turn, allows for the best data granularity achieved in the IDS field to date, with a corresponding improvement in decision engine results. Additionally, system calls can be sampled in real-time without the issues which log file management can introduce into an IDS algorithm.

The performance of IDSs is usually measured by considering the attack detection rate plotted against the false alarm rate. This method of evaluation takes into account the base rate fallacy, and provides a true indication of the positive and negative impacts of deploying a given IDS. A low detection rate usually means a lower false alarm rate, which reduces the burden on a system administrator but consequently increases the chance of success for attackers. Conversely, a high detection rate will provide better protection but at the expense of numerous false alarms, accompanied by an increase in the required involvement from administration staff and a decrease in the actual effectiveness of the system. Numerous techniques have been suggested as ways to overcome this base rate fallacy, such as decision engine fusion [21], [22], the use of multiple decision engines [23] and the use of hybrid methodologies [24]. At heart, however, all these methods still rely on the same raw data, and use similar ways of interpreting this data. The research outlined in this paper introduces a completely novel way of interpreting the raw system call traces, utilising a true semantic interpretation to drastically improve results.

Having discussed the data options available to IDS designers, the next key ingredient is the decision engine. A decision engine, as the name suggests, is the algorithmic process by which a preprocessed feature extracted from the raw data discussed above is converted into an output which can be used to classify samples. Put simply, a DE assesses kernel activity and labels it as either normal or anomalous. Many different types of DE have been applied to the IDS problem. Some are quite simple, such as the easily implemented yet still powerful STIDE process [20], [25], [26], which uses a calculated ratio to classify system call patterns, but some DEs can become extremely complex, introducing large training burdens and consuming non-trivial amounts of system resources.

One of the major areas in which performance gains can be made in an IDS is the decision engine, and as such, significant effort has been expended in refining the performance of the DE. One technique which has been soundly investigated in search of performance gains is the Hidden Markov model [26], [27], [28], [29]. Hidden Markov models provide an extremely strong sequence recognition capability, and hence are well-suited to a system call analysis. Artificial neural networks, including self organising maps [30], radial basis function (RBF) networks [31], and multilayer perceptrons (MLPs) [32], have also been used as the DE for HIDS, capitalising on their strong pattern

recognition capability. Statistical methods, such as described in [33], provide an alternative approach, and are well suited to offline processing and handling large amounts of training data.

All the decision-making methodologies discussed above, as well as the many other DEs which have been used in an IDS setting, rely on appropriate training to reach their full effectiveness. Training data selected for this purpose must be representative of the system to be protected. This introduces a new consideration into the training process; each host is significantly different in function and behaviour, and hence almost certainly will require bespoke training data. The problem is further complicated by the dynamic nature of host behaviour, with baselines changing over time, necessitating periodic retraining. Ideally, the majority of training should be able to be conducted offline in a centralised manner, and then distributed to the individual hosts as required. Whilst a level of tuning will always be necessary, a robust feature and good initial training data will reduce the processing load on the eventual host. As the robustness of the feature used by the DE increases, so too does the portability of the trained IDS. In this paper, a semantic theory of intrusion detection, based on established natural language methods [34], [35] has been introduced to create novel semantic features for decision engines. As demonstrated by this paper's research, a semantic feature allows for a previously unseen level of portability and hence offers significant efficiency gains when large-scale deployments are considered.

The new method presented in this paper centres on using a full semantic feature as the input to a neural network. The semantic feature is derived by analysing discontiguous system call patterns, and has been demonstrated to be extremely powerful by this research. It is well suited to the extreme learning machine artificial neural network, with the synergies between this DE and the new feature leading to much better performance in sustained deployments. The semantic feature represents the core of this research and can be applied to any decision engine, albeit with some modification. Artificial neural networks, such as the ELM, are able to apply the new semantic feature simply and rapidly by counting occurrences of semantic phrases, as discussed in Section 5. Similarly, Hidden Markov models are also easily able to apply a semantic feature by searching for token sequences consisting of allowable semantic units. Given the strong demonstrated results outlined in Section 6, the use of more complicated DEs, accompanied by a greater processing overhead, is not warranted for single host deployments. DEs which are unable to apply the feature simply are better suited to a decision fusion application, such as in a distributed IDS arrangement, where the more complex threat is better defeated with broader detection methodologies.

This research has produced an algorithm suitable for the monitoring of a full system, evaluating all processes concurrently. By contrast, other methods use a per-process evaluation approach whereby each individual process is examined for deviances from the normal state defined for that particular process. This latter method has the potential to achieve a good level of accuracy, more so than for a whole-of-system approach. The source of this accuracy lies in reducing the decision horizon, limiting the scope of analysis to a single process. The undisputed benefit of this method is an increase in accuracy, and this is a powerful feature of the per-process method.

The major criticism of the per-process approach is that this method does not scale well. Separate evaluation for each process must be conducted, which in a busy system imposes a significant processing burden. Additionally, specific training data for each monitored process must be provided, and the IDS trained individually for each process. The semantic approach can also be incorporated easily into a per-process IDS. Extra security could then be provided by including a per-process IDS for specific high-risk attack vectors, such as the notorious SMB/Samba service.

A recent development in IDS theory is the mimicry attack, whereby an attacker obfuscates their malicious payload at the system call level in an attempt to bypass the system's HIDS. This technique, introduced by [36], is particularly important for sequence based IDS, such as the research proposed in this paper. Various methods have been proposed in an attempt to address this type of attack, ranging from the obfuscation of the system baseline in order to complicate the mimicry process [37], through increased DE performance to produce a tighter decision envelope [38], to the consideration of system call arguments in an attempt to increase the amount of data available for DE training [39], [40].

No public data is available for the evaluation of mimicry attacks, but some papers propose a method of generating such data using genetic algorithms [41], [42]. A semantic algorithm, such as that proposed by this research, should have a natural resilience to mimicry attacks, but further testing is infeasible at this time due to the absence of publicly available data. The inferred resilience is discussed more fully in Section 5.3.

3 Preliminaries

3.1 The System Call Approach to IDS

The system call approach to intrusion detection was first suggested by Forrest [20], based on the hypothesis that only running code will affect a system. As such, all anomalies should leave relics in the system calls executed by the kernel. Root processes were prioritised by this seminal research, with a risk assessment between the damage caused by super user actions and normal user actions forming the base of this decision.

Forrest used this core hypothesis as the basis for an artificial immune system approach to intrusion detection, building up a collection of normal system call traces for each program and using this as a self/non-self classifier. Of significance, only the system calls themselves are examined under this methodology, with the arguments to the individual instructions having no weight.

The richness of data contained in system call patterns is unrivalled. System calls represent the rawest interaction between a program and the host system, and have virtually no abstraction of data. Other methods such as log file analysis introduce an unavoidable level of obfuscation as they rely on data which has already been interpreted and formatted to produce the logs; decisions, classifications and clustering made on the basis of this data automatically inherits all assumptions made in parsing the raw source data to

form the log files and similar interpreted data collections. Therefore, the best granularity in intrusion detection systems should be seen in those algorithms which use raw system call data.

The analysis of system call patterns is usually performed without considering the arguments passed to each system call. This process loses some inherent information, but still allows for accurate classifications based on the positional relationships between system call sequences. Research such as [43], by contrast, actively considers the arguments passed to each system call, producing a different kind of semantic analysis. This form of semantic analysis is distinct to the structure proposed by this paper, as our work considers the semantic patterns in system call traces, rather than the semantic meaning of the arguments in each system call.

3.2 An Overview of the Extreme Learning Machine

The extreme learning machine methodology, [15], is an extremely powerful decision engine in the artificial neural network family. Neural networks have been thoroughly researched, with a general consensus as to their strong pattern recognition capabilities. Like many types of decision engine, however, neural networks suffer from a high training overhead, traditionally requiring extensive resources and many iterations to reach an effective level of training.

The key characteristic of an ELM is that it conducts its training in one pass, using the Moore-Penrose pseudo-inverse to solve a least-squares equation, hence avoiding much of the traditional training problem associated with neural networks. The expedited training speed comes with a high processing overhead, but the impact of this requirement is small, as HIDS have access to the CPU of the host itself. Alternatively, offline training is possible as well, allowing for a large proportion of the processing load to be completed prior to end-user deployment.

When training an ELM, only the weights between the hidden layer and output nodes are adjusted; indeed, the weights between the input nodes and the hidden layer, along with any bias values, can be randomly and statically assigned, provided that all resultant weights and biases belong to the same continuous probability density function [15]. This central feature has two key effects, namely:

1. An ELM completes its training much faster than a traditional MLP, and avoids local minima concerns.
2. The processing requirement for this faster training is significantly greater than for a comparably sized traditional MLP.

The first point makes the ELM approach extremely attractive for use as part of a HIDS deployment. The more rapid training and smaller on-going footprint of an ELM reduces the long term burden imposed by the IDS, without unduly affecting decision granularity. The higher processing load when conducting initial training is concentrated temporally at the start of deployment, and can be performed offline to minimise system impact.

3.2.1 *The Use of the ELM as an IDS DE*

The ELM is a relatively new innovation in the machine learning field. It has many attractive characteristics, as it is an extremely lightweight but accurate MLP implementation. In our earlier work [44], we demonstrate that the ELM is a suitable DE for NIDS applications. Cheng et al. [45] provide a detailed analysis of the benefits of the ELM in a NIDS, concluding that it is a rapid training and high performing DE option. Interestingly, the authors of [45] were forced to use the ageing KDD data set for similar reasons to this paper's research.

Both these works, [44], [45] focus on the NIDS area, rather than the HIDS area; whilst closely related, the two families of IDS are quite distinct, and hence results cannot be directly compared. Furthermore, the research in [45] uses a signature based methodology, classifying into multiple classes. This approach is not directly comparable with the anomaly based system proposed by this paper. The ELM is a novel DE, but provides an identical functionality to a traditional MLP. The rapid training is highly attractive for the dynamic world of computer systems, but the innovation of this paper lies in the new semantic analysis rather than the evaluating DE. Indeed, as shown in Section 6, Fig. 1, the ELM performs better with the new semantic feature than the existing syntactic features, underscoring the strength of this new semantic approach.

4 Contiguous and Discontiguous Semantic Analysis

Semantic theory has been used in computer science for some time. This branch of study provides various methods for parsing programming languages, and allows for a robust approach to computer system design. Importantly, semantic theory does not have to be forced to fit a computer science application; rather, programming languages and system architectures naturally share a similar logic and structure to natural language, and hence can be analysed and modelled by semantic tools.

The application of semantic principles to a computer environment centres on defining a scalable set of rules governing the combination of terminating units. These rules consequently allow complicated concepts to be expressed in terms of simple components. The translation of highly abstract user actions to low-level kernel events is an involved process, and the formal structure provided by semantically inspired rules greatly assists in producing an effective interface. This relationship between user activity and kernel events has inherently underpinned all programming for some time, and certainly throughout the modern era.

The syntax required by programming languages is a reflection of the underlying semantic principles, and allow for efficient compiling and interpretation of user-produced code. It also suggests that the system call traces which result from the execution of programs written with these languages can be subjected to a semantic analysis.

Based on the core principles above, this paper proposed that a context-free grammar (CFG) could be applied to the system call traces as well as to the language structure used in creating the multitude of high-level programs present in any operating system. The mechanics and rules of a CFG are detailed in Sections 4.1 and 4.2, with the original semantic feature proposed by this paper defined mathematically in Section 4.3.

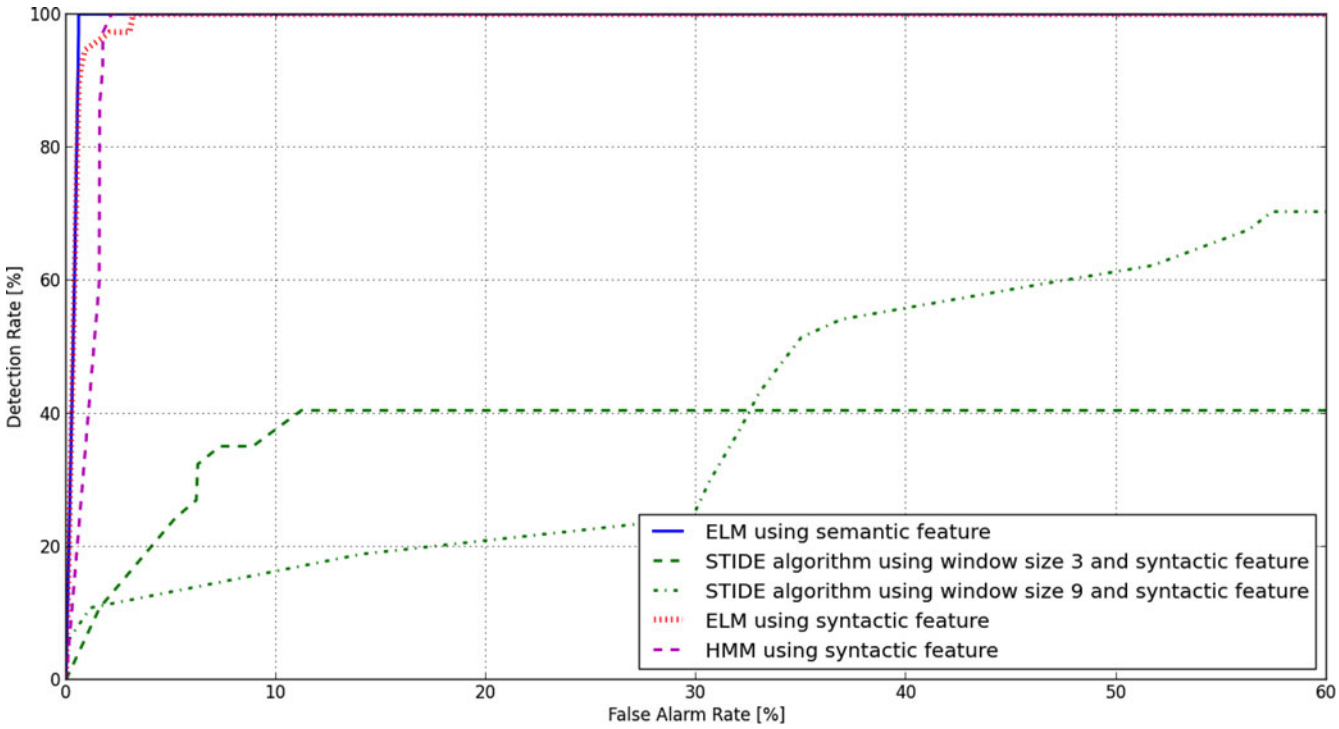

Fig. 1. ROC curves for assorted methodologies when assessing the KDD 98 data set.

The application of a true semantic analysis to system call patterns differs from current thinking significantly. Current methods analyse contiguous patterns of system calls and use these patterns as the basis for the decision engine's classification. Whilst good results have been produced, this is merely a syntactic analysis and does not represent the full richness of the underlying semantic structure. This paper proposes and verifies that a grouping of discontiguous system calls provide a much more powerful and robust feature for intrusion detection.

Under this new approach, system calls can be viewed as 'letters', with a string of contiguous system calls thus forming a 'word'. By applying a CFG, the resulting word lists can now be combined to form 'phrases'. It should be noted that when compiling the multi-word lists, the eventual phrases may not in fact have occurred in the training data under consideration. This does not invalidate these phrases from a full system perspective, however, and the algorithm is inherently able to function with smaller volumes of training data by allowing these phrases to persist within the language structure. Additionally, by not requiring each allowable phrase to be specifically seen during training, marked efficiency and accuracy gains are possible.

As discussed in Section 1, the training process for host-based intrusion Detection systems is lengthy, intensive, and specific to each particular host. The sensitivity of detection and false alarm rates means that a complete retraining process is often required when an IDS is deployed to a new host, or when the host's function changes significantly. The semantic feature proposed in this paper mitigates this sensitivity markedly, with an unprecedented level of portability present in the new IDS. The results presented in Section 6 clearly show that the feature proposed in this paper is easily transferable between operating systems, even without allowing for retraining.

4.1 Definitions

1. Let $\mathcal{T} = \{\text{architecture specific system calls}\}$.
2. Let $\mathcal{N} = \{\exists x \in \mathcal{T} : y = x_i, x_j, x_k \ldots\}$, or in other words, $\mathcal{N} = \{\text{all possible sequences of system calls}\}$.
3. Let $\mathcal{G}$ represent a known normal trace, and $\mathcal{A}$ represent a known anomalous trace.

4.2 Syntactic Development

From the definitions above, $\mathcal{T}$ represents terminating syntactic units, and $\mathcal{N}$ represents non-terminating syntactic units. As discussed in [34] and assuming context free grammar, the syntax of a system call trace can be defined as follows:

$$\exists x \in \mathcal{T} : \mathcal{G} \rightarrow x\mathcal{G}'. \tag{1}$$

Using Production 1, a sequence of training data $[\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_N]$ yields a new set

$$\mathcal{B} = \{\exists x_i, x_j, x_k, \ldots : x \in \mathcal{T} | (x_i, x_j, \ldots) \in [\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_N]\}. \tag{2}$$

This new set $\mathcal{B}$ has elements which represent the phrases made up by sequences of system calls. Whilst $\mathcal{B} \subset \mathcal{N}$, there is no guarantee that any specific element of $\mathcal{B}$, denoted b_q, has been observed in the training data $[\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_N]$. Indeed, b_q, being made up of disparate system calls, may not occur in the corpus at all. Rather, it represents the

TABLE 1
Full Feature Format

	Phrase Length				
Sample Number	1	2	3	4	5
Raw value: $i_R \in \mathcal{I}_\mathcal{R}$	Dict. 1 Count	Dict. 2 Count	Dict. 3 Count	Dict. 4 Count	Dict. 5 Count
Sample 1 (Raw)	1014	1118	713	397	130
Sample 2 (Raw)	345	145	4	0	0
Normalised value $i_N \in \mathcal{I}_\mathcal{N}$	$i_R[1]/\max(\mathcal{I}_\mathcal{R})$	$i_R[2]/\max(\mathcal{I}_\mathcal{R})$	$i_R[3]/\max(\mathcal{I}_\mathcal{R})$	$i_R[4]\max(\mathcal{I}_\mathcal{R})$	$i_R[5]/\max(\mathcal{I}_\mathcal{R})$
Sample 1 (Normalised)	0.0412	0.0454	0.0289	0.016	0.005
Sample 2 (Normalised)	0.0140	0.0059	0.0002	0.000	0.000

possibility of that call sequence occurring, noting the observed calls from $\mathcal{T}$ in $[\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_N]$.

4.3 Semantic Hypothesis

Of note, an anomalous trace $\mathcal{A}$ will exhibit a very similar *syntactic* pattern to a normal trace $\mathcal{G}$. Semantically, however, a strong difference is evident.

The following inequality was hypothesised as the core element of this paper's innovation, where $\mathcal{G}_n$ represents a previously unseen normal trace:

$$|\{\exists x \in \mathcal{B} : \mathcal{G}_n \rightarrow x\mathcal{G}'_n\}| \gg |\{\exists x \in \mathcal{B} : \mathcal{A} \rightarrow x\mathcal{A}'\}|. \quad (3)$$

Equation (3) indicates that the occurrence of valid semantic units extracted from the training data in a new normal trace should be significantly greater that the occurrence of these semantic units in a new anomalous trace. The set $\mathcal{B}$ is generated from the training data, and consists of the observed semantic units in this data. Of note, $\mathcal{B} \subset \mathcal{N}$. By reducing the allowed set of semantic units to those observed in the training data, it is then possible to use this reduced set as the basis for a decision feature by calculating the occurrence counts proposed in Equation (3). The implementation of this theory is detailed in Section 5.

In summary, *semantic* analysis, as opposed to *syntactic* analysis, suggests that a corpus of phrases can be generalised from known normal system traces, and will include all normal semantic units in its membership, irrespective of whether a specific phrase has been observed in the training data, presupposing the initial training set is expansive, but not necessarily exhaustive. Equation (3) represents a decision method suitable for use as the basis for a decision engine, with the decision engine feature generated by constructing a vector of the valid phrase counts for varying lengths. The hypothesis presented as Equation (3) was verified by the experiments and results detailed in Sections 5 and 6, performing extremely strongly.

5 Method

5.1 Algorithm

Given the semantic approach discussed above, especially Equation (3), several steps are required to apply this concept to intrusion detection systems. First, the training data must be processed to extract a dictionary containing every contiguous system call trace present in the training samples. This step is equivalent to using multiple window lengths under Forrest's methodology [20], [25], [26], [46] and [47], where the maximum window length allowed is in fact the length of each trace. Each dictionary entry extracted at this stage forms a conceptual 'word', or a 'phrase' of length 1.

Second, these words are then used to construct further dictionaries consisting of every possible combination of the words up to a specified phrase length. The experiments in this paper used phrases of lengths ranging from one word to five words. The upper limit of length 5 was applied as the number of phrases detected in each trace of the KDD98 training data at this length started to drop significantly, indicating that longer phrases would not be representative of the true nature of the corpus. In a real-time deployment, traces would consist of more system call elements, and hence longer phrases could also be used, thus further increasing the richness of the feature. It is important to note that there is no requirement for the derived phrases to have been observed in the training data. At this stage, it is sufficient that their presence is theoretically possible based on a combination of observed words. This is a key difference of this new method when compared to existing approaches.

Having these dictionaries themselves does not provide a usable feature per se; second pass processing is required to extract occurrence counts of these different length phrases. In this step, the training data is re-examined and the system call patterns compared against the theoretically possible phrase lists. This resulting numerical data represents the number of phrases consisting of discontiguous words occurring in each training data sample.

Again, this differs from a contiguous approach as more weight is inherently given to longer phrases of discontiguous words. After normalisation and standard data treatment routines, this information is used to train a decision engine. Table 1 shows the structure of the 5-element feature used in these experiments. When new data is received, it is compared against existing phrase dictionaries, with the discontiguous word phrase-count again used as the input to the decision engine. The now trained decision engine is then able to classify this new sample based on this feature alone. Pseudocode for the entire process is provided below:

5.2 Training Overhead

Building the phrase dictionaries for a semantic IDS is, at present, a lengthy task. Even given the reduction of $\mathcal{T}$ by the creation of $\mathcal{B}$ from the training data, the resulting phrase dictionaries in their native form are large. As the phrase length increases, so too does the size of the dictionary; as the length of phrase is directly proportional to the information content and relevance of the feature, this means that the most valuable phrase dictionaries take the longest to produce. In one sense, the design tradeoff

required by a semantic IDS is longer data processing for greatly increased accuracy.

```
1: function GETWORDS(traces)
2:     for all traces do
3:         counter ← 1
4:         for system calls in trace do
5:             word =systemcall+nextcountercalls
6:             if word in wordDictionary then
7:                 Increment count of word
8:             else
9:                 Add word to wordDictionary
10:            end if
11:            counter + = 1
12:        end for
13:    end for
14:    return wordDictionary
15: end function
16: function GENPHRASES(word dictionary, length)
17:    Create new phrase dictionary for phrases of \
18:        given length
19:    for all words in word dictionary do
20:        while current phrase length < length do
21:            currentPhrase ← currentWord
22:            for currentWord do
23:                currentPhrase + = next dictionary\
24:                    word
25:            end for
26:        end while
27:        Add phrase to phraseDictionary
28:        word list start position++
29:    end for
30:    return phraseDictionary
31: end function
32: function GETPHRASECOUNT(trace)
33:    featureVector = new array with \
34:        length=number of dictionaries
35:    for all Phrase Dictionaries do
36:        i ← phrase length for dictionary
37:        phraseCount ← 0
38:        for all Phrases in Dictionary do
39:            if phrase present in trace then
40:                phraseCount++
41:            end if
42:            featureVector[i] ←phrase count
43:        end for
44:    end for
45:    return featureVector
46: end function
47: function EVALUATESYSTEMCALLTRACE(newTrace)
48:    newFeature ← getPhraseCount(newTrace)
49:    normalise newFeature
50:    feature → trained decision engine
51:    deResult ← decision engine output
52:    if deResult > global threshold then
53:        classification ← anomalous
54:    else
55:        classification ← normal
56:    end if
57:    return classification
58: end function
```

Fortunately, once the phrase dictionaries have been created, feature extraction and DE training takes no longer than for syntactic methods. As such, the current semantic harvesting process should be viewed as a mostly offline task, which consequently has little impact. Compilation of the phrase dictionaries is only required once, however, and can be accomplished in days. As such, the one-off processing burden is easily affordable within the lifetime of the IDS.

5.3 Theoretical Resilience to Mimicry Attacks

The semantic theory proposed by this research has an inherent theoretical resilience to mimicry attacks. This type of attack, as defined by [36], describes the process followed by skilled attackers of modifying the system call footprint of their payload to prevent detection by an IDS. This is done by the insertion of virtual NOPs, or *no-operation* functions, or through the creation of convoluted system call patterns which obfuscate the intent of the payload sequence.

The semantic algorithm proposed here has a natural resilience to this class of attacks. The semantic evaluation process assigns significant weight to long semantic phrases; if a semantic phrase is broken by the inclusion of an attacker's system call then the long semantic phrase no longer counts towards the assessment of that trace as normal. Inclusion of multiple attacking system calls in the trace will have the effect of fracturing all long semantic chains, which in turn means the trace is likely to be classified as anomalous once more. As such, even if an attacker includes similar semantic phrases in their obfuscation attempts, the longer natural phrases will still have been broken, resulting in the classification of the activity as anomalous.

5.4 Validating Experiments

5.4.1 KDD98 Data Set

To validate the core theory and algorithm presented above, several experiments were conducted using the KDD98 data set. Four different combinations of IDS feature were tested on this data set, thus allowing direct comparison between the benefits of each approach. These trials were:

1. The standard STIDE approach proposed by Forrest [20].
2. The use of contiguous syntactic units of varying window sizes as the input to an ELM.
3. The use of different length contiguous syntactic units as the input to an HMM.
4. The use of the new feature proposed in this paper as the input to an ELM, representing a true semantic analysis of the system call corpus.

Each trial used the same data from the KDD98 data set [48]. The training data for each combination consisted of 500 known normal traces, and the validation data consisted of 4,500 previously unseen known normal traces. Each trace was collected from the Solaris BSM data extracted from the host *Pascal*, running Solaris 2.5.1 [48]. The resulting data files after formatting consisted of the system call traces for each process running on the host during the sampling period. The attack traces comprised a total of 37 attacks belonging to 10 attack families. This represents all attacks present in the data set with detectable relics in the BSM files.

TABLE 2
ELM Feature Format for Trial 2

	Word Length				
Sample Number	3	4	5	6	7
Raw value: $i_R \in \mathcal{I}_\mathcal{R}$	Dict. 1 Count	Dict. 1 Count	Dict. 1 Count	Dict. 1 Count	Dict. 1 Count
Sample 1 (Raw)	10	10	10	9	7
Sample 2 (Raw)	9	6	6	5	4
Normalised value $i_N \in \mathcal{I}_\mathcal{N}$	$i_R[1]/\max(\mathcal{I}_\mathcal{R})$	$i_R[2]/\max(\mathcal{I}_\mathcal{R})$	$i_R[3]/\max(\mathcal{I}_\mathcal{R})$	$i_R[4]\max(\mathcal{I}_\mathcal{R})$	$i_R[5]/\max(\mathcal{I}_\mathcal{R})$
Sample 1 (Normalised)	0.1538	0.1538	0.1538	0.1384	0.1077
Sample 2 (Normalised)	0.1384	0.0923	0.0923	0.0769	0.0615

For evaluation purposes, detection rate, D, and false alarm rate, F were defined as:

$$D = \frac{\text{number of detected attacks}}{\text{number of attacks present}} \times 100\%,$$

$$F = \frac{\text{number of false alerts}}{\text{number of traces in validation data}} \times 100\%.$$

Trial 1 was implemented by applying the STIDE method detailed in [20], [25], [26], [46]. Different length window sizes were applied under the same experimental conditions as the other trials, with the minor difference that a complete system call list was provided to the STIDE algorithm as it was compiling its initial sequence databases. This is an important consideration, as there is no guarantee that all system calls are present in the training data. The STIDE algorithm is particularly sensitive to any omitted calls, and pays an unrealistically stark false alarm rate penalty if a complete system call list is not provided. This allowance was also made for the Hidden Markov model trial for similar reasons, but was not required for the ELM trials due to the increased robustness of the feature proposed in this paper. ROC curves were generated by varying the decision threshold and plotting the resultant detection rate and false alarm rate. This, in turn, allowed an evaluation of the merit of each algorithm by considering the area under the curve [49]. As discussed in [49], the performance of an algorithm is effectively profiled by this method, with greater area under the ROC curve equating to superior performance.

Trial 2 used the basic word, or phrase-length 1, dictionary to provide an input source for an ELM. This represents the most simplistic application of the novel feature introduced in this paper. After training, each incoming trace was assessed against the basic word dictionary, with the counts of each window length used as the neural network input, as shown in Table 2. This is similar to the full feature used in Trial 4, and starts to take advantage of the robustness inherently present in a full semantic approach. By applying this method, even though only a syntactic assessment, the decision engine is able to simultaneously consider the underlying syntax of different length words. For this experiment, words of length 3 to 7 were allowed, matching the five inputs used for the full feature in Trial 4. The lower limit of length 3 was selected to avoid unduly restricting this trial by requiring processing of the trivial lengths 1 and 2, with the upper limit selected to limit the outlier-effect of longer words, which are low-count but high-impact. ROC curves in this case were also generated by varying the decision threshold.

Trial 3 uses the strong sequence recognition capability of Hidden Markov models to capitalise on the seminal system call approach proposed by Forrest [20]. Under this method, a level of abstraction is introduced and the state transition which produces system call patterns becomes viewable as a stochastic process. As such, HMMs are able to predict this state transition with a high level of accuracy. The system parameters used in this experiment are shown below:

- $N = 2$, with the state either being 'Normal' or 'Anomalous'.
- $M = \{\text{all allowable system calls}\}$.
- T as specified by the individual experiment iterations; equivalent to STIDE window length.

In applying HMMs for this experiment, the expedited training schemes and general methodology outlined in [23], [27], [29], [50] were used to craft the final implementation. ROC curves for this experiment were plotted by using the Viterbi algorithm [51] to predict the most likely state path through a sequence of observations, and plotting the resulting D and F for different sequence lengths.

Trial 4 represents the full implementation of the proposed semantic feature and decision methodology. In this experiment, a wholly semantic approach was adopted by using phrases of lengths 1 to 5. These phrases were extracted from the training data and have the structure shown in Table 1. Each element of the training data was condensed into five normalised numeric values, representing the phrase occurrence counts at each of the specified depths. The resulting feature vectors were then used to train an ELM prior to testing on the validation and attack data. To determine whether a new trace represents an attack or not, a threshold was applied to the neural network output with a classification being made in response to this assessment. ROC curves were generated by varying the threshold and plotting the resultant detection and false alarm rates.

5.4.2 Portability Testing Using the UNM Data Set

As discussed in Section 1, portability of an IDS is a challenging task. In order to generate high-quality results, significant effort, time and resources must be invested in the training process for each intrusion Detection system. Furthermore, this training is very specific, particularly for host-based systems, meaning that the training cost must be borne again for each individual deployment.

To some extent, this problem cannot be mitigated completely, as each host will have peculiarities and idiosyncrasies which must be accounted for in order to avoid unacceptably high false alarm rates. Indeed, the same host used in a different role, such as a web server

being recast as a file server, will have a vastly different baseline. Notwithstanding the persistent nature of some aspects of this problem, any reduction in training time when redeploying IDS is a significant benefit, and highly desirable for real-world applications.

As part of this paper's research, the portability problem was considered and investigated. Whilst a large range of metrics can be used to quantify performance in pursuit of this aim, this research simply used detection and false alarm rate performance to quantify the transference factor of each algorithm. Future research effort could be valuably expended in optimising the transference process.

A new data set was selected to evaluate the portability of the various IDS methods explored in the first phase of experimentation, namely the UNM corpus [52]. This data set has been widely used in IDS research, and represents a collection of synthetic and live traces for various operating systems. Importantly, one of the operating systems is the SunOS 4.1.1, or Solaris 1, system. This system is the precursor to the OS used in the KDD98 experiment, and as such this data set provides a viable candidate for portability testing, with the similarity of the operating system allowing certain assumptions to be made about the commonality of system calls and underlying structure.

The Solaris traces provided as part of the UNM data set consist of 618 known normal traces, and eight different attack traces. To evaluate the portability of the IDS algorithms specified in Trials 1-4, the trained decision engines were applied to this new data. False alarm rate was calculated using the known normal traces, and detection rate calculated by using the attack traces. Whilst not a particularly large data sample, the subset of the UNM data set used is sufficient to compare performance between the methodologies, with the semantic feature clearly outperforming the other methods. No allowance was made for different system calls in the second data set, which accounts for much of the reduced performance seen across all methods. Notwithstanding this, the semantic feature clearly provides a credible capability, and future research will focus on smoothing the transference process further.

5.5 Evaluation against Modern Attacks Using the ADFA Linux Data Set

To provide a modern perspective for performance evaluation, the ADFA-LD was created. This data set uses a fully patched Ubuntu Linux 11.04 [53] installation as the host OS. Apache Version 2.2.17 [54] running PHP Version 5.3.5 [55] were loaded to provide for web based attacks, with TikiWiki Version 8.1 [56] installed to provide a web-application attack vector through the known vulnerability described in [57]. This configuration represents a realistic modern target with small security flaws which can be exploited incrementally to provide a full system compromise.

The system was attacked by a certified penetration tester [58] using current best-practice methodology. Attacks used included web-based exploitation, simulated social engineering, poisoned executables, remotely triggered vulnerabilities, remote password brute force attacks and system manipulation using the C100 webshell. Payloads included variably encoded command shell and Meterpreter [59] shellcode delivered using a wide range of vectors. Full details of the new data set have been submitted for publication and are under review at the time of writing, but are available in the meantime by contacting the authors. The ADFA-LD consists of 833 normal traces for training the IDS, 4,373 normal traces for evaluating FAR and 60 different attack sets, each consisting of multiple traces.

6 Results

6.1 Core Performance Using KDD98 and ADFA-LD

The first research question investigated centred on the innate performance of the new semantic algorithm. Two data sets were used for this evaluation, namely the KDD98 data set and the new ADFA-LD. Whilst undeniably dated, the KDD98 data set represents a benchmark within the field and is still actively used by researchers, for example [12], [13], [14]. Results were generated using this data set to allow direct comparison with other research, and whilst sub-optimal, use of a common data set allows for independent verification of results. The ADFA-LD, as discussed in Section 5.5, provides a modern challenge for the algorithm and gives a better indication of performance against contemporary attacks. Each DE method was trained using the same set of normal traces, with false alarm rates calculated by then processing a separate set of normal traces and calculating the number of alerts. The attack traces were then classified, with detection rate calculated from the number of alerts arising from this assessment.

No special consideration was given to selecting either training data or validation data, with samples randomly chosen. All traces represent the system calls from a single process running on the host over the selected time interval. Fig. 1 shows the ROC curves for each trial, with saturated performance for the new semantic algorithm achieved with 100 percent DR for 0.6 percent FAR, which is significantly better than the results produced by competing algorithms.

The ADFA-LD is a much harder data set, and absolute performance was significantly worse for all algorithms when compared to the accuracy produced classifying legacy KDD98 data. ROC curves are shown in Fig. 2. Despite the worse overall performance, the performance of the semantic based ELM is again clearly superior to all other algorithms as it maximises the area under the curve [49]. An additional DE, the single-state SVM, was included in this trial to demonstrate that the observed superior performance originates from the semantic feature itself, rather than the DE. This point is further reinforced by considering the performance of the ELM with a semantic feature and without the semantic feature, shown in Fig. 1, with a clear performance benefit demonstrated when the new semantic feature is used.

6.2 Portability and Robustness of the Algorithm

The second key research question addressed by this paper is the comparative resilience of the new algorithm to changes in the baseline, coupled with the portability of the semantic dictionaries between similar operating systems. This

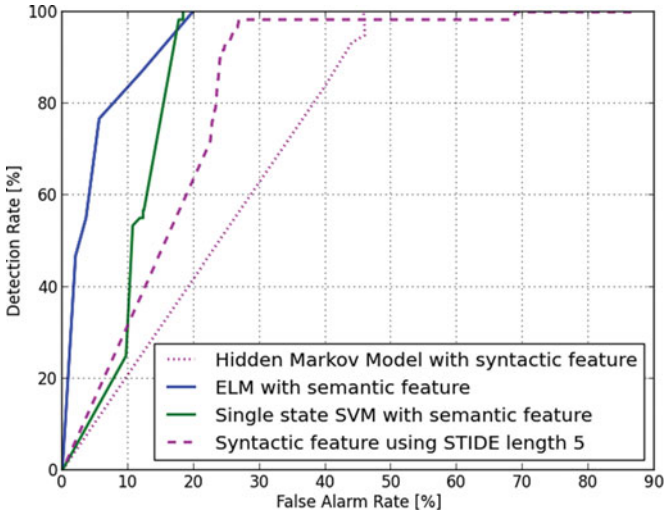

Fig. 2. ROC curves for assorted methodologies when assessing the ADFA-LD.

research question was investigated by taking systems trained using KDD98 data and using them to classify the UNM data set. Only a section of the whole data set is appropriate for this phase, namely those traces which belong to the same operating system type, but of a different version.

Six hundred and eighteen normal traces from this data set were used for validation and false alarm rate calculation, with the eight attack traces used to calculate detection rate. The key difference between the UNM data set and the KDD98 data set is that the UNM traces are grouped by specific program, whilst the KDD98 traces reflect all activity on a host at a given time. As such, the UNM data set can be viewed as more specific, whilst the KDD98 data set contributes much greater breadth to the assessment problem.

The ROC curves for this phase of experimentation are shown in Fig. 3. Trial 1, the standard STIDE approach, was not applied to the UNM data as the different system call sets between the two versions of the operating system unduly disadvantages this method, producing low detection rate and high false alarm rates.

7 Discussion

These results clearly show the superiority of a semantic approach, with comparative results from other algorithms show in Table 3. The ROC curve for the full semantic feature

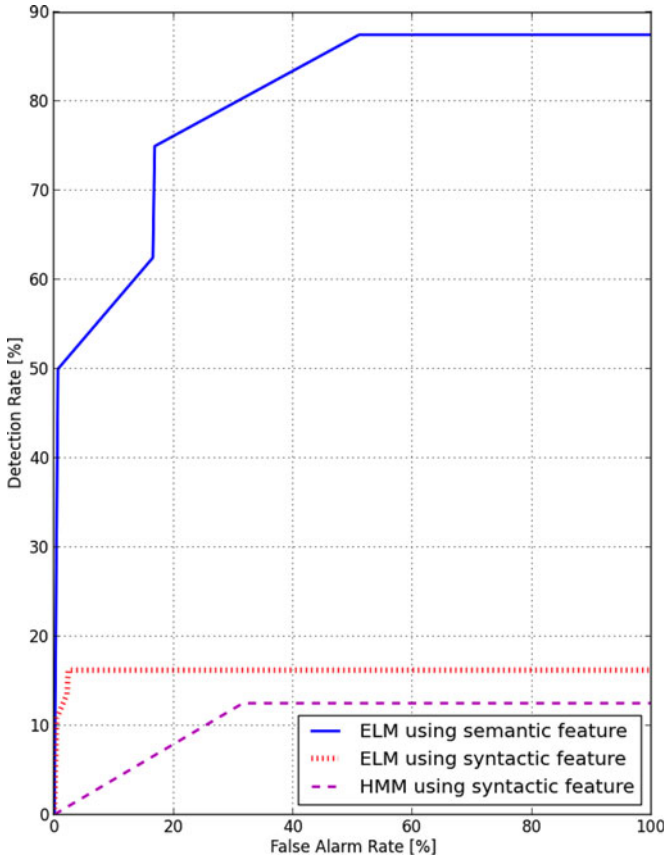

Fig. 3. ROC curves when assessing the UNM data set after training on the KDD98 data set.

on KDD98 is excellent, and is functionally very close to a theoretically perfect system. The benefit of using multiple length syntactic features has also being clearly demonstrated in both Trials 2 and 3. By simultaneously considering five different window lengths, the accuracy of the ELM is close to that of the full semantic feature. Hidden Markov model performance using a syntactic feature is also good, performing better than many of the implementations detailed in Table 3.

The STIDE method used in Trial 1 did not perform as well as the other algorithms, which is initially surprising. This feature represents the seminal work in system call based IDS [20], and performs well in many applications; the key difference in this set of experiments is that the STIDE algorithm is required to process at a full kernel

TABLE 3
Comparison between Contemporary IDS Algorithms

Algorithm	Detection Rate [%]	False Alarm Rate [%]
Data mining of audit files [60]	80.2	Not cited
Multivariate statistical analysis of audit data [33]	90	40
HMM and entropy analysis of system calls [61]	91.7	10.0
System call n-gram sliding window (assorted decision engines) [46]	$95.3 < DR < 96.9$	~ 6.0
RBF ANN analysing system calls [31]	96 mean	5.4 mean
MLP ANN on subset of KDD98 [62]	99.2	4.94
SVM on subset of KDD98 [62]	99.6	4.17
kNN with Smooth Binary Weighted RBF [63]	96.3	6.2
Rough Set Clustering [64]	95.9	7.2
ELM using original semantic feature proposed in this paper	**100.0**	**0.6**

level, rather than being allowed to evaluate individual programs. Trial 1 represents a real-world application of the STIDE concept, where host-based intrusion detection systems are expected to monitor all programs concurrently. STIDE can be applied effectively on a program level, but as demonstrated in these experiments does not translate well to a full-system deployment.

This algorithm was not applied in the second phase of experimentation as it is extremely sensitive to the initial system call databases, which are extracted from the training data. As there is a non-trivial difference in the system call sets between Solaris 1 and Solaris 4, STIDE did not perform well and does not demonstrate portability. Notwithstanding this, STIDE is an effective IDS solution for single programs, and represents the earliest work in this field. It is a powerful algorithm, simply implemented, and produces good results within its design constraints. The purpose of including it in these trials was to demonstrate the significant difference between the problem of monitoring a single program, and monitoring the kernel as a single entity.

The portability of the full semantic feature was demonstrated during the second phase of trials. As can be seen from Fig. 3, this feature performed much better than either of the other two alternatives. By way of a broader comparison, the snapshot of contemporary algorithms included in Table 3 shows that all three trials used in this second phase perform much better than other approaches when in their primary training environment, in this case the KDD98 data set (see Fig. 1).

Despite this clearly superior performance of the new method, the usable data in the UNM data set was not substantial enough to draw definitive and permanent conclusions. With only 618 normal traces for validation and eight attack traces, these results should be viewed as an indication of potential only. Regardless, there is a clear performance difference between the various algorithms, with the semantic feature performing far better than its competitors. Further research is required to extend this initial finding, and to improve the transference process.

Portability between versions of the same OS, such as tested here, is a challenging task. Changes in system architecture from version to version result in a myriad of small differences, with a pronounced cumulative effect. In the applications considered in this paper, these differences manifest in a different set of systems calls, which is in effect, a different set of terminating semantic units, $\mathcal{T}$. In turn, this has the effect of forcing the transferred algorithm to function with only a subset of the potential system baseline. This factor is responsible for the significantly worse performance when assessing the UNM data set, and is not easily avoided.

Capitalising on the portability of the semantic feature is not an inherently simple task, but has many applications throughout the computer security sphere. On one hand, the generalisability of the semantic rules mean that much more offline training can be conducted, and in a more centralised manner when a good initial condition can be generated across different environments. This in turn means that the tuning process required at each host is less intensive, allowing for more expeditious deployment and more accurate results. Additionally, the use of a central semantic dictionary facilitates collaboration between distributed HIDS, as this approach provides a common and intuitive communication protocol.

Furthermore, a semantic approach greatly degrades the ability of an attacker to bypass security systems. Any such attempt has a clear semantic pattern, as shown by the results presented in this paper, and will thus be thwarted by the guardian IDS. This concept can be extended to include misuse detection, efficiency profiling, and network-centric analysis.

By reducing the false alarm rate to such a low level, this new feature elevates anomaly-based IDS to the accuracy levels normally only seen in signature-based IDS. Importantly, however, this new feature presents a true zero-day attack detection capability. As an anomaly-based HIDS, each attack which this system encounters is, effectively, a new attack. Hence, true protection is provided against previously unseen attacks, without the high false alarm rates traditionally associated with the anomaly-based approach.

The ELM neural network performed well throughout the experiments, as expected. This decision engine is, in effect, a standard MLP, which, along with other neural network types, has been extensively applied to IDS as seen in research such as [30], [31], [65], [66], [67]. Whist an innovation in its own field, the ELM is not a significantly different engine within the IDS context; notwithstanding this, the rapid training of this decision engine coupled with the robustness of the semantic feature opens up areas such as dynamically updating baselines as a viable possibility. The processing overhead of the ELM is offset to some extent by the robustness of the semantic feature, reducing re-training events to the bare minimum and consequently reducing the drain on the host system.

8 Conclusion

This paper has clearly demonstrated the superior results possible when using a full semantic analysis of system calls to derive a new feature. The ELM methodology has been verified as applicable to the IDS problem, with potential synergies uncovered due to the rapidity of decision engine training possible using this scheme. Portability between different versions of the same operating system has been investigated, and promising results suggest that the semantic approach introduced by this paper is extremely applicable to the task.

Public data sets were used for evaluation of the new algorithm proposed in this paper in order to allow comparison with existing approaches. Given the strong results obtained, further development will be conducted using contemporary data generated specifically for this purpose, in order to fully profile the strengths and weaknesses of the semantic approach and the transference process. The data generated during this experimentation will be made publicly available in due course, with the first portion already available as the ADFA-LD.

Future research will investigate the transference process further, along with attempts to reduce the training overhead and enhance the inherent resilience of the new semantic feature to mimicry attacks.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support from ARC (Australian Research Council) Projects LP110100602, LP100100404 & LP100200538.

REFERENCES

[1] J.P. Farwell and R. Rohozinski, "Stuxnet and the Future of Cyber War," *Survival*, vol. 53, no. 1, pp. 23-40, 2011.
[2] T. Chen, "Stuxnet, the Real Start of Cyber Warfare?" *IEEE Network*, vol. 24, no. 6, pp. 2-3, Nov./Dec. 2010.
[3] J. Naughton, "How the Flame Virus Has Changed Everything for Online Security Firms," http://www.guardian.co.uk/technology/2012/jun/17/flame-virus-online-security, 2012.
[4] S. Richmond and C. Williams, "Millions of Internet Users Hit by Massive Sony PlayStation Data Theft," http://www.telegraph.co.uk/technology/news/8475728/Millions-of-internet-users-hit-by-massive-Sony-PlayStation-data-theft.html, 2012.
[5] Defence Signals Directorate—Cyber Security Operations Centre, "Top 35 Mitigation Strategies for Targeted Cyber Intrusions," http://www.dsd.gov.au/publications/Top_35_Mitigations.pdf, Mar. 2012.
[6] C. Brown, A. Cowperthwaite, A. Hijazi, and A. SoMayaji, "Analysis of the 1999 DARPA/Lincoln Laboratory IDS Evaluation Data with NetADHICT," *Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications, (CISDA '09)*, pp. 1-7, July 2009.
[7] P. Owezarski, "A Database of Anomalous Traffic for Assessing Profile Based IDS," *Proc. Second Int'l Conf. Traffic Monitoring and Analysis (TMA '10)*, pp. 59-72, 2010.
[8] V. Engen, J. Vincent, and K. Phalp, "Exploring Discrepancies in Findings Obtained with the KDD Cup '99 Data Set," *Intelligent Data Analysis*, vol. 15, no. 2, pp. 251-276, 2011.
[9] S. Petrovic, G. Alvarez, A. Orfila, and J. Carbo, "Labelling Clusters in an Intrusion Detection System Using a Combination of Clustering Evaluation Techniques," *Proc. 39th Ann. Hawaii Int'l Conf. System Sciences (HICSS '06)*, vol. 6, p. 129b, Jan. 2006.
[10] M. Mahoney and P. Chan, "An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection," *Proc. Sixth Int'l Symp. Recent Advances in Intrusion Detection*, pp. 220-237, 2003.
[11] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory," *ACM Trans. Information and System Security*, vol. 3, no. 4, pp. 262-294, http://doi.acm.org/10.1145/382912.382923, Nov. 2000.
[12] W. Liu and J. OuYang, "Clustering Algorithm for High Dimensional Data Stream over Sliding Windows," *Proc. IEEE 10th Int'l Conf. Trust, Security and Privacy in Computing and Comm. (TrustCom)*, pp. 1537-1542, Nov. 2011.
[13] H. Bahrbegi, A. Navin, A. Ahrabi, M. Mirnia, and A. Mollanejad, "A New System to Evaluate GA-Based Clustering Algorithms in Intrusion Detection Alert Management System," *Proc. Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pp. 115-120, Dec. 2010.
[14] J.-H. Lee, J.-H. Lee, S.-G. Sohn, J.-H. Ryu, and T.-M. Chung, "Effective Value of Decision Tree with KDD 99 Intrusion Detection Datasets for Intrusion Detection System," *Proc. 10th Int'l Conf. Advanced Comm. Technology (ICACT)*, vol. 2, pp. 1170-1175, Feb. 2008.
[15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks," *Proc. IEEE Int'l Joint Conf. Neural Networks*, vol. 2, pp. 985-990, 2004.
[16] F. Bin Hamid Ali and Y.Y. Len, "Development of Host Based Intrusion Detection System for Log Files," *Proc. IEEE Symp. Business, Eng. and Industrial Applications (ISBEIA)*, pp. 281-285, Sept. 2011.
[17] L. Ying, Z. Yan, and O. Yang-jia, "The Design and Implementation of Host-Based Intrusion Detection System," *Proc. Third Int'l Symp. Intelligent Information Technology and Security Informatics (IITSI)*, pp. 595-598, Apr. 2010.
[18] J. Wang and Y. Hu, "PROFS-Performance-Oriented Data Reorganization for Log-Structured File System on Multi-Zone Disks," *Proc. Ninth Int'l Symp. Modeling, Analysis and Simulation of Computer and Telecomm. Systems*, pp. 285-292, 2001.
[19] D. Bacon, "File System Measurements and Their Application to the Design of Efficient Operation Logging Algorithms," *Proc. 10th Symp. Reliable Distributed Systems*, pp. 21-30, Sept. 1991.
[20] S. Forrest, S. Hofmeyr, A. SoMayaji, and T. Longstaff, "A Sense of Self for Unix Processes," *Proc. IEEE Symp. Security and Privacy*, pp. 120-128, May 1996.
[21] C. Feng, J. Peng, H. Qiao, and J.W. Rozenblit, "Alert Fusion for a Computer Host Based Intrusion Detection System," *Proc. 14th Ann. IEEE Int'l Conf. Workshops on the Eng. Computer-Based Systems (ECBS '07)*, pp. 433-440, Mar. 2007.
[22] A. Siraj, R. Vaughn, and S. Bridges, "Intrusion Sensor Data Fusion in an Intelligent Intrusion Detection System Architecture," *Proc. 37th Ann. Hawaii Int'l Conf. System Sciences*, p. 10, Jan. 2004.
[23] X. Hoang and J. Hu, "An Efficient Hidden Markov Model Training Scheme for Anomaly Intrusion Detection of Server Applications Based on System Calls," *Proc. IEEE Int'l Conf. Networks (ICON 2004)*, vol. 2, pp. 470-474, Nov. 2004.
[24] C. Raman and A. Negi, "A Hybrid Method to Intrusion Detection Systems Using HMM," *Proc. Second Int'l Conf. Distributed Computing and Internet Technology*, pp. 389-396, http://dx.doi.org/10.1007/11604655_44, 2005.
[25] S. Forrest, S.A. Hofmeyr, and A. SoMayaji, "Computer Immunology," *Comm. ACM*, vol. 40, no. 10, pp. 88-96, http://doi.acm.org/10.1145/262793.262811, Oct. 1997.
[26] S.A. Hofmeyr, S. Forrest, and A. SoMayaji, "Intrusion Detection Using Sequences of System Calls," *J. Computer Security*, vol. 6, no. 3, p. 151, http://search.ebscohost.com/login.aspx?direct=true&db=tsh&AN=1531432&site=ehost-live, 1998.
[27] X.D. Hoang, J. Hu, and P. Bertok, "A Multi-Layer Model for Anomaly Intrusion Detection Using Program Sequences of System Calls," *Proc. 11th IEEE Int'l. Conf. Networks*, pp. 531-536, 2003.
[28] S.-B. Cho and H.-J. Park, "Efficient Anomaly Detection by Modeling Privilege Flows Using Hidden Markov Model," *Computers and Security*, vol. 22, no. 1, pp. 45-55, 2003.
[29] J. Hu, X. Yu, D. Qiu, and H.-H. Chen, "A Simple and Efficient Hidden Markov Model Scheme for Host-Based Anomaly Intrusion Detection," *IEEE Network*, vol. 23, no. 1, pp. 42-47, Jan./Feb. 2009.
[30] P. Lichodzijewski, A. Nur Zincir-Heywood, and M. Heywood, "Host-Based Intrusion Detection Using Self-Organizing Maps," *Proc. Int'l Joint Conf. Neural Networks (IJCNN '02)*, vol. 2, pp. 1714-1719, 2002.
[31] U. Ahmed and A. Masood, "Host Based Intrusion Detection Using RBF Neural Networks," *Proc. Int'l Conf. Emerging Technologies (ICET '09)*, pp. 48-51, Oct. 2009.
[32] M. Abdel Azim, A.I. Abdel Fatah, and M. Awad, "Performance Analysis of Artificial Neural Network Intrusion Detection Systems," *Proc. Int'l Conf. Electrical and Electronics Eng. (ELECO '09)*, pp. 385-389, 2009.
[33] N. Ye, S. Emran, Q. Chen, and S. Vilbert, "Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection," *IEEE Trans. Computers*, vol. 51, no. 7, pp. 810-820, July 2002.
[34] R. Grishman, *Computational Linguistics: An Introduction*. Cambridge Univ. Press, 1986.
[35] E. Charniak, *Statistical Language Learning*. MIT Press, 1993.
[36] D. Wagner and P. Soto, "Mimicry Attacks on Host-Based Intrusion Detection Systems," *Proc. Ninth ACM Conf. Computer and Comm. Security (CCS '02)*, pp. 255-264, http://doi.acm.org/10.1145/586110.586145, 2002.
[37] D. Bruschi, L. Cavallaro, and A. Lanzi, "An Efficient Technique for Preventing Mimicry and Impossible Paths Execution Attacks," *Proc. IEEE Int'l Performance, Computing, and Comm. Conf. (IPCCC '07)*, pp. 418-425, Apr. 2007.
[38] D. Gao, M. Reiter, and D. Song, "Beyond Output Voting: Detecting Compromised Replicas Using HMM-Based Behavioral Distance," *IEEE Trans. Dependable and Secure Computing*, vol. 6, no. 2, pp. 96-110, Apr.-June 2009.
[39] U. Larson, D. Nilsson, E. Jonsson, and S. Lindskog, "Using System Call Information to Reveal Hidden Attack Manifestations," *Proc. First Int'l Workshop Security and Comm. Networks (IWSCN)*, pp. 1-8, May 2009.

[40] S. Bhatkar, A. Chaturvedi, and R. Sekar, "Dataflow Anomaly Detection," *Proc. Symp. Security and Privacy*, May 2006.
[41] H. Gunes Kayacik, A. Zincir-Heywood, M. Heywood, and S. Burschka, "Generating Mimicry Attacks Using Genetic Programming: A Benchmarking Study," *Proc. IEEE Symp. Computational Intelligence in Cyber Security (CICS '09)*, pp. 136-143, Apr. 2009.
[42] H. Kayacik, A. Zincir-Heywood, and M. Heywood, "Automatically Evading IDS Using GP Authored Attacks," *Proc. IEEE Symp. Computational Intelligence in Security and Defense Applications (CISDA '07)*, pp. 153-160, Apr. 2007.
[43] A. Liu, X. Jiang, J. Jin, F. Mao, and J. Chen, "Enhancing System-Called-Based Intrusion Detection with Protocol Context," *Proc. Fifth Int'l Conf. Emerging Security Information, Systems and Technologies (SECURWARE '11)*, pp. 103-108, Aug. 2011.
[44] G. Creech and F. Jiang, "The Application of Extreme Learning Machines to the Network Intrusion Detection Problem," *AIP Conf. Proc.*, vol. 1479, no. 1, pp. 1506-1511, http://link.aip.org/link/?APC/1479/1506/1, 2012.
[45] C. Cheng, W.P. Tay, and G.-B. Huang, "Extreme Learning Machines for Intrusion Detection," *Proc. Int'l Joint Conf. Neural Networks (IJCNN)*, pp. 1-8, June 2012.
[46] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternative Data Models," *Proc. IEEE Symp. Security and Privacy*, pp. 133-145, 1999.
[47] S.A. Hofmeyr and S. Forrest, "Architecture for an Artificial Immune System," *Evolutionary Computation*, vol. 8, no. 4, pp. 443-473, 2000.
[48] R.K. Gunningham, R.P. Lippmann, D.J. Fried, S.L. Garfinkel, I. Graf, K.R. Kendall, S.E. Webster, D. Wyschogrod, and M.A. Zissman, "KDD98 Intrusion Detection Dataset," *Proc. Int'l Conf. Knowledge Discovery and Data Mining '98)*, http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998data.html, 1998.
[49] A.P. Bradley, "The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145-1159, http://www.sciencedirect.com/science/article/pii/S0031320396001422, 1997.
[50] X.D. Hoang, J. Hu, and P. Bertok, "A Program-Based Anomaly Intrusion Detection Scheme Using Multiple Detection Engines and Fuzzy Inference," *J. Network and Computer Applications*, vol. 32, no. 6, pp. 1219-1228, http://www.sciencedirect.com/science/article/pii/S108480450900071X, 2009.
[51] A. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Information Theory*, vol. 13, no. 2, pp. 260-269, Apr. 1967.
[52] Computer Science Department, "University of New Mexico Intrusion Detection Dataset," http://www.cs.unm.edu/~immsec/systemcalls.htm, 2012.
[53] "Ubuntu Linux," http://www.ubuntu.com, May 2012.
[54] "The Apache Software Foundation," http://apache.org, May 2012.
[55] "PHP: Hypertext Processor," http://www.php.net, May 2012.
[56] "TikiWiki: CMS Groupware," http://info.tiki.org/Tiki+Wiki+CMS+Groupware, May 2012.
[57] "Tiki Wiki CMS Groupware Remote PHP Code Injection." 2012.
[58] "Offensive Security Certified Professional," http://www.offensive-security.com/information-security-certifications/oscp-offensive-security-certified-professional/, Nov. 2012.
[59] "Metasploit Penetration Testing Software," http://www.metasploit.com, Apr. 2012.
[60] W. Lee, S. Stolfo, and K. Mok, "A Data Mining Framework for Building Intrusion Detection Models," *Proc. IEEE Symp. Security and Privacy*, pp. 120-132, 1999.
[61] D. Yeung and Y. Ding, "Host-Based Intrusion Detection Using Dynamic and Static Behavioural Models," *Pattern Recognition*, vol. 36, no. 1, pp. 229-243, 2003.
[62] W.-H. Chen, S.-H. Hsu, and H.-P. Shen, "Application of SVM and ANN for Intrusion Detection," *Computers and Operations Research*, vol. 32, no. 10, pp. 2617-2634, http://www.sciencedirect.com/science/article/pii/S0305054804000711, 2005.
[63] A. Sharma, A.K. Pujari, and K.K. Paliwal, "Intrusion Detection Using Text Processing Techniques with a Kernel Based Similarity Measure," *Computers and Security*, vol. 26, no. 7, pp. 488-495, http://www.sciencedirect.com/science/article/pii/S0167404807001113, 2007.
[64] S. Rawat, V. Gulati, and A. Pujari, "A Fast Host-Based Intrusion Detection System Using Rough Set Theory," *Trans. Rough Sets IV*, pp. 144-161, Springer, 2005.
[65] M.R. Norouzian and S. Merati, "Classifying Attacks in a Network Intrusion Detection System Based on Artificial Neural Networks," *Proc. 13th Int'l Conf. Advanced Comm. Technology (ICACT)*, pp. 868-873, 2011.
[66] G. Song, J. Zhang, and Z. Sun, "The Research of Dynamic Change Learning Rate Strategy in BP Neural Network and Application in Network Intrusion Detection," *Proc. Third Int'l Conf. Innovative Computing Information and Control (ICICIC '08)*, pp. 513-513, 2008.
[67] Z. Linhui, F. Xin, and D. Yaping, "A Novel Adaptive Intrusion Detection Approach Based on Comparison of Neural Networks and Idiotypic Networks," *Proc. Second Int'l Workshop Nonlinear Dynamics and Synchronization (INDS '09)*, pp. 203-208, 2009.

Gideon Creech received the master's of engineering science degree from the University of New South Wales (UNSW). He is currently working toward the PhD degree at the University of New South Wales. He is a lieutenant-commander in the Royal Australian Navy. His research interests include cyber warfare, hacking, intrusion detection systems, and computer security. He is a student member of the IEEE.

Jiankun Hu received the BE degree from Hunan University, China, in 1983, the PhD degree in control engineering from the Harbin Institute of Technology, China, in 1993, and the master's degree in research in computer science and software engineering from Monash University, Australia, in 2000. He is a full professor and research director of the Cyber Security Lab, School of Engineering and IT, University of New South Wales at the Australian Defence Force Academy (UNSW@ADFA), Canberra, Australia. He has worked at Ruhr University Germany on the prestigious German Alexander von Humboldt Fellowship 1995-1996; research fellow at Delft University of the Netherlands 1997-1998, and research fellow at Melbourne University, Australia 1998-1999. His main research interests include the field of cyber security including biometrics security where he has published many papers in high-quality conferences and journals including *IEEE Transactions on Pattern Analysis and Machine Intelligence*. He has served on the editorial boards of up to seven international journals and served as security symposium chair of IEEE flagship conferences of IEEE ICC and IEEE Globecom. He has received seven ARC (Australian Research Council) Grants and is now serving on the prestigious Panel of Mathematics, Information and Computing Sciences (MIC), and ARC ERA (The Excellence in Research for Australia) Evaluation Committee.