An Anomaly Detection System based on Ensemble of Detectors with Effective Pruning Techniques

Amirreza Soudi, Wael Khreich, and Abdelwahab Hamou-Lhadj
Software Behaviour Analysis (SBA) Research Lab, Department of Electrical and Computer Engineering,
Concordia University, Montreal, QC, Canada
Email: {am_soudi, wkhreich, abdelw}@ece.concordia.ca

***Abstract*—Anomaly detection systems rely on machine learning techniques to model the normal behavior of the system. This model is used during operation to detect anomalies due to attacks or design faults. Ensemble methods have been used to improve the overall detection accuracy by combining the outputs of several accurate and diverse models. Existing Boolean combination techniques either require an exponential number of combinations or sequential combinations that grow linearly with the number of iterations, which make them difficult to scale up and analyze. In this paper, we propose PBC (Pruning Boolean Combination), an efficient approach for selecting and combining anomaly detectors. PBC relies on two novel pruning techniques that we have developed to aggressively prune redundant and trivial detectors. Compared to existing work, PBC reduces significantly the number of detectors to combine, while keeping similar accuracy. We show the effectiveness of PBC when applying it to a large dataset.**

***Keywords*—Intrusion Detection Systems; Anomaly Detection Systems; Multiple-Detector Systems; Boolean Combination; Pruning Techniques**

I. Introduction

Intrusion Detection Systems (IDSs) have become important tools that help security administrators to identify and analyze unauthorized computer or network activities, from both outsider and insider attacks. The dominant detection methodologies are signature-based and anomaly-based approaches. Signature-based (or misuse) IDSs look for events that correspond to patterns of known attacks. They can provide a high level of accuracy, however are limited to detecting known attacks and vulnerable to polymorphic attacks (which are capable of changing their signatures as they propagate).

Anomaly detection systems (ADSs), on the other hand, monitor for significant deviations from normal system behavior. They are typically trained (using machine learning and data mining techniques) on datasets collected over a period of attack-free activities to learn the normal system behavior, and then deployed to detect deviations from the expected system behavior. These deviations are reported as anomalous events, although they are not necessarily malicious activities or attacks because they may also correspond to coding or configuration errors. An ADS can detect novel attacks (not known during training time) but generate large number of false alarms, because it is difficult to obtain complete descriptions of complex system behavior, which may change over time.

IDSs can also be categorized depending on their location of deployment into network- and host-based detection systems (a more comprehensive taxonomy of IDSs can be found in [30]). Network-based IDSs monitor and analyze traffic to and from all devices on the network. They may be be located anywhere in the network; integrated in network devices (e.g., switches and routers) or as a standalone device. A host-based IDS runs on a host computer and monitors sensitive activities on this host, such as unauthorized access, modification of files, or system calls.

In this work, we focus on host-based anomaly detection using system calls – the gateway between user and kernel mode. Short sequences of system calls have been shown consistent with normal host operation, and can be used to detect attacks [11], [41]. A large number of research studies have investigated different machine learning and data mining techniques for detecting anomalies in system call sequences [12]. Among these, techniques based on Hidden Markov Models (HMMs) – probabilistic models for sequential data – have been shown to produce a high level of detection accuracy [8], [13], [17], [18], [20], [33], [37], [39], [41], [43].

The success of an ADS depends largely on the model of normal behavior. A single HMM may not, however, provide adequate approximation of the underlying data distribution of a complex host system behavior, due to the many local maxima of the likelihood function [20]. Ensemble methods have been used to improve the overall system accuracy by combining the outputs of several *accurate and diverse* models [7], [23], [25], [44]. In particular, combining the outputs from multiple HMMs, each trained with a different number of states, in the Receiver Operating Characteristics (ROC) space according to the Iterative Boolean Combination (IBC), has been shown to provide a significant improvement in the detection accuracy of system call anomalies [21].

The IBC is a general decision-level combination technique that attempts to select the decision thresholds (from each input detector) and the Boolean functions that maximize the overall ROC convex hull of the combined ensemble [21]. As presented in Algorithm 1, given K soft detectors that assign scores or probabilities to the input samples, (which can be converted to a crisp detector by setting a threshold on the scores as further detailed in section III) , the IBC algorithm starts by combining the outputs of the first two detectors (in the order of input), and then proceeds *sequentially* by combining the resulting combinations with the outputs of third detector, and so on, until the K^{th} detector is combined. It can then re-iterate to combine the resulting combinations with the original detectors.

The IBC technique provides a practical way for combining

relatively large number of detectors while avoiding the exponential explosion of Boolean combinations. As detailed in Section IV-D, applying all Boolean functions using an exhaustive brute-force search to determine optimal combinations leads to an exponential number of combinations, which is prohibitive even for a small number of detectors [1]. Even the *pairwise* Bruteforce Boolean Combination (BBC2), which is used as a baseline reference in our experiments, requires an exponential number of combinations, which is equal to the square of the number of detectors (see Algorithm 2 without the pruning mechanisms).

However, the sequential combinations of soft detectors according to IBC still faces challenges, when a large number of detectors (K) is presented for combinations. First, it produces a sequence of combination rules that grows linearly with K (and the number of iterations), which becomes difficult to analyse and understand. Furthermore, it makes the algorithm sensitive to the order in which the detectors are input for combinations, which increases the effort required to find best subset for operations. In any case, combining all available detectors without any pruning mechanism may be inefficient due to the redundancy in their outputs.

In this work, we propose an ADS based on a Pruned Boolean Combination (PBC) algorithm, which employs two novel pruning techniques to select a subset of diverse and accurate detectors for combination, while discarding the remaining ones. Although both pruning techniques are based on Cohen's Kappa [5] measure, they differ in the way they compute the diversity and accuracy of the selected detectors for combinations. While the first technique relies only on Kappa measure (MinMax-Kappa), the second uses both Kappa and the ROC convex hull (ROCCH-Kappa). Therefore, our proposed PBC approach provides an efficient way to prune and combine large number of detectors, which avoids the exponential explosion of combinations of brute-force techniques, and reduces the sequence of combinations provided by IBC.

We evaluate our PBC-based ADS using ADFA Linux Dataset (ADFA-LD)[1], which has been recently made publicly available on the website of the University of New South Wales [6]. The performance of the PBC using both pruning techniques are compared to that of IBC and BBC2 in terms of ROC analysis and time complexity. The results show that PBC with both pruning techniques are capable of maintaining similar overall accuracy as measured by the ROC curves to that of IBC and BBC2. However, the time required for searching and selecting (or pruning) the subset of detectors from the same ensemble of detectors is on average three magnitudes lower for PBC with MinMax-Kappa pruning than that of BBC2. Furthermore, the experimental results show that PBC with both pruning techniques always provides two crisp detectors for combination (during the operations), while IBC provided an average of 11 detectors to achieve the same operating point.

The remaining of this paper is organized as follows. The next section reviews anomaly detection systems based on system call traces with a focus on hidden Markov models detectors. In Section III, we provide background information about ROC analysis and a summary of the Boolean combination techniques in the ROC space. Section IV presents our novel PBC approach for pruning and combination of anomaly detectors, describes our efficient pruning techniques, and provides a comparative complexity analysis for all Boolean combination techniques discussed in the paper. Section V describes the dataset, the experimental protocol, and the evaluation metrics used in our experiments. The results are provided and discussed in Section VI, followed by the threats to validity in Section VII, as well as the conclusion and future work in Section VIII.

[1] http://www.cybersecurity.unsw.adfa.edu.au/ADFA IDS Datasets/

II. HMM-based Anomaly Detection using System Call Sequences

The temporal order of system calls has been shown consistent with the normal behavior of a privileged process, while an unusual burst will occur during an attack [11], [41]. The authors proposed a host-based ADS using a simple sequence matching techniques for detecting anomalous system call sequences generated by UNIX privileged processes [11]. System call traces provided for training are first segmented into fixed-length *contiguous* sequences, using a fixed-size sliding window, shifted by one symbol to build the normal profile. These sequences are then stored in a database that represents the normal process behavior. During operations, a sliding window (having the same size used to construct the normal profile) to scan the system calls generated by the monitored process for anomalies – sequences that are not found in the normal database.

Several statistical and machine learning techniques have been investigated over the last two decades for detecting system call anomalies [12]. Application of machine learning techniques include neural network [15], k-nearest neighbors [29], Markov models or n-grams [19], [31], Bayesian models [24]. Among these, techniques based on discrete HMMs have been shown to produce a high level of detection accuracy [4], [8], [13], [14], [17], [18], [20], [39]–[41], [43].

A discrete HMM is a stochastic process for sequential data [22], [36]. An HMM is determined by two interrelated mechanisms – a latent Markov chain having a finite number of states N, and a set of observation probability distributions, each one associated with a state. Starting from an initial state $S_i \in \{S_1, ..., S_N\}$, determined by the initial state probability distribution π_i, at each discrete-time instant, the process transits from state S_i to state S_j according to the transition probability distribution a_{ij}. The process then emits a symbol v_k, from a finite alphabet $V = \{v_1, \ldots, v_M\}$ of size M symbols, according to the discrete-output probability distribution $b_j(v_k)$ of the current state S_j. HMM is commonly parametrized by $\lambda = (\pi, A, B)$, where the vector $\pi = \{\pi_i\}$ is the initial state probability distribution, matrix $A = \{a_{ij}\}$ is the state transition probability distribution, and matrix $B = \{b_j(v_k)\}$ is the state output probability distribution, ($1 \leq i, j \leq N$ and $1 \leq k \leq M$).

A well trained HMM provides a compact detector that captures the underlying structure of a process based on the temporal order of system calls, and detects deviations from normal system call sequences with high accuracy and tolerance to noise. Training an HMM from a sequence (or a block) of observation symbols, $o_{1:T}$, aims at estimating HMM parameters λ to best fit the training data. Typically, parameters

estimation consists of maximizing the likelihood of the training data over HMM parameters space, $P(o_{1:T} \mid \lambda)$. Since this likelihood depends on the latent states, there is no known analytical solution to the learning problem. Iterative optimization techniques, such as the Baum-Welch (BW) algorithm [2], are applied to estimate the HMM parameters over several training iterations, until the likelihood function is maximized. During operation, the likelihood of a new observation sequence $o_{1:T}$ given a trained HMM λ, $P(o_{1:T} \mid \lambda)$ is typically evaluated by using the Forward-Backward (FB) algorithm [22], [36]. Setting a threshold on the output probabilities of HMM, provides a decision whether the sequence is normal or anomalous.

Few researchers tried to investigate the effect of the number of states on the performance of HMM detectors, and found that N value may have a great impact on the overall performance [20], [42]. In particular, combining the outputs from multiple HMMs, each trained with a different N value, in the ROC space according to the IBC technique (as described in the next section), has been shown to provide a significant improvement in the detection accuracy of system call anomalies [21].

III. Boolean Combination of Detectors in the ROC Space

This section provides information about ROC analysis and summarizes the Boolean combination in the ROC space [21]. A crisp detector outputs a decision or a class label (e.g., normal or anomaly) while a soft detector such as HMM assigns scores to the input samples, which can be converted to a crisp detector by setting a decision threshold on the scores. Given the responses of a crisp detector on a validation set, the true positive rate (tpr) is the proportion of positives correctly classified over the total number of positive samples. The false positive rate (fpr) is the proportion of negatives incorrectly classified over the total number of negative samples. The positive (or target) class is typically the class of interest, which is the anomalous class for an ADS.

A ROC curve is a plot of tpr against fpr [9]. A crisp detector produces a single data point in the ROC space, while a soft detector produces a ROC curve by varying the decision thresholds. In practice, an empirical ROC plot is obtained by connecting the observed (tpr, fpr) pairs of a soft detector at each decision threshold. A point a is *superior* to another point b in the ROC space, if $fpr(a) \leq fpr(b)$ and $tpr(a) \geq tpr(b)$. The ROC convex hull (ROCCH) is therefore the outer envelope connecting superior points in the ROC space. A ROC curve allows to visualize the performance of detectors and to select optimal operational points, without committing to a single decision threshold or to fixed error costs. The area under the ROC curve (AUC), or under the ROCCH, provides a general measure for evaluation and selection of detectors [9].

The IBC is a general decision-level combination technique that attempts to select the decision thresholds (from each input detector) and the Boolean functions that maximize the overall ROCCH of the combined ensemble [21]. The core of IBC (only for the first iteration) is described in Algorithm 1.

The IBC applies each Boolean function to combine the responses corresponding to each decision threshold from the first detector to those from the second detector. Fused responses are then mapped to vertices in the ROC space, and their ROC convex hull (ROCCH) is computed. Vertices that are superior to the ROCCH of original detectors are then selected. The set ($\mathcal{S}$) of decision thresholds from each detector and Boolean functions corresponding to these vertices is stored, and the ROCCH is updated to include emerging vertices. The responses corresponding to each decision threshold from the third detector are then combined with the responses of each emerging vertex, and so on, until the last detector in the pool is combined. The BC technique yields a final ROCCH for visualization and selection of operating points, and the set of selected thresholds and Boolean functions, $\mathcal{S}$, for each vertex on the composite ROCCH to be applied during operations. Although not shown in Algorithm 1, the original IBC algorithm can iterate by re-combining the resulting combination (of all detectors) on the ROCCH with each of the original detectors (sequentially) until the ROCCH stops improving [21].

However, as stated previously, combining all available detectors without pruning may be inefficient due to the redundancy in their outputs. The sequential combinations of soft detectors, illustrated in the loop in line 16 of Algorithm 1, produces a sequence of combination rules that grows linearly with K (and the number of iterations), which becomes difficult to track, analyse and understand when the value of K becomes

Algorithm 1: IBC$(D_1, D_2, \ldots, D_K, \mathcal{V})$: Iterative Boolean Combination

input : K soft detectors $(D_1, D_2, \ldots, D_K)$ and a validation set $\mathcal{V}$ of size $|\mathcal{V}|$
output: ROCCH of combined detectors.
- Each vertex is the result of 2 to K combination of crisp detectors.
- Each combination selects the best decision thresholds from different detectors (D_i, t_j) and Boolean function (stored in the set $\mathcal{S}$)

1 $n_k \leftarrow$ number of decision thresholds of D_k using $\mathcal{V}$ // num. of vertices on ROC(D_k).
2 $BooleanFunctions \leftarrow \{a \wedge b, \neg a \wedge b, a \wedge \neg b, \neg(a \wedge b), a \vee b, \neg a \vee b, a \vee \neg b, \neg(a \vee b), a \oplus b, a \equiv b\}$
3 **compute** ROCCH$_1$ of the first two detectors (D_1 and D_2)
4 **allocate** $\boldsymbol{F}$ an array of size: $[2, n_1 \times n_2]$ // temporary storage of combination results.
5 **foreach** $bf \in BooleanFunctions$ **do**
6 **for** $i \leftarrow 1$ **to** n_1 **do**
7 $\boldsymbol{R_1} \leftarrow (D_1, t_i)$ // responses of D_1 at decision threshold t_i using $\mathcal{V}$.
8 **for** $j \leftarrow 1$ **to** n_2 **do**
9 $R_2 \leftarrow (D_2, t_j)$ // responses of D_2 at decision threshold t_j using $\mathcal{V}$.
10 $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_1}, \boldsymbol{R_2})$ // combine responses using current Boolean func.
11 **compute** (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$ // map combination to ROC plane
12 **push** (tpr, fpr) onto $\boldsymbol{F}$
13 **compute** *ROCCH$_2$ of all ROC points in* $\boldsymbol{F}$
14 $n_{ev} \leftarrow$ number of emerging vertices
15 $\mathcal{S}_2 \leftarrow \{(D_1, t_i), (D_2, t_j), bf\}$ // set of selected decision thresholds from each detector and Boolean functions for emerging vertices.
16 **for** $k \leftarrow 3$ **to** K **do**
17 **allocate** $\boldsymbol{F}$ of size: $[2, n_k \times n_{ev}]$
18 **foreach** $bf \in BooleanFunctions$ **do**
19 **for** $i \leftarrow 1$ **to** n_{ev} **do**
20 $\boldsymbol{R_i} \leftarrow S_{k-1}(i)$ // responses from previous combinations.
21 **for** $j \leftarrow 1$ **to** n_k **do**
22 $\boldsymbol{R_k} \leftarrow (D_k, t_j)$
23 $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_i}, \boldsymbol{R_k})$
24 **compute** (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$
25 **push** (tpr, fpr) onto $\boldsymbol{F}$
26 **compute** *ROCCH$_k$ of all ROC points in* $\boldsymbol{F}$
27 $n_{ev} \leftarrow$ number of emerging vertices
28 $\mathcal{S}_k \leftarrow \{\mathcal{S}_{k-1}(i), (D_k, t_j), bf\}$
 // $\mathcal{S}_k$ is the set of the selected subsets from the previous combinations; the decision thresholds from the newly-combined detector; and the Boolean functions that yields to the emerging vertices on the ROCCH.
29 **store** $\mathcal{S}_k : 2 \leq k \leq K$
30 **return** *ROCCH$_K$*

Algorithm 2: PBC($D_1, D_2, \ldots, D_K, \mathcal{V}$): Pruned Boolean Combination

input : K soft detectors $(D_1, D_2, \ldots, D_K)$ and a validation set $\mathcal{V}$ of size $|\mathcal{V}|$
output: ROCCH of combined detectors.
- Each vertex is the result of exact 2 combination of crisp detectors.
- Each combination selects the best decision thresholds from different detectors (D_i, t_j) and Boolean function (stored in the set $\mathcal{S}$)

$n_k \leftarrow$ number of decision thresholds of D_k using $\mathcal{V}$ // num. of vertices on ROC(D_k).
let $n = \sum_{k=1}^{K} n_k$
$BooleanFunctions \leftarrow \{a \wedge b, \neg a \wedge b, a \wedge \neg b, \neg(a \wedge b), a \vee b, \neg a \vee b, a \vee \neg b, \neg(a \vee b), a \oplus b, a \equiv b\}$
allocate $\boldsymbol{C}$ an array of size: $[|\mathcal{V}|, n]$ // storage of all crisp detectors' decisons.
convert soft detectors to crisp detectors
for $i \leftarrow 1$ **to** K **do**
  for $j \leftarrow 1$ **to** n_i **do**
    $\boldsymbol{R} \leftarrow (D_i, t_j)$ // responses of D_i at decision threshold t_j using $\mathcal{V}$.
    push $\boldsymbol{R}$ onto $\boldsymbol{C}$

choose Pruning Technique {MinMax-Kappa, ROCCH-Kappa}
reduce n to U // $U \ll n$: is a user defined max number of detectors
return $\boldsymbol{C}_{\text{selected}} \leftarrow \boldsymbol{C}$ - Pruned Detectors
// Subset of size U detectors selected from all original detectors and returned for combination

allocate $\boldsymbol{F}$ an array of size: $[2, U^2 \times size(BooleanFunctions)]$
// temporary storage of combination results.
foreach $bf \in BooleanFunctions$ **do**
  for $i \leftarrow 1$ **to** U **do**
    $\boldsymbol{R_1} \leftarrow \boldsymbol{C}_{\text{selected}}[i]$ // Retrieve Decision Vector
    for $j \leftarrow 1$ **to** U **do**
      $\boldsymbol{R_2} \leftarrow \boldsymbol{C}_{\text{selected}}[j]$
      $\boldsymbol{R_c} \leftarrow bf(\boldsymbol{R_1}, \boldsymbol{R_2})$ // combine responses using current Boolean func.
      compute (tpr, fpr) of $\boldsymbol{R_c}$ using $\mathcal{V}$ // map combination to ROC plane
      push (tpr, fpr) onto $\boldsymbol{F}$

compute *ROCCH of all ROC points in* $\boldsymbol{F}$
$n_{ev} \leftarrow$ number of emerging vertices
$\mathcal{S} \leftarrow \{(D_1, t_i), (D_2, t_j), \ldots, (D_k, t_k), bf\}$ // set of selected decision thresholds from each detector and Boolean functions for emerging vertices.
store $\mathcal{S}$; **return** *ROCCH*

large. In addition, the IBC algorithm is sensitive to the order in which the detectors are input for combinations, which increases the effort required to find best subset for operations.

IV. Pruning Boolean Combination (PBC) Approach

In this section we describe our Pruned Boolean Combination (PBC) algorithm, which is based on two novel pruning techniques that select a subset of diverse and accurate detectors for combination, while discarding the remaining ones. PBC can avoid both the exponential explosion of combinations provided by the brute-force approach and the sequential combination of IBC. The main difference is that the PBC algorithm proceeds by thresholding all available soft detectors into crisp ones. This step is, in fact, not essential for PBC because thresholding could be done outside the algorithm and the crisp detectors are directly input for pruning. In contrast with IBC, all available crisp detectors are input to one of the proposed pruning techniques, MinMax-Kappa or ROCCH-Kappa described in Section IV-B and IV-C, to select the best subset of crisp detectors for Boolean combination and control its size U (i.e., the number of combined detectors). Without pruning brute-force pairwise combination of all available detectors may not be feasible for large number of detectors, $\mathcal{O}(N^2)$ (where N is number of available detectors), for further details see Section IV-D.

The boost in performance achieved with an ensemble of detectors is often attributed to the concept of diversity. While it is generally accepted that an ensemble should contain diverse models to improve the performance, there is no clear definition of diversity neither a consensus about the measure of diversity and its computation. In practice, several measures of diversity have been proposed to quantify the level of agreement or measure the dependency among the ensemble members [26].

A. Kappa Measure

Cohen's Kappa statistic (or simply Kappa hereafter) is one of the well-known and widely used measure of agreement between raters [5]. Kappa has lately gained some popularity for ensemble combination, especially the Kappa-error diagrams which help visualizing individual accuracy and diversity in a two dimensional plot [27], [32]. Our pruning techniques, described in the next sections, are based on Kappa and inspired by the Kappa-error diagram only for visualization of Kappa against the false positives and true positives (as shown in Figures 1 and 2).

TABLE I: Contingency table between two detector decisions

	D_2 correct	D_2 wrong
D_1 correct	a	b
D_1 wrong	c	d

Consider the contingency table of two detectors, D_1 and D_2, presented in Table I, where, for instance, a is the number of examples on which both detectors agree. The sum of all element in Table I is equal to the size of validation set, $a + b + c + d = |\mathcal{V}|$. The Kappa ($\kappa$) measure between two detectors is therefore computed based on the element of the contingency table according to Equation 1.

$$\kappa = \frac{2(ad - bc)}{(a+b)(b+d) + (a+c)(c+d)} \tag{1}$$

Kappa takes on values between -1 and 1; lower values means a high level of disagreement or more diverse opinions, while higher values indicate a high level of agreement or similarity in responses between detectors. When both detectors provide the same vector of decisions (they agree on every example) then $\kappa = 1$. On the other hand, when $\kappa = 0$ the detectors are independent (any agreement is totally due to chance). Negative Kappa values can be interpreted as both detectors agrees less than what would be expected just by chance. More importantly, negative values account for negative correlations, which can be useful for combination, but this rarely occurs in practice [32].

B. MinMax-Kappa Pruning

The proposed pruning technique starts by computing the Kappa values between each detector's decision vector and the true decision labels (or ground truth), and then sorting them in ascending order. Detectors with large Kappa values ($\kappa \approx \kappa_{max}$) are accurate and hence should be selected; however, they provide less diverse decisions among themselves. Therefore, the technique attempts to select complementary detectors by choosing those with Kappa values close to $\kappa \approx \kappa_{min}$. In practice, κ_{max} and κ_{min} values depend on the data and the

detectors. Trivial detectors (providing always either positive or negative decisions) also reside at $\kappa_{min} \approx 0$. In such cases, these are filtered out before selected the complementary detectors. When detectors with negative Kappa values exist, it is always beneficial to select them since they provide detectors with negative correlations or complementary errors compared those with κ_{max}.

The MinMax-Kappa pruning technique takes two parameters, the total number of detectors and the ratio of detectors to be selected close to κ_{max} and κ_{min}. In our experiments, we experimented with different parameters sets and presented the results for 50 selected detectors with a ration of 50% (which means half of the detectors are selected from the region with higher values of Kappa, while the remain half are chosen from the region with lower values). In fact, these are user-defined parameters that are constrained by the resources available during operations. Our experiments showed that the results of the pruning algorithms are not very sensitive to small changes in these parameters

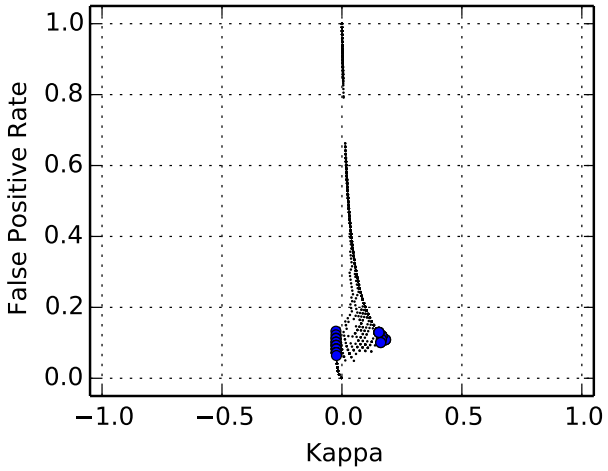
Figure 1 shows an example of the selected detectors from our experiment, according to the MinMax-Kappa technique. The Kappa values (on the X-axis) for the same selected detectors are plotted against false positive (Figure 1a) and true positive (Figure 1b) rates. These points (large, blue) are the selected detectors, $\mathcal{C}_{\text{selected}}$ in Algorithm 2. All remaining detectors (small points) are pruned. For illustration, Figure 3a map the selected point to the ROC space, which could also be compared to our second pruning technique.

C. ROCCH-Kappa Pruning

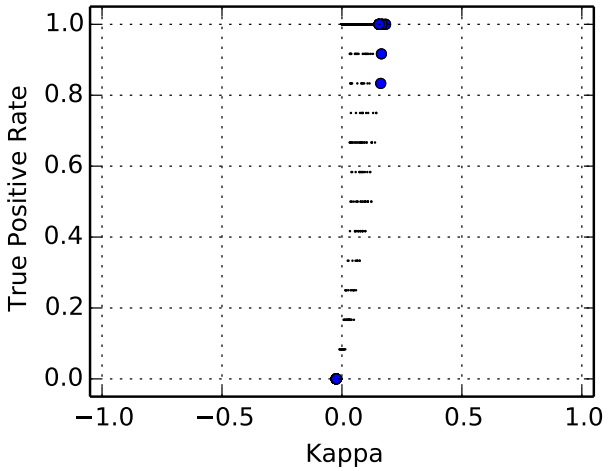
This technique also tries to select accurate detectors and then find a set of detectors with complimentary errors. In contrast with MinMax-Kappa, this technique considers the detectors that are on the facet of the ROCCH in the ROC space. These detectors are selected since they are the most accurate available detectors that covers the whole range of fpr and tpr trade-off. The Kappa measure is then used to select diverse detectors with reference to those on ROCCH. The technique computes the Kappa value of each detector on the ROCCH and all the remaining detectors, sort the resulting values in ascending order. Detectors with larger values of Kappa are discarded, since they provide similar decisions (or errors) to those provided by the select detectors (on the ROCCH).

The most diverse detectors are those that provide negative or close to zero Kappa values ($\kappa \leq 0$), however are not trivial detectors (providing always either positive or negative decisions). The only parameter of the ROCCH-Kappa is the number of detectors to be selected for each detector on the ROCCH. In practice, the number of detectors on the ROCCH is small ([3-20] detectors).

Figure 2 shows an example of the selected detectors from our experiment, according to the ROCCH-Kappa technique, where the Kappa values (on the X-axis) for the selected detectors are plotted against false positive (Figure 2a) and true positive (Figure 2b) rates. Similarly, these selected points (large, blue) are the $\mathcal{C}_{\text{selected}}$ in Algorithm 2, while all remaining detectors (small points) can be pruned. Figure 3b map the detectors selected by ROCCH-Kappa to the ROC



(a) κ-fpr diagram



(b) κ-tpr diagram

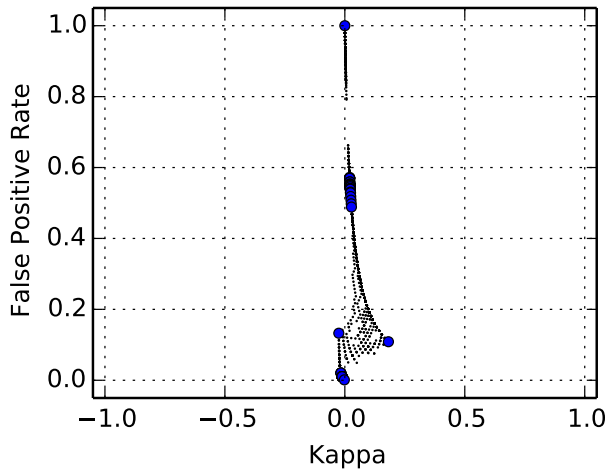
Fig. 1: Illustration of selected detectors (large blue circles) based on MinMax-Kappa pruning technique. All remaining detectors (small black dots) are pruned.

space, which could be compared to those selected by MinMax-Kappa in Figure 3a.

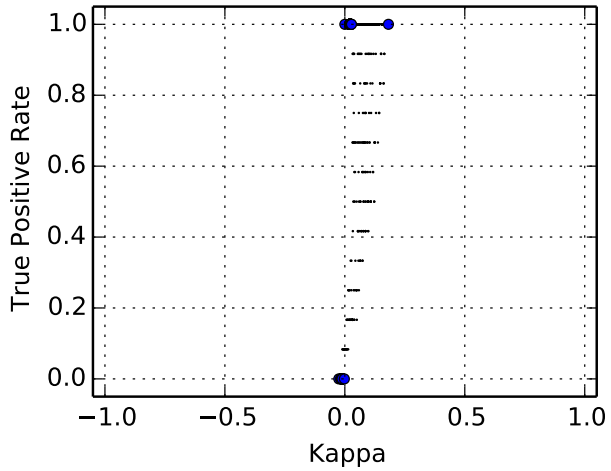
D. Complexity Analysis

Given K soft detectors, let n_i be the number of decision thresholds or crisp detectors produced by each of the soft detector D_i, $i = 1, \ldots, K$, on the validation set $\mathcal{V}$. Let $n = \sum_{i=1}^{K} n_i$ the total number of crisp detectors in the ensembles, and $n_{avg} = n/K$ the average number of crisp detectors produced by soft detectors.

A brute-force search for optimal combination is infeasible in practice due to the doubly exponential combinations. In fact, for n crisp detectors there are 2^n possible outcomes that can be combined in 2^{2^n} ways, which makes the brute-force combination impractical even for small n values [1]. Even only pairwise combination of n crisp detectors, which

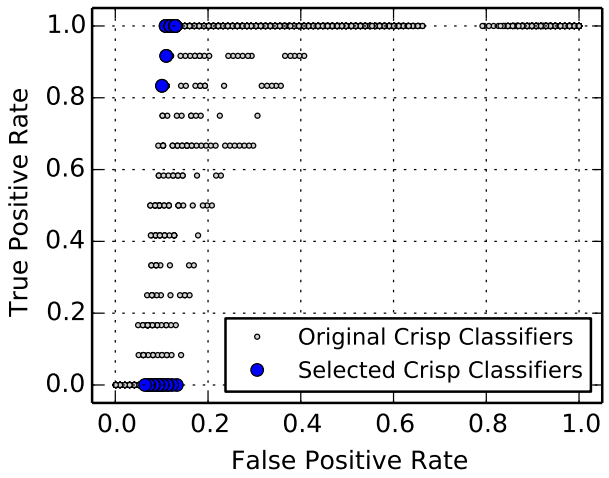


(a) κ-fpr diagram

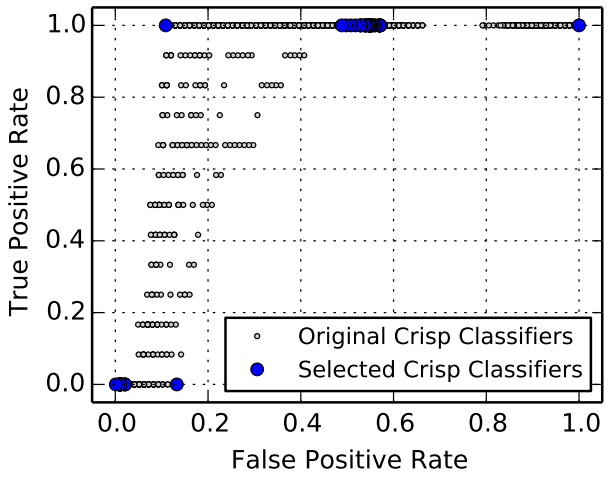


(b) κ-tpr diagram

Fig. 2: Illustration of selected detectors (large blue circles) based on ROCCH-Kappa pruning technique. All remaining detectors (small black dots) are pruned.



(a) MinMax-Kappa pruning



(b) ROCCH-Kappa pruning

Fig. 3: Illustration of the selected detectors for combination mapped onto the ROC space (large blue circles). All other detectors (small black dots) can be pruned.

requires $\mathcal{O}(n^2)$ Boolean operations, may not be feasible in practise for large n values. The sequential combination of the IBC algorithm reduces its worst-case time complexity to $\mathcal{O}(n_{avg}^2 + Kn_{avg})$ Boolean operations.

Both pruning techniques are capable of reducing the size of the selected subset of detectors (for Boolean combination) up to a user defined maximum number (U). The worst-case time complexity required by MinMax-Kappa technique to select U crisp detectors (and prune the rest) is $\mathcal{O}(n(\log n + 1) + U^2)$. It requires about $n(\log n + 1)$ operations for computing and sorting the Kappa values for all crisp detectors, and U^2 for the pairwise Boolean combinations of the U retained detectors. For ROCCH-Kappa technique however the worst-case time complexity is of the order $\mathcal{O}(n(\log n + n_{ev}) + U^2)$, where n_{ev} is the number of emerging vertices which is typically around ten. The additional n_{ev} factor is due to the computation of Kappa is repeated n_{ev} times for each emerging point on the ROCCH. For numerical comparison, Table III shows the average time for each combination and pruning technique used in our experiments.

V. Experiments on System Call Dataset

We evaluate the accuracy and efficiency of our pruning techniques using a modern system call dataset, called ADFA Linux Dataset (ADFA-LD), which has been recently made publicly available on the website of the University of New South Wales [6]. The ADFA-LD dataset is generated by exploiting various security vulnerabilities in a Ubuntu operating system (OS) hosting a web server. The systems consists of a fully patched Ubuntu Linux 11.04 OS with an Apache 2.2.17 web server, PHP 5.3.5 server side scripting engine, TikiWiki 8.1 content management system, FTP server, MySQL 14.14 database management system and an SSH server. First they collect normal system call traces by letting users perform

basic operations, such as web browsing and Latex document preparations under controlled situations. The anomalous system call traces are collected while the system is being under six types of attack vectors resulting in a total of 60 attacks. These attacks were launched by a certified penetration tester against the system and included web-based exploitation, simulated social engineering, poisoned executable, remotely triggered vulnerabilities, remote password brute-force attacks and system manipulation. The authors of the ADFA-LD have organized the dataset into 833 normal traces for training the anomaly detectors, and 4373 normal traces and 60 anomalous traces for testing.

In our experiments we used the following experimental setup. First, we trained 20 HMMs with different numbers of states (i.e., $K = 20$ soft detectors), using the 833 normal traces provided for training in the ADFA-LD dataset. On average, the output of each HMM is thresholded into $n_{avg} = 100$ thresholds or crisp detectors, which provides $n = K \times n_{avg} = 20 \times 100 = 2000$ crisp detectors in the ensemble.

The objective is to choose the most accurate subset from this ensemble for combination while pruning the remaining detectors according to our MinMax-Kappa or ROCCH-Kappa technique. Then, the ROC curves and AUC results are compared to those of IBC and the pairwise Bruteforce Boolean Crisp combination (BBC2). BBC2 is another baseline, which is the PBC Algorithm 2, but without any pruning mechanism.

For evaluation of performance, a 5-fold cross-validation (5FCV) is applied to the test set comprising the 4373 normal and the 60 anomalous traces. Since the number of anomalous traces is relatively small with reference to the normal ones, we applied the cross validation to partition the normal and anomalous sets separately, such that we keep the same ratio (normal to anomalous) and guarantee that all folds include the anomalies. Accordingly, each fold contains 874 traces selected at random from the 4373 normal traces and 12 attacks traces selected at random from the 60 attack traces.

In contrast with the standard way of applying the 5FCV, we used one fold for computing the Boolean combination according to each of the combination technique, and the remaining four folds (i.e., 3498 normal traces and 48 attack traces) are used for evaluating and benchmarking the detection performance. This is because we wanted to test the boost in performance while using a small number of anomalies during combination.

TABLE II: Average AUC values and their standard deviations over the 5FCV for each techniques. Design on one folds and evaluated on four fold.

Method Name	Mean	Std
BBC2	0.97426	0.001
IBC	0.97276	0.001
MinMax-Kappa	0.97074	0.003
ROCCH-Kappa	0.97051	0.003

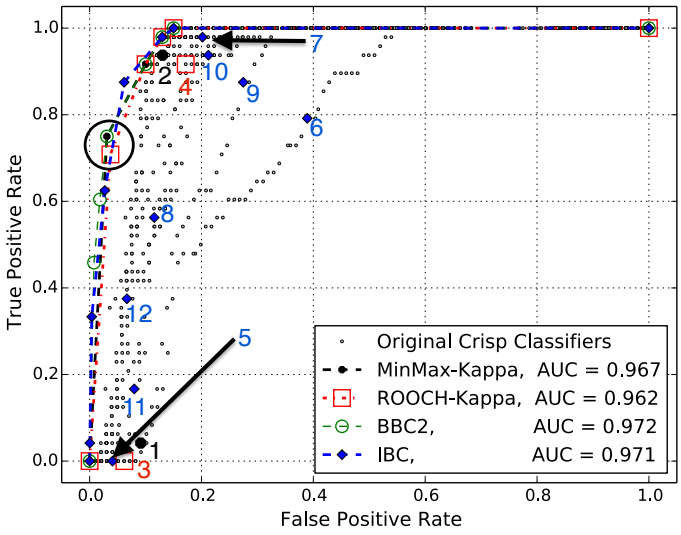


Fig. 4: One of the ROC curves results of 5-folds cross-validation of Boolean combination on one fold and evaluated on four folds. The numbers represent the crisp detectors selected for combination (by each technique) to achieve about the same operational point as denoted by the large black circle

VI. Results

Figure 4 shows the ROC curves and the AUC performance for both pruning techniques proposed in this paper (MinMax- and ROCCH-Kappa) compared to those of IBC and BBC2 (PBC with no pruning mechanism). These results are for one of the 5FCV experiments (the combinations are computed on one fold and evaluated on four) as described in Section V. As shown in the figure the results are comparable both in term of AUC values and the shape of the ROC curves. This is also confirmed in Table II, where the AUC values of each combination technique is averaged over the 5FCV experiments.

Table III shows the average over the 5FCV of the pruning time (for MinMax-Kappa and ROCCH-Kappa), combination time, and number of Boolean operations required by each technique to achieve the final ROCCH (as shown in Figure 4 for one fold). We set the maximum number of selected detectors to $U = 50$ for both MinMax-Kappa and ROCCH-Kappa (this can be further optimized, but gave good trade-off between performance and time complexity). Therefore, the input to

TABLE III: Comparison of pruning and combination time (seconds) and number of Boolean operations required to achieve the final ROCCH during the design phase, and the number of selected detectors during the operational phase. All values are averaged over 5FCV.

	Design Phase			**Operations**
Method Name	Pruning Time	Combination Time	# Boolean operations	# Combined detectors
BBC2	N/A	16364	4,000,000	2
IBC	N/A	11	11,000	11
MinMax-Kappa	1.6	15	19,701	2
ROCCH-Kappa	11.8	38	37,701	2

these pruning techniques is $n = 2000$ crisp detectors, while the output is a subset of a maximum size of 50 detectors provided for Boolean combination (in PBC Algorithm). MinMax-Kappa took about 1.6 seconds in average to select the best subset, while ROCCH-Kappa took about ten times more due the n_{ev} factor, which is described in Section IV-D, to prune the ensemble of $n = 2000$ detectors. Furthermore, MinMax-Kappa is able to select a smaller number of detectors than $U = 50$ on average and computes the Kappa values once, which explains the reduction in combination time and number of Boolean operations compared to those of ROCCH-Kappa, as shown in Table III. Thereby, although both techniques provide similar AUC performance, MinMax-Kappa is slightly preferred due to its improved efficiency compared to ROCCH-Kappa.

Table III also shows that our MinMax-Kappa techniques was able to achieve the same AUC performance of BBC2 (the pairwise Boolean combination of the 2000 detectors), by selecting less than 50 detectors out of the 2000 ones. More interestingly, MinMax-Kappa achieved these results with an average time of three magnitudes lower than that of BBC2, and about 200 times fewer Boolean operations.

Compared to IBC, MinMax-Kappa requires, on average, slightly more combination time and a larger number of Boolean operations to achieve the same AUC performance. However, the number of selected detectors and Boolean functions required to realize each vertex on the ROCCH is on average five times more according to our experiments. For instance, to achieve the final operating points denoted with a large black circle on Figure 4, MinMax-Kappa uses two detectors and one Boolean function according to the following formula:

$$\neg D_1 \oplus D_2$$

Note that in Figure 4, we only shown the number of crisp detectors not to clutter the figures (i.e., 1 means D_1)
Similarly, ROCCH-Kappa uses the following detectors and Boolean function to achieve similar operating point:

$$\neg D_3 \oplus D_4$$

However to achieve similar point of operations, IBC requires eight detectors combined according to the following Boolean operations:

$$(((((((D_5 \oplus D_6) \wedge \neg D_{11}) \wedge D_7) \oplus D_8) \equiv D_9) \vee D_{10}) \wedge D_{12})$$

The average number of combined detectors on the final ROCCH over the 5FCV is about 10 detectors when using IBC compared to two detectors when using PBC with MinMax-Kappa pruning as shown in the last column of Table III. This sequence of combination rules grows linearly with the number of soft detector K, which makes IBC results difficult to analyse and understand for large K values. In contrast to IBC, the combination of two detectors according to combination of PBC with MinMax-Kappa are insensitive to order in which detectors are input to the algorithm, which makes the search for the best subset of detectors easier. However, MinMax-Kappa requires an optimization of the maximum number of detectors U that trades off the complexity and the accuracy. Setting an ADS based on two HMMs into operations, requires less time and memory resources to provide the output probabilities of the input system call sequences. In addition, operating a small number of detectors becomes critical in application of anomaly detection mobile security, due to the constraint on power resources.

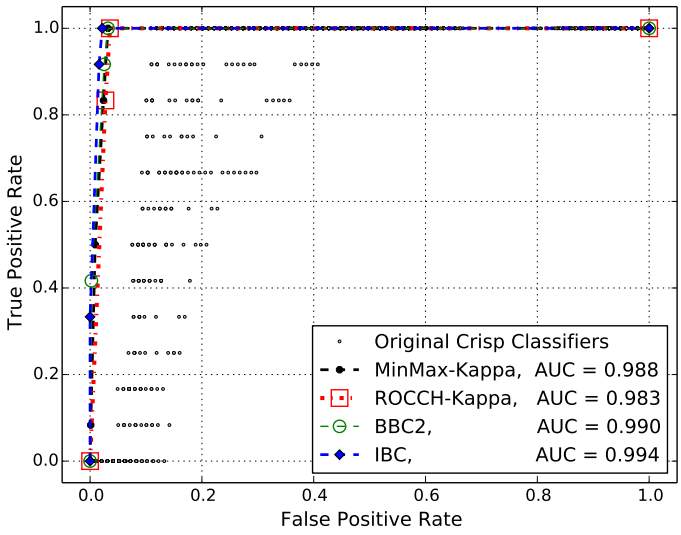


Fig. 5: One of the ROC curves results of 5-folds cross-validation of Boolean combination on four folds and evaluated on one fold.

We conducted an alternative case study to check the impact on ROC and AUC performance of all Boolean techniques, when the number of anomalies is increased during the design phase. Therefore, instead of using one fold (comprising 12 attack traces and 874 normal traces) for selecting the crisp detectors and the corresponding Boolean functions, as described in Section V, we used 4 folds (48 attack traces and 3498 normal traces) and one fold for evaluation of performance.

Figure 5 shows the ROC curves and the AUC performance for all techniques, PBC with MinMax-Kappa and ROCCH-Kappa, IBC and BBC2. Again, the presented results are for one of the 5FCV experiments (but the combinations are computed on four folds and evaluated on one). As shown in the figure, all techniques provide comparable results; however, with a large improvement in detection accuracy over those presented in Figure 4. For instance, for detecting all attacks ($tpr = 100\%$) the false positive rate is now $fpr \approx 2\%$ compared to $fpr \approx 16\%$ in Figure 4. This boost in performance can be also seen in Table IV in terms of average AUC values for each combination technique over the 5FCV experiments. The results of this experiments show, as expected, that when the system is provided with more normal or attack traces, the overall

TABLE IV: Average AUC values and their standard deviations over the 5FCV for each techniques. Design on four folds and evaluated on one fold.

Method Name	Mean	Std
BBC2	0.98177	0.00636
IBC	0.98003	0.01127
MinMax-Kappa	0.97806	0.00640
ROCCH-Kappa	0.97578	0.00585

performance of all Boolean combination techniques improves. In such cases, there is no need to retrain the original detectors (HMMs in our case), which is a time consuming process, but the design phase of Boolean combination techniques must be repeated. This provides an advantage for IBC and PBC, since they are efficient in selecting the detectors for final operations. However, our PBC approach will always provide two detectors for each emerging point on the ROCCH, which is less costly to operate and easier to analyse in real-world setting.

VII. Threats to Validity

A threat to internal validity exists in the implementation of the IBC, BBC2 and PBC algorithms as well as in conducting the experiments for anomaly detection. We have mitigated this threat by manually verifying the outputs.

We have conducted experiments using only one system call dataset derived from the Linux operating system, which consists a threat to external validity of this study. More experiments are therefore required to generalize the presented results to other datasets, operating systems and other software vulnerabilities.

Evasion attacks could also pose a threat to validity. For instance, *mimicry* attacks try to mimic the normal system behavior will go undetected with the ADSs that are based on individual system calls or their temporal order [38]. Mimicry attacks could be conducted by an attacker who is able to launch his attack without tempering the normal order of system calls by, for instance, replacing foreign system call sequences (which can be easily detected) with normal ones or by using system call arguments [38].

The manifestations of such mimicry attacks could be detected by including additional features, such as system call arguments [3], return values extracted from the call stack information [10], and the user identity [28]. An added advantage of multiple detector systems, combined without our PBC, is that they can combine different detectors trained on various features, such as system call arguments, return values and other information flow features to help mitigating such evasion attacks.

VIII. Conclusion

In this paper, we proposed PBC, an efficient approach for selecting and combining anomaly detectors, which relies on two novel pruning techniques. During the design phase, the PBC is able to select a small subset of diverse and accurate detectors for Boolean combinations, while discarding the remaining ones. The pruning techniques we developed at the core of PBC rely on Kappa measure (MinMax-Kappa) and on the ROC convex hull (ROCCH-Kappa) to aggressively prune redundant and trivial detectors.

The results on ADFA-LD system call datasets show that PBC with both pruning techniques are capable of maintaining similar overall accuracy as measured by the ROC curves to that of IBC and BBC2. Therefore, our proposed PBC-based ADS is able to prune and combine large number of detectors without suffering from the exponential explosion in number of combinations provided with the pairwise brute-force Boolean combination techniques. This has been shown analytically (in the time complexity analysis) and confirmed in the experimental results.

During the operational phase, PBC with both pruning techniques always provides two crisp detectors for each combination, while IBC requires an average of 11 detectors to achieve the same operating point (in terms of true and false positive rates). The proposed PBC approach is also general and can be applied to combine any soft or crisp detectors or two-class classifiers in a wide range of applications that requires combination of decisions.

Future work involves conducting more experiments using different real-world datasets. Another interesting direction is to investigate the potential improvement by re-combining the resulting combinations with the selected detectors, and explore other measures of diversity. We also intend to implement the new techniques in TotalADS [35], a tool we have developped to support multiple anomaly detectors. Finally, we need to investigate how we can reduce the size of traces to enable better scalability. Example of trace abstraction techniques are presented in [16], [34].

Acknowledgment

This research is partly supported by a grant from Natural Sciences and Engineering Research Council of Canada (NSERC), Defence Research and Development Canada (DRDC) Valcartier (QC), and Ericsson Canada.

References

[1] M. Barreno, A. Cardenas, and D. Tygar, "Optimal roc for a combination of classifiers," in *Advances in Neural Information Processing Systems (NIPS) 20, 2008*, Jan. 2008.

[2] L. E. Baum, G. S. Petrie, and N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[3] S. Bhatkar, A. Chaturvedi, and R. Sekar, "Dataflow anomaly detection," in *IEEE Symposium on Security and Privacy*, 2006.

[4] Y.-S. Chen and Y.-M. Chen, "Combining incremental hidden Markov model and Adaboost algorithm for anomaly intrusion detection," in *CSI-KDD '09: Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, (New York, NY, USA), pp. 3–9, ACM, 2009.

[5] W. W. Cohen, "Fast effective rule induction," in *Proc. of the 12th International Conference on Machine Learning* (A. Prieditis and S. Russell, eds.), (Tahoe City, CA), pp. 115–123, Morgan Kaufmann, July 1995.

[6] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," in *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, (Shanghai, China), pp. 4487–4492, Apr. 2013.

[7] T. G. Dietterich, "Ensemble methods in machine learning," in *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, (London, UK), pp. 1–15, Springer-Verlag, 2000.

[8] Y. Du, H. Wang, and Y. Pang, "A hidden Markov models-based anomaly intrusion detection method," *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, vol. 5, pp. 4348–4351, 2004.

[9] T. Fawcett, "An introduction to ROC analysis," *Pattern Recogn. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[10] H. Feng, O. Kolesnikov, P. Fogla, W. Lee, and W. Gong, "Anomaly detection using call stack information," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pp. 62–75, 2003.

[11] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," in *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, pp. 120–128, 1996.

[12] S. Forrest, S. Hofmeyr, and A. Somayaji, "The evolution of system-call monitoring," in *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pp. 418–430, Dec. 2008.

[13] B. Gao, H.-Y. Ma, and Y.-H. Yang, "HMMs (Hidden Markov Models) based on anomaly intrusion detection method," *Proceedings of 2002 International Conference on Machine Learning and Cybernetics*, vol. 1, pp. 381–385, 2002.

[14] F. Gao, J. Sun, and Z. Wei, "The prediction role of hidden Markov model in intrusion detection," in *Canadian Conference on Electrical and Computer Engineering*, vol. 2, (Montreal, Canada), pp. 893–896, Institute of Electrical and Electronics Engineers Inc., 2003.

[15] A. K. Ghosh, A. Schwartzbard, and M. Schatz, "Learning program behavior profiles for intrusion detection," in *Proceedings of the Workshop on Intrusion Detection and Network Monitoring*, (Berkeley, CA, USA), pp. 51–62, USENIX Association, 1999.

[16] A. Hamou-Lhadj, "The Concept of Trace Summarization," in *Program Comprehension through Dynamic Analysis (PCODA), 2005, Proceedings of the 1st International Workshop on*, pp. 43–47, 2005.

[17] X. Hoang and J. Hu, "An efficient hidden Markov model training scheme for anomaly intrusion detection of server applications based on system calls," in *IEEE International Conference on Networks, ICON*, vol. 2, (Singapore), pp. 470–474, 2004.

[18] J. Hu, "Host-based anomaly intrusion detection," in *Handbook of Information and Communication Security* (P. Stavroulakis and M. Stamp, eds.), pp. 235–255, Springer Berlin Heidelberg, 2010.

[19] S. Jha, K. Tan, and R. Maxion, "Markov chains, classifiers, and intrusion detection," in *Proceedings of the Computer Security Foundations Workshop*, pp. 206–219, 2001.

[20] W. Khreich, E. Granger, R. Sabourin, and A. Miri, "Combining Hidden Markov Models for anomaly detection," in *International Conference on Communications (ICC)*, (Dresden, Germany), pp. 1–6, June 2009.

[21] W. Khreich, E. Granger, A. Miri, and R. Sabourin, "Boolean combination of classifiers in the ROC space," in *20th International Conference on Pattern Recognition*, (Istanbul, Turkey), pp. 4299–4303, Aug. 23–26 2010.

[22] W. Khreich, E. Granger, A. Miri, and R. Sabourin, "A survey of techniques for incremental learning of HMM parameters," *Information Sciences*, vol. 197, pp. 105–130, Feb. 2012.

[23] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, 1998.

[24] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," in *Proceedings of the 19th Annual Computer Security Applications Conference*, ACSAC '03, (Washington, DC, USA), IEEE Computer Society, 2003.

[25] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Hoboken, NJ: Wiley, 2004.

[26] L. Kuncheva, "That elusive diversity in classifier ensembles," *Pattern Recognition and Image Analysis*, vol. 2652, pp. 1126–1138, 2003.

[27] L. I. Kuncheva, "A bound on kappa-error diagrams for analysis of classifier ensembles," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 3, pp. 494–501, 2013.

[28] U. Larson, D. Nilsson, E. Jonsson, and S. Lindskog, "Using system call information to reveal hidden attack manifestations," in *Security and Communication Networks (IWSCN), 2009 Proceedings of the 1st International Workshop on*, pp. 1–8, May 2009.

[29] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & Security*, vol. 21, no. 5, pp. 439–448, 2002.

[30] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[31] C. Marceau, "Characterizing the behavior of a program using multiple-length n-grams," in *NSPW '00: Proceedings of the 2000 workshop on New security paradigms*, (New York, NY, USA), pp. 101–110, ACM Press, 2000.

[32] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, pp. 211–218, 1997.

[33] S. S.Murtaza, A. Sultana, A. Hamou-Lhadj, M. Couture, "On the Comparison of User Space and Kernel Space Traces in Identification of Software Anomalies," in *Software Maintenance and Reengineering (CSMR), 2012, Proceedings of the 16th European Conference on*, pp. 127–136, 2012.

[34] S. S.Murtaza, W. Khreich, A. Hamou-Lhadj, M. Couture, "A Host-based Anomaly Detection Approach by Representing System Calls as States of Kernel Modules," in *Software Reliability Engineering (ISSRE), 2013, Proceedings of the 24th IEEE International Symposium on*, pp. 431–440, 2013.

[35] S. S. Murtaza, A. Hamou-Lhadj, W. Khreich, M. Couture, "TotalADS: Automated Software Anomaly Detection System," in *Source Code Analysis and Manipulation (SCAM), 2014, Proceedings of the 14th IEEE International Working Conference on*, pp. 83–88, 2014.

[36] L. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[37] A. Sultana, A. Hamou-Lhadj, M. Couture, "An Improved Hidden Markov Model for Anomaly Detection Using Frequent Common Patterns," in *Communications, The Communication and Information Systems Security Symposium, 2012 Proceedings of the IEEE International Conference on*, pp. 1113–1117, 2012.

[38] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, (Washington, DC, United States), pp. 255–264, 2002.

[39] W. Wang, X.-H. Guan, and X.-L. Zhang, "Modeling program behaviors by hidden Markov models for intrusion detection," *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, vol. 5, pp. 2830–2835, 2004.

[40] P. Wang, L. Shi, B. Wang, Y. Wu, and Y. Liu, "Survey on HMM based anomaly intrusion detection using system calls," in *Computer Science and Education (ICCSE), 2010 5th International Conference on*, pp. 102–105, Aug. 2010.

[41] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: alternative data models," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, (Oakland, CA, USA), pp. 133–45, 1999.

[42] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognition*, vol. 36, no. 1, pp. 229–243, 2003.

[43] X. Zhang, P. Fan, and Z. Zhu, "A new anomaly detection method based on hierarchical HMM," in *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pp. 249–252, 2003.

[44] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st ed., 2012.