Anomaly Detection Techniques Based on Kappa-Pruned Ensembles

Md. Shariful Islam, Wael Khreich, and Abdelwahab Hamou-Lhadj, *Member; IEEE*

***Abstract*— Ensemble based Anomaly Detection Systems (ADSs), using Boolean combination, have been shown to reduce the false alarm rate over that of a single detector. However, the existing Boolean combination methods rely on an exponential number of combinations making them impractical, even for a small number of detectors. In this paper, we propose weighted pruning based Boolean combination, an efficient approach for selecting and combining accurate and diverse anomaly detectors. It works in three phases. The first phase selects a subset of the available base diverse soft detectors by pruning all the redundant soft detectors based on a weighted version of Cohen's kappa measure of agreement. The second phase selects a subset of diverse and accurate crisp detectors from the base soft detectors (selected in Phase1) based on the unweighted kappa measure. The selected complementary crisp detectors are then combined in the final phase using Boolean combinations. The results on two large scale datasets show that the proposed weighted pruning approach is able to maintain and even improve the accuracy of existing Boolean combination techniques, while significantly reducing the combination time and the number of detectors selected for combination.**

***Index Terms*—Anomaly Detection Systems, Ensemble Methods, Multiple-Detector Systems, Weighted Kappa, Software Reliability and Security.**

ACRONYMS AND ABBREVIATIONS

5FCV	5-Fold Cross Validation
ADFA-LD	ADFA Linux Dataset
ADS	Anomaly Detection System
AUC	Area Under the Curve
BBC2	Pair-wise Brute-force Boolean Combination
BW	Baum-Welc
CANALI-WD	CANALI Windows Dataset
EM	Expectation-Maximization
FB	Forward-Backward
HIDS	Host-based Intrusion Detection System
HMM	Hidden Markov Model
IBC	Iterative Boolean Combination
NIDS	Network Intrusion Detection System
OC-SVM	One-Class Support Vector Machine
PBC	Pruning Boolean Combination
ROC	Receiver Operating Characteristics
ROCCH	Receiver Operating Characteristics Convex Hull
STIDE	Sequence Time-Delay Embedding
WPBC2	Pair-wise Weighted Pruning Boolean Combination
WPIBC	Weighted Pruning Iterative Boolean Combination

Md Shariful Islam is with the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada (e-mail: mdsha_i@ece.concordia.ca).

Wael Khreich is with the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada (e-mail: wkhreich@ece.concordia.ca).

Abdelwahab Hamou-Lhadj is with the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada (e-mail: abdelw@ece.concordia.ca).

NOTATIONS

avg	average value on 5-Fold Cross Validation results
fpr	false positive rate outputted by a crisp detector
kp-fpr	kappa versus false positive rate plotting diagram
kp-tpr	kappa versus true positive rate plotting diagram
max	maximum value on 5-Fold Cross Validation results
min	minimum value on 5-Fold Cross Validation results
std	standard deviation on 5-Fold Cross Validation
tpr	true positive rate outputted by a crisp detector

I. INTRODUCTION

Intrusion Detection Systems (IDS) are divided into two categories: Network Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS). A NIDS monitors and analyzes network traffic. It is transparent (i.e., it can move in different locations) and independent (i.e., it can work in different network topologies). An HIDS works on a host computer and monitors user activities to detect unauthorized access, illegitimate modification of configuration files, and other unwanted behaviors. IDS can be further classified into two categories: Signature-based (or misuse) IDS and Anomaly Detection Systems (ADS). The former can only detect known attacks [30], whereas the latter, the focus of this paper, is capable of detecting novel attacks by analyzing deviations from the normal behavior of a system.

Anomaly detection methods often vary in their design depending on the application domain [62], but the common practice is to train a model that characterizes the normal behavior of a system. The model is used later as a baseline to detect deviations (anomalies) from normalcy. When a trained model produces scores instead of a decision (i.e., normal or anomalous), it is called a soft detector. When it produces a decision instead of scores, it is called a crisp detector. A soft detector can be converted into one or more crisp detectors by setting different thresholds on the output scores. A single crisp detector, however, is known to generate an excessive number of false alarms, which is one the main reasons that limits deployment of ADS in commercial settings [20].

Training an anomaly detector is a one-class classification problem that depends on the normal (or healthy) data. The detector, which in our case is based on HMM, is trained using the normal traces and expected to represent the normal behavior of the system. It should be noted that we used HMM

because of it has been shown to provide best accuracy [41] compared to other classification techniques. Our approach, however, can be used with any other one-class classification method. The combination of several soft or crisp detectors requires a labeled dataset including both normal and anomalous traces to be able to select the best crisp detectors and the Boolean combination functions to optimize the performance (i.e., minimize the *fpr* and maximize the *tpr*). In practice, the evaluation of an anomaly detection system requires a labeled validation set to select the operating thresholds (e.g., tolerable *fpr*). Our approach takes further advantage of this validation set to select the operating thresholds and Boolean functions that yield the most accurate and concise ensemble of detectors, which will be used during system operation.

In this paper, we propose a weighted pruning of Boolean combinations that selects the best subset of diverse base soft detectors by pruning all the redundant ones. Each diverse base soft detector is then used independently to select the complementary crisp detectors instead of brute-force search like in PBC. The complementary crisp detectors are then combined by leveraging both Pair-wise Brute-force Boolean Combination (BBC2) and Iterative Boolean Combination (IBC) [1] [21], which shows that the pruning approach can be used with any Boolean combination approach.

We leverage both weighted and unweighted Cohen's kappa [46][58] in order to select the best subset of diverse base soft detectors. Weighted Cohen's kappa is a special case of simple kappa (unweighted kappa) that is particularly used when the agreements between two detectors are ordinal instead of nominal. In our case, the scores of a soft detector are ordinal and the decision of a crisp detector based on a given threshold is nominal. Our weighted pruning approach prunes both soft and crisp detectors based on the ordinal agreements and the nominal agreements between two detectors. The selected diverse and accurate crisp detectors are then used for Boolean combination. During combination, we leverage both the pair-wise and iterative Boolean combinations introduced by Barreno et al. [1] and Khreich et al. [21], respectively. The proposed Pair-wise Weighted Pruning Boolean Combination (namely called WPBC2) fuses and combines all possible pairs of crisp detectors generated from the selected diverse base soft detectors. Whereas, the Weighted Pruning Iterative Boolean Combination (namely called WPIBC) fuses and combines the selected diverse base soft detectors sequentially until no significant improvement is possible. Another major contribution of this paper is the evaluation of our approach for detecting anomalies at the system call levels. We compare the performance of WPBC2 and WPIBC to that achieved with the original BBC2 and IBC techniques. In addition, we compare the performance of our approaches to Pruning Boolean Combination (PBC) [45].

In sum, the main contributions of this paper are:

1. We propose an anomaly detection approach that enforces the diversities among the combined soft and crisp detectors using weighted and unweighted Cohen's kappa [46].
2. The approach can be used with both pair-wise and iterative Boolean combination techniques [1][21], and easily adaptable to other Boolean combination methods.
3. We evaluate our approach on two large publicly available system call datasets: ADFA Linux Dataset (ADFA-LD) [6] and CANALI Windows Dataset (CANALI-WD) [47].
4. We show that our approach outperforms BBC2, IBC, and PBC by achieving lower false positive rate, while maintaining and improving the detection accuracy, measured using AUC.

The organization of this paper is as follows. The related anomaly detection approaches are discussed in Section II. HMM based anomaly detection systems using system call sequences are discussed in Section III. The convex hull under the ROC space and related Boolean combination techniques are explained in Section IV. The proposed weighted pruning based Boolean combination techniques are discussed in Section V. Section VI presents the proposed method for pruning the Boolean combination as well as the results of the experiments that are carried out on two benchmarks datasets of system call sequences. The effects of pruning based Boolean combination are discussed in Section VII. The limitations of our approach are discussed in Section VIII. The conclusion and future work are reported in Section IX.

II. Related Research

Anomaly detection is an important component used to enhance system reliability and security in a variety of domains. In their detailed survey, Chandola et al. [62] showed that anomaly detection is used in a wide variety of applications such as fraud detection for credit cards, insurance or health care, intrusion detection for cyber-security, fault detection in safety critical systems, and military surveillance for enemy activities. Anomaly detection techniques have also been shown useful in enforcing the reliability of software and hardware systems. Shariyar et al. [67] presented an approach and a supporting tool to detect program functions that are likely to introduce faults in a software system by examining historical execution traces. Their approach can be used to enhance testing and other software verification methods. In a recent study, Sha et al. [68] proposed an approach based on anomaly detection to ensure the safety of cloud-based IT infrastructures. Bovenzi et al. [69] proposed an approach for revealing anomalies at the operating system level to support online diagnosis activities of complex software systems. Yang et al. [70] proposed an efficient method for detecting abnormal executions of Java programs using sequential pattern mining. Gizopoulos et al. [71] argued that the huge investment in the design and production of multicore processors may be put at risk because of reliability threats, mainly due to the existence of bugs and vulnerabilities, unless these systems are equipped with robust anomaly detection tools. They proposed multicore processor architectures that integrate solutions for online error detection, diagnosis, recovery, and repair during field operation.

Several studies (e.g., [11, 41]) showed that the temporal order of system calls issued by a process to request kernel services is effective in describing normal process behavior. This has led to a considerable amount of research studies that investigated various techniques for detecting anomalies at the system call level (see survey in [12]). Among these, sequence time-delay embedding (STIDE) and Hidden Markov Models (HMMs) are the most commonly used [41]. Ensemble methods have been proposed to improve the overall ADS accuracy by combining the outputs of several accurate and diverse models [7] [23] [25] [44]. In particular, combining the outputs from multiple crisp HMM (Hidden Markov Model) detectors generated from multiple soft HMM detectors, each trained with a different number of states, in the Receiver Operating Characteristics (ROC) space, has been shown to provide a significant improvement in the detection accuracy of system call anomalies [11][21][41].

Among existing combination approaches, the Pair-wise Brute-force Boolean Combination (BBC2) combines all possible pairs of crisp detectors by fusing all types of Boolean functions [1]. BBC2 reported a significant improvement in reducing false alarms as compared to a single learned model [1]. However, 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]. To address this, Khreich et al. [21] proposed an Iterative Boolean Combination (IBC) approach for combining relatively a large number of soft HMM detectors while avoiding the exponential explosion of BBC2. However, IBC produces a sequence of combination rules that grows linearly with both the number of soft HMM detectors and the number of iterations, which is difficult to analyze and understand. Furthermore, the algorithm is sensitive to the order of the combined crisp HMM detectors, making it challenging to find the best subset for combination operations.

It is clear that if the number of combined crisp detectors (K) increases, the computation time and complexity also increase linearly for IBC and exponentially for BBC2. Moreover, we also know that the performance of an ensemble method is highly dependent on the diversity of combined detectors [27] [53]. The question is: how can we select the smallest and most diverse subset of detectors (among all the available ones) that can maintain or improve the detection accuracy (while reducing the false alarm rate) using the smallest number of Boolean combinations?

In previous work [45], we proposed an effective Pruning Boolean Combination (PBC) method based on Cohen's kappa [5] coefficient (a statistical measure of the degree of agreement between two classifiers). MinMax-Kappa, a pruning technique of PBC, selects a small subset of diverse and accurate crisp HMM detectors based on measuring the kappa coefficients between each crisp HMM detector's decision and the true decision labels (or ground truth), provided by the validation set (comprising both normal and attack traces). MinMax-Kappa computes the kappa values for all possible crisp HMM detectors, and then sets Min (Minimum kappa value) and Max (Maximum kappa value) boundaries with sorting them in ascending order. After that, MinMax-Kappa selects 50% crisp HMM detectors whose kappa values are close to Min and another 50% crisp HMM detectors whose kappa values are close to Max. However, PBC uses the kappa coefficients between two crisp HMM detectors, it cannot ensure the diversity among soft HMM detectors. For example, if the scores of a subset of available soft HMM detectors on a validation set are almost the same, the responses of the crisp HMM detectors at a decision threshold of these redundant soft HMM detectors will probably be the same. Particularly, the computed kappa values for each crisp HMM detectors generated from these redundant soft HMM detectors will probably be almost equal. So, if the kappa value of one of these redundant crisp HMM detectors is close to Min or Max, the chances of selecting the rest of the redundant crisp HMM detectors are very high. Therefore, only one soft HMM detector from this subset of redundant soft HMM detectors should be used while the rest of the redundant soft HMM detectors should be pruned before converting them into crisp HMM detectors.

III. Hidden Markov Models for Anomaly Detection Using System Call Sequences

The sequences of system calls collected from the system call traces are known to provide a stable signature of normal behavior of a process [11] [41] [47]. There are two properties of system call sequences that make them potential features for anomaly detection. The first property is uniqueness where different processes generate different patterns of system call sequences. The second one is the matching probability, tends to be low when an intruder is attempting to alter the normal sequential pattern of system calls of a process. Researchers from diverse disciplines use these two important properties of system call sequence, which have been proposed in a large number of anomaly detection techniques such as neural network [15], k-nearest neighbors [29], Markov models [19] [31], and Bayesian models [24].

To our knowledge, the very first approach for anomaly detection is based on sequence matching [11] [41]. During training, this approach builds the normal profile by segmenting the full-length sequences of system calls into a fixed-length contiguous sub-sequences using a fixed-size sliding window, shifted one by one symbol. In testing, an unknown sequence of system calls is also segmented into sub-sequences (as in training) and classified as normal if all sub-sequences are present in the normal profile. Otherwise, it is classified as an attack.

HMM has been shown to be a very effective method to model a system's behavior over time [36]. An HMM is a stochastic model for sequential data determined by the two interrelated mechanisms –a latent Markov chain having a finite number of states and a set of observation probability distributions, each one associated with a state. An HMM is typically determined by three parameters $\lambda = (A, B, \pi)$, which represent the states and transition probability distribution (A) of a system in a Markov process, the observation probability distribution (B) of observation sequences that come from the

temporal order of executions of a system, and the initial state probability distribution (π) of each hidden state in a Markov process. The first parameter, A, is usually hidden in an HMM. The only physical events are the observation sequence (B) that is associated with the hidden states of a Markov process. Fig. 1 illustrates a generic topology of an HMM, $\lambda = (A, B, \pi)$ [48].

Number of Hidden States (N): To learn an HMM, we have to set the number of hidden states (N) in a Markov process. Let the distinct states be $S_i,\ i = \{0,1,\dots,N-1\}$. The notation $X_t = S_i$ represents the hidden state sequence at time t.

Number of Observation Symbols (M): To learn an HMM, we have to set the number of observation symbols (*M*). Let the distinct observation symbols be $R_k,\ k = \{0,1,\dots,M-1\}$. The notation $O_t = R_k$ represents the observed symbol R_k at time *t* for the given observation sequence $\mathcal{O} - (\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_{T-1})$, where T is the length of the observation sequence.

State Transition Distribution (A): The first row stochastic process is the hidden state transition probability distribution matrix $A = \{a_{ij}\}$. A is an $N \times N$ square matrix and the probability of each element $\{a_{ij}\}$ is denoted in Equation (1) as:

$$a_{ij} = P\big(state\ S_j\ at\ t+1 \big| state\ S_i\ at\ t\big), \quad i,j = \{0,1,\dots,N-1\} \tag{1}$$

The transition from one state to the next is a Markov process of order one [36]. This means the next state depends only on the current state and its probability value. As the original states are "hidden" in HMM, we cannot directly compute the probability values in the past. But we are able to observe the observation symbols for the current state S_i at time t from a given observation sequence $\mathcal{O}$ to learn an HMM model.

Observation Symbol Distribution (B): The second row stochastic process is the observation symbol probability distribution matrix $B = \{b_j(R_k)\}$. B is an $N \times M$ dimensional matrix that is computed based on the observation sequences (i.e., the temporal order of executions of a system). The probability of each element $b_j(R_k)$ is denoted in Equation (2) as:

$$b_j(R_k) = P\big(observation\ symbol\ R_k\ at\ t \big| state\ S_j\ at\ t\big) \tag{2}$$

Initial State Distribution (π): The third row stochastic process is the initial state probability distribution $\pi = \{\pi_i\}$. π is a $1 \times N$ row matrix and the probability of each element $\{\pi_j\}$ is denoted in Equation (3) as:

$$\pi_i = P(state\ S_i\ at\ t = 0) \tag{3}$$

A. Training an Ergodic HMM

The behavior of a system can be discrete (e.g., symbols from a finite alphabet) or continuous (e.g., signals from a speech, music, etc.). In our case, the behavior of a process in UNIX or Windows system can be represented as a discrete sequence of system calls. Since a discrete HMM is a stochastic process for sequential data [36] [48], we can use it to learn the behavior of a process. A well-trained HMM model using the discrete normal sequences of system calls can be used as a potential model for detecting anomalies. Practically, training an HMM using a discrete sequence of observation $\mathcal{O}$-$(\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_{T-1})$ aims at maximizing the likelihood function $P(\mathcal{O}|\ \lambda)$ over the parameter space represented by A, B, and π. The Baum-Welch (BW) algorithm is one of the most commonly used Expectation-Maximization (EM) algorithm for learning the HMM parameters [2]. The BW algorithm is an iterative procedure to estimate the HMM parameters. It uses a Forward-Backward (FB) algorithm [48] at each iteration to efficiently evaluate the likelihood function $P(\mathcal{O}|\ \lambda)$, and then updates the model parameters until the likelihood function stops improving or a maximum number of iterations is reached. In our experiments, we have chosen the BW algorithm to train all HMMs using the system calls datasets.

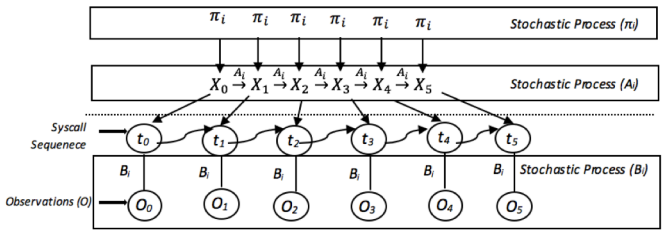


Fig. 1. A general topology for an HMM model.

The user-defined three initial distributions of A, B, and π, and two fixed-value parameters of M and N have an impact on the performance of HMM. The common solution for the initial distributions of A, B and π is the random initialization and the use of validation set to select the best parameters [49]. We have also initialized the distributions of A, B and π randomly and repeated the training process ten times. The initial distributions for which we obtain the highest AUC on the validation set are selected. The alphabet size M is defined by the number of distinct system calls in a system. However, it is difficult to define the number of states N in advance. The reason for that is a single HMM trained with a predefined number of states N may have limited chances to fit the underlying structure of the data [36]. In fact, the underlying distribution of sequences of system calls at different states varies according to the architectural complexity of a system and results in many local maxima of the log-likelihood function [20].

To tackle the variations in the underlying distribution of the sequences of system calls, ensemble HMMs have shown to be a better choice than a single HMM [7] [23]. The ensemble methods have been reported that the diversity among the ensemble classifiers is an essential factor in increasing the accuracy. In particular, Khreich et al., [21] showed that the Iterative Boolean Combination (IBC) of the responses of several accurate and diverse HMM detectors significantly increases the accuracy while reducing the number of false alarms. We have also trained different discrete-time ergotic HMMs with various N using the BW algorithm. These ergodic HMMs are the primary inputs to the proposed weighted

pruning approach for Boolean combination.

B. Soft and Crisp HMM Detectors

In a binary classification problem, any trained model that produces a score instead of a decision (i.e., positive or negative) is called a soft detector. During operation, a trained HMM (λ) outputs a score computed by the FB algorithm. The score is the likelihood or the probability $P(\mathcal{O}_{1:T}|\,\lambda)$ for a given new observation sequence $\mathcal{O}_{1:T}$. Normally, the score provided by λ should be significantly high, if the new observation sequence $\mathcal{O}_{1:T}$ is normal; otherwise, it is considered as an anomaly if the score is comparatively low. Since the output of a trained HMM is a score instead of a decision (normal or anomalous), then this model λ is a soft detector.

On the other hand, in a binary classification problem, any trained model that produces a decision (i.e., positive or negative) instead of a score is called a crisp detector. We can convert a soft detector to one or more crisp detectors by setting one or more thresholds θ on the output scores produced by a soft detector. A crisp detector always gives a decision whether the testing sequence is normal ($score \geq \theta$) or anomalous ($score < \theta$) based on a predefined threshold, θ.

However, because of the limited amount of representative data, complex behavior of a system, imbalanced distributions of classes, it is difficult to determine a threshold on the scores that will always separate the normal and anomalous sequences during operation [21]. Therefore, a single crisp HMM detector may generate a large number of false alarms. The Boolean combination of responses of multiple crisp HMM detectors (crisp-HMMs) in the ROC space have been shown to decrease the false alarm rate [21]. The crisp-HMMs are produced by setting various thresholds on the scores of the multiple soft HMM detectors (soft-HMMs). The following section introduces the two most useful Boolean combination techniques BBC2 and IBC for combining crisp-HMMs and also reports on their limitations.

IV. ROC-Based Boolean Combination Techniques

The ROC curve is a commonly used metric for evaluation of detectors' performance. It plots the performances of a binary classifier in a 2-D space [9], where, y-axis represents the true positive rate (*tpr*) and x-axis represents the false positive rate (*fpr*) for every possible crisp detector. The *tpr* is the proportion of correctly classified positive responses over the total number of positive samples tested by a crisp detector. The *fpr* is the proportion of incorrectly classified negative responses over the total number of negative samples tested by a crisp detector. Therefore, a single crisp detector plots a single point (*fpr*, *tpr*) in a ROC space, while a soft detector produces a ROC curve by connecting all the possible crisp detector's points at various decision thresholds.

A. The ROC Convex Hull (ROCCH)

All the points in a ROC space can be classified into two groups superior and inferior based on their *tpr* and *fpr*. Suppose *a* and *b* are two operating points in the ROC space, *a* is defined as superior to *b*, if $fpr_a \leq fpr_b$ and $tpr_a \geq tpr_b$. If a ROC curve has $tpr(*) > fpr(*)$ for all its points $(*)$, then it is a proper ROC curve. The ROC convex hull (ROCCH) is therefore the piece-wise outer envelope connecting only its superior points [6] [9] [52]. The linear interpolation is used to connect the two adjacent superior points so that, no points in a ROC space lies out of the final ROCCH curve. The accuracy of a ROCCH curve is measured by the Area Under the Curve (AUC).

The ROCCH can be used for the combination of two or more crisp classifiers in a ROC space [50] [51]. However, ROCCH combination rules discard the inferior points without verifying their combination in order to improve the system performance. The following sub-section B introduces the Boolean combination approaches [1] [21] [54] of multiple ROC curves and showed that the new composite ROCCH improves the AUC as compared to the original ROCCH.

B. The Boolean Combination of ROC Curves

The very first Boolean combination approach, proposed by Daugman [54], used only the conjunction (AND) and disjunction (OR) rules and fused on all the responses in a ROC space. The author applied these rules in a biometric test and concluded that the new composite ROCCH may increase the AUC of the ROC curve. As a consequence, other researchers also applied the AND or OR combination to combine soft detectors [55] [56].

For example, consider a pair of soft detectors (S_a, S_b) and the various decision thresholds are T_a and T_b, respectively. In a pair-wise combination, the AND or OR rules are fused between each pair of converted crisp detectors $\left(C_i^a, C_j^b\right)$. The optimum thresholds are then selected based on the Neyman-Person test[1] [52]. Finally, the selected optimum thresholds along with the corresponding Boolean functions are stored and used during operation.

However, the AND and OR combinations cannot provide optimal thresholds when the training and validation datasets are limited and imbalanced [21]. The reason for the limited and imbalance data may lead to the appearance of large concavities in the resulting ROC curves [57]. In particular, the false alarm may be increased, if we fuse the best detector and the worst detector. But, the diversity among the combined detectors is an important factor in order to improve the performance while reducing the false alarm rate [53].

Therefore, further improvement is possible by including the other Boolean rules, in addition to the AND and OR rules. The following sub-sections introduce the two most common combination techniques using all Boolean rules: Pair-wise Brute-force Boolean Combination (BBC2) [1] and Iterative Boolean Combination (IBC) [21]. We also report on the limitations and complexities of these techniques.

C. Pair-wise Brute-force Boolean Combination (BBC2)

The Pair-wise Brute-force Boolean Combination (BBC2) fuses all possible pairs of crisp detectors generated from all the available soft detectors using all Boolean functions. As

[1] The point (tpr_{opt}, fpr_{opt}) of a crisp detector in a ROC space, is optimum, if all the other points for the same value of fpr_{opt}, the value of tpr_{opt} is maximum.

BBC2 uses all Boolean functions, it implicitly combines responses of both accurate and diverse crisp detectors at both superior and inferior points in the ROC space. However, the pair-wise brute-force strategy is computationally expensive due to the high number of permutations. For example, if the number of crisp detectors is N, there are N^2 possible combinations for only one Boolean function. Barreno et al. [1] reported that exploiting all Boolean functions using an exhaustive brute-force search to determine optimum points leads to an exponential number of combinations.

D. The Iterative Boolean Combination (IBC)

IBC avoids the impractical exponential explosion associated with the BBC2 by combining the emerging responses on a composite ROCCH sequentially. It first combines the first two ROC curves of the first two soft detectors. Then, the combined ROCCH, particularly, the emerging points are combined with the next ROC curve, and so on until the K^{th} ROC curve is combined. IBC repeats these sequential combinations iteratively until there are no further improvements or it reaches to a predefined maximum number of iterations. However, in practice, IBC requires a sequence of combinations of 11 to 20 crisp detectors to reach a final point on the final composite ROCCH [45]. In fact, it grows linearly with the increase of the number of iterations. Tracking and analyzing such a long sequence of combination rules during testing time increase the complexity of IBC. Moreover, the order of combined crisp detectors makes the IBC algorithm more sensitive to finding the best subset.

It is evident that the computation time and complexity increase exponentially for BBC2 and linearly for IBC with the increase of the number of combined soft detectors (K), and thus making them inefficient. Our proposed pruning approach select the smallest and most diverse subset of detectors (among all available ones), which does not only reduce the computation time and complexity for Boolean combinations but also maintains or improves the detection accuracy (while reducing the false alarm rate) using the smallest number of Boolean combinations.

V. Proposed Weighted Pruning Technique

The proposed weighted pruning based Boolean combination approach leverages both weighted and unweighted kappa measures of (dis)agreement. The main novelty of this work is to ensure that the diversity among the scores of all the available ensemble of soft detectors by pruning the redundant soft detectors using weighted kappa. Then, our approach applies the unweighted kappa based MinMax-Kappa pruning technique (one of the pruning techniques of PBC) individually on each selected diverse base soft detectors and selects the complementary crisp detectors. At the end, we merge all the selected complementary crisp detectors from each selected diverse base soft detectors and use them for Boolean combination.

A. Kappa Measure of (Dis)Agreement

Cohen's kappa or simply called kappa is a statistical tool that is widely used for measuring the inter-rater reliability or (dis)agreement between raters [5]. There are two types of kappa coefficients that can be used in computing the inter-rater reliability. The unweighted kappa coefficient is the simplest version of kappa [58] that is used only for nominal category. The weighted kappa coefficient is an extended version of kappa [46] that is used when the category is ordinal [59]. Our pruning techniques leverage both kappa coefficients. The weighted kappa coefficient is used to prune the redundant soft detectors when the level of scores is ordinal (thresholds). And the unweighted kappa coefficient is used to prune the trivial and redundant crisp detectors when the decision is nominal (anomaly/normal).

TABLE I
CONTINGENCY MATRIX

		D1	
		Positive/$level_1$	Negative/$level_2$
D2	Positive/$level_1$	a_{11}	a_{12}
	Negative/$level_2$	a_{21}	a_{22}

The contingency matrix for both kappa coefficients of (dis)agreement is defined on two detectors. Let the two detectors be *D1* and *D2* and the contingency matrix is $C_{n\times n}$. Here, n is the order of levels. For unweighted kappa coefficient, n is fixed to two that is either positive or negative. For weighted kappa coefficient, n is equal to the number of levels or thresholds with the assumption that both detectors have the same number of constant levels or thresholds. An example of a contingency table $C_{2\times 2}$ for $n=2$, is given in Table I. Where, each element a_{ij} represents the number of instances on which detector *D1* and detector *D2* agree at $level_i$ and $level_j$. The sum of all elements in Table I is equal to the size of the validation set.

For the weighted kappa coefficient, we need to define the weighted matrix W in addition to the contingency matrix C. Among the many possible weighting schemes, the linear weighting scheme is effective when one order is important than the next one [60]. We also use linear weight when the order is the number of thresholds and the distance between two thresholds is important to define whether two soft detectors are similar or diverse. We can compute the linear weighting matrix W using (4).

$$W = w_{ij} = 1 - \frac{abs(i-j)}{n-1} \tag{4}$$

When C and W are the same dimensional square matrices, the kappa coefficient for both unweighted and weighted kappa can be computed based on the Hadamard product (o) [46] or element-wise product of matrices according to (5):

$$kp = \frac{p_a - p_\varepsilon}{1 - p_\varepsilon} \tag{5}$$

where $p_a = sum(CoW)$ is the proportion of weighted agreement (for unweighted kappa, $W = I$ means complete agreement). The parameter p_ε is the proportion of agreement due to chance and computed using (6) as:

(a)

$S(D_1)$-	3.9|4.3|3.4|3.2|3.6|3.1|3.0|3.0|2.3|2.6|2.5|2.2|0.8|1.4|0.8|1.9|0.8|1.1|1.3|0.8|0.8|0.8|1.1|10.5|0.8
$S(D_2)$-	3.2|3.4|3.1|3.0|3.7|3.0|3.0|3.0|2.3|2.5|2.5|2.2|0.9|1.2|0.9|1.8|0.9|1.0|1.2|0.9|0.9|0.9|1.3|10.8|0.9
$S(D_3)$-	2.9|2.8|3.2|3.0|3.1|2.9|2.8|2.9|2.2|2.5|2.4|2.3|1.9|2.7|1.9|1.9|1.9|2.4|2.0|1.9|1.9|1.9|2.4|11.2|1.9

(b)

Testing soft detectors D_2 and D_3	Base soft detector (D_1)			
Levels	$S \geq 3$	$3 > S \geq 2$	$2 > S \geq 1$	$1 > S \geq 0$
$S \geq 3$	9_4_1.00	0_0_0.66	0_0_0.33	0_0_0.00
$3 > S \geq 2$	0_5_0.66	4_4_1.00	0_4_0.66	0_0_0.33
$2 > S \geq 1$	0_0_0.33	0_0_0.66	5_1_1.00	0_7_0.66
$1 > S \geq 0$	0_0_0.00	0_0_0.33	0_0_0.66	7_0_1.00

(c)

Ground Truth-11111000000000000000000	Ground Truth-11111000000000000000000
D_1: R($S \geq 3$)- 11111111000000000000010 D_2: R($S \geq 3$)- 11111111000000000000010 D_3: R($S \geq 3$)- 00111000000000000000010	D_1: R($S \geq 2$)- 11111111111000000000010 D_2: R($S \geq 2$)- 11111111111000000000010 D_3: R($S \geq 2$)- 11111111111010001100011 0
D_1: R($S \geq 1$)- 11111111111101010110001 10 D_2: R($S \geq 1$)- 11111111111101010110001 10 D_3: R($S \geq 1$)- 11111111111111111111111	D_1: R($S \geq 0$)- 11111111111111111111111 D_2: R($S \geq 0$)- 11111111111111111111111 D_3: R($S \geq 0$)- 11111111111111111111111

(d)

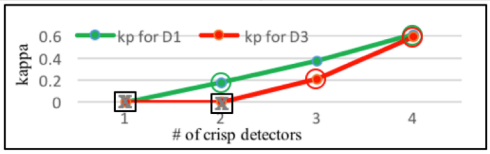


Fig. 2. A simple example of weighted and unweighted kappa for pruning redundant soft and crisp detectors

$$p_\varepsilon = (c_{n\times 1} \times r_{1\times n}) oW \tag{6}$$

Here, $c_{n\times 1}$ denotes a column matrix in which each element is the sum of each row of C. Similarly, $r_{1\times n}$ is a row matrix in which each element is the sum of each column of C. The kappa coefficient kp computes the inter-rater reliability based on the proportion of agreement (p_a) and agreement due to chance (p_ε), where the degrees of disagreement are controlled by the weight matrix W ($W = I$ for unweighted kappa that means no degrees of disagreement). Therefore, $kp = 1$ indicates perfect agreement (i.e., both detectors agree at the same level for every instances) and $kp = 0$ indicates that any agreement is totally due to chance. The value of kp might also be negative. Negative values indicate both detectors are negatively correlated, and such complementary detectors are important in the combination of ensemble techniques [27] [53].

In the rest of this paper, we use the running example shown in Fig. 2 to describe the phases of our approach. In this example, we have selected three HMM-based detectors, D_1, D_2, and D_3 by varying the number of hidden states. Fig. 2 (a) shows the scores of each detector.

Phase1-Pruning Using Weighted Kappa: The first phase of Algorithm 1 describes the steps for pruning the redundant soft detectors using weighted kappa coefficient kp. Suppose, we have K soft detectors and they produce $S_k\{k = 1 \ldots K\}$ score vectors using a validation set V. In the example of Fig. 2, K = 3 and the scores for each detector are shown in Fig. 2 (a). Let the number of thresholds of each soft detector be n_k. In the example of Fig. 2, $n_k = 4$. Therefore, we have K ROC curves (S_k, n_k) with probably K different AUC values. In each iteration (lines 7-18 in Algorithm 1), we select one out of K available soft detectors for which the AUC is maximum and

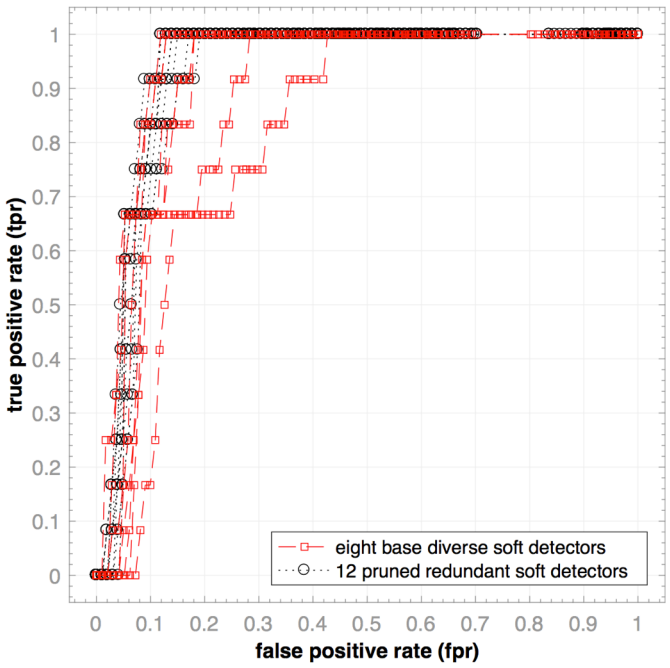
Algorithm 1: $PSCDs(S_1, \ldots S_K, T_1, \ldots T_K, lab)$: Pruning Soft and Crisp Detectors

input: scores of K soft detectors $\{S_1, \ldots S_K\}$ on a validation set along with their thresholds $\{T_1, \ldots T_K\}$, and true labels lab of size $|lab|$.
output: selected $L \ll K$ diverse base soft detectors $\{B_1, \ldots B_L\}$ along with their complementary crisp detectors or thresholds $\{\theta_1, \ldots \theta_L\}$ where $\theta_l \ll T_l$ ($\theta_l = 12$ and $T_l = 100$ on average)

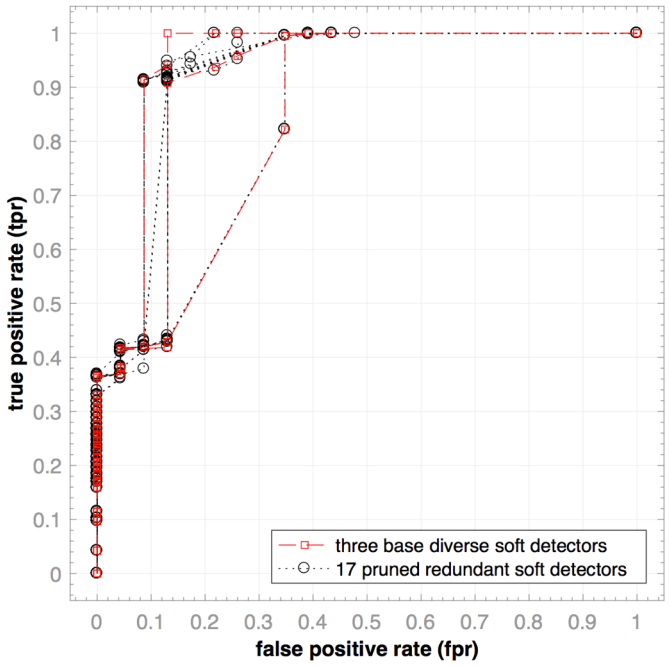
1 // *Phase1*-pruning soft detectors using weighted kappa
2 **allocate** an array $AUC_{all}[1{:}K]$ // temporary store *auc* of each S_k
3 **for** $k \leftarrow 1$ **to** K **do**
4 **compute** *auc* of $ROC(S_k, T_k)$
5 **push** *auc* onto AUC_{all}
6 **allocate** an empty array $B = []$ //store selected diverse soft detectors
7 **while** (K)
8 **select** base soft detector: $S_b \leftarrow max_k[AUC_{all}(k)]$
9 **store** S_b onto B // store S_b as a base soft detector
10 **let** $n_b \leftarrow$ number of order/levels/thresholds in T_b
11 **update** $K \leftarrow K - S_b$ // remove S_b from K soft detectors
12 **update** $AUC_{all} \leftarrow AUC_{all} - AUC_{all}(S_b)$ // remove *auc* for S_b
13 **let** $n \leftarrow$ the size of $|K|$
14 **for** $k \leftarrow 1$ **to** n **do**
15 **compute** linear weighted kappa kp between S_k and S_b using n_b
16 **if** $0.80 < kp <= 1$
17 **update** $K \leftarrow K - S_k$ // remove S_k as a redundant copy of S_b
18 **update** $AUC_{all} \leftarrow AUC_{all} - AUC_{all}(S_k)$ // remove *auc* for S_k
19 // -----*Phase2*- pruning crisp detectors using unweighted kappa----------
20 **let** $L \leftarrow$ number of selected diverse base soft detectors in B
21 **let** $m \leftarrow$ number of selected complementary crisp detectors from $S_b \in B$
22 **allocate** an empty array $\theta = []$ //store thresholds of each complementary crisp //detectors
23 **for** $b \leftarrow 1$ **to** L **do**
24 **let** $n_b \leftarrow$ number of crisp detectors or thresholds in $T_b \in S_b$
25 **allocate** an array $U[1{:}n_b]$ // store temporary kappa coefficients
26 **allocate** an array $V[|lab|{:}n_b]$ //store temporary responses
27 **for** $j \leftarrow 1$ **to** n_b **do**
28 $r \leftarrow S_b \geq t_j$ //temporary responses at decision threshold $t_j \in T_b$
29 **compute** unweighted kappa kp between r and lab
30 **push** kp onto U and r onto V
31 **filter** U and V by removing trivial detectors
32 **select** m complementary crisp detectors using $MinMaxKappa(U, V)$ pruning technique
33 **map** m selected complementary crisp detectors into θ_b thresholds
34 **store** θ_b thresholds onto θ // store θ_b complementary crisp detectors of S_b
35 **return** $B < S_1, \ldots S_L >$ and $\theta < \theta_1, \ldots, \theta_L >$

use it as a base soft detector S_b. We store S_b onto B (line 9 in Algorithm 1) for the next Phase2. Now, we compute the weighted kappa coefficients kp between S_b and each of the rest $K \leftarrow K - S_b$ soft detectors where the thresholds n_k of S_b are used as an order or levels. Then, the soft detectors among the $K - S_b$ soft detectors which perfectly agree ($0.8 < kp \leq 1$) with S_b based on the computed weighted kappa kp, are pruned as a redundant copy of S_b. Let say, the number of redundant detectors we found in each iteration is $0 \leq K' \leq K - 1$, and then we remove them from the available K detectors as: $K \leftarrow K - K'$. We repeat this process until K is zero.

Using the example shown in Fig. 2, we have $S_b = D_1$ because the AUC of D_1 is maximum. We then store D_1 in B as a base soft detector. Suppose, n_k of $D1$ is equal to four different levels ($S \geq 3;\ 3 > S \geq 2; 2 > S \geq 1;\ and\ 1 > S \geq 0$) of scores $S(D_1)$. First, we have to compute the contingency and weighted matrices between base ($S_b = D_1$) and each of the rest two ($K \leftarrow K - S_b$) soft detectors D_2 and D_3. Fig. 2(b) shows the contingency tables ($C_{4\times 4}$) for four different levels.



(a) Diverse and redundant soft detectors on ADFA-LD dataset

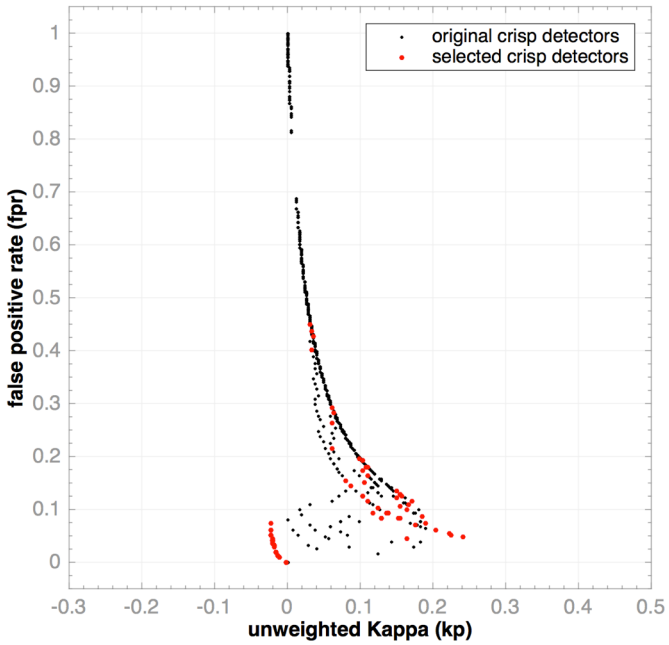


(b) Diverse and redundant soft detectors on CANALI-WD dataset

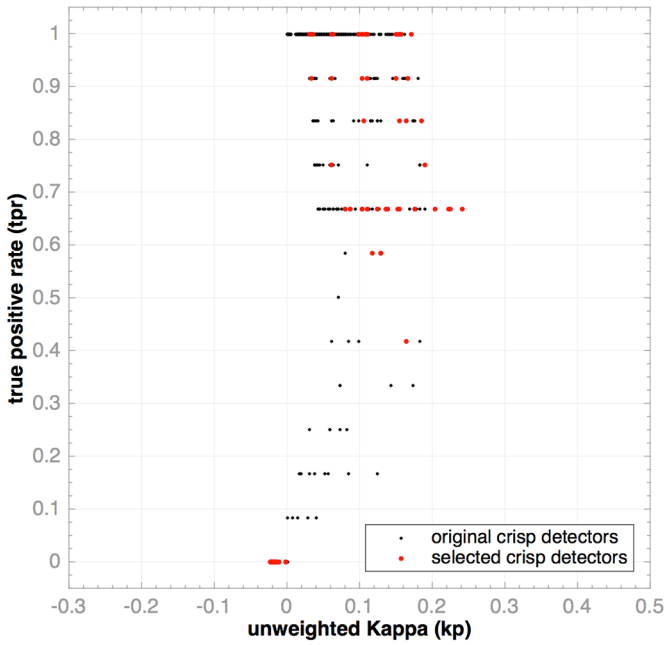
Fig. 3. Example of selected base soft detectors (green solid lines) with pruning redundant soft detectors (doted black lines) under the ROC space using weighted kappa (*Phase1* in Algorithm 1) on ADFA-LD dataset (a) and CANALI-WD dataset (b).

Since the dimension of the contingency and weighted matrices are the same, we put them together, where, each cell $c_{ij}(\#\_\#\_w_{ij})$ in Fig. 2(b) represents three values: The first and second values represent the number of samples agreed at levels i and j of the two contingency tables between D_1 and D_2 and between D_1 and D_3, respectively. The third value is the linear weight, computed using Equation (4).

Based on the contingency and weighted matrices between



(a) *kp-fpr* diagram



(b) *kp-tpr* diagram

Fig. 4. Example of selected complementary crisp detectors (red bold points) under the simple kappa versus true positive rate (kp-tpr) diagram (a) and kappa versus false positive rate (kp-fpr) diagram (b) with pruning trivial and redundant crisp detectors (small black points) from the L base soft detectors (selected by *Phase1* in Algorithm 1) using *MinMax-Kappa* pruning technique (*Phase2* in Algorithm 1) on ADFA-LD dataset.

two detectors, we can compute the weighted kappa (kp) coefficients using Equation (5). The weighted kappa kp between D_1 and D_2 is 1, meaning that both are in perfect agreement (i.e., $kp \in 0.8 < kp \leq 1$) at the same level for every instances, and thus D_2 should be pruned (lines 15 to 18 in Algorithm 1). However, the weighted kappa kp between D_1 and D_3 is 45.65, meaning poor agreement (i.e., $kp \notin$

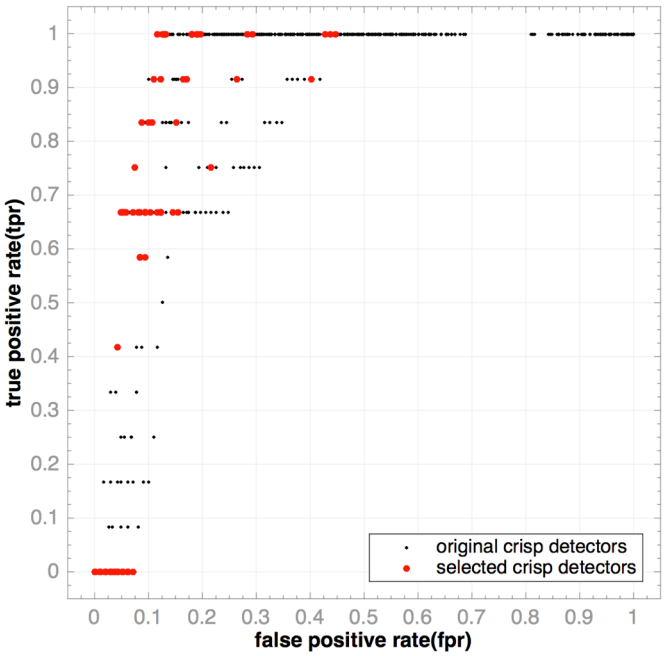


Fig. 5. Example of selected complementary crisp detectors (red bold points) under the ROC space with pruning trivial and redundant crisp detectors (small black points) from the L base soft detectors (selected by *Phase1* in Algorithm 1) using *MinMax-Kappa* pruning technique (*Phase2* in Algorithm 1) on ADFA-LD dataset

$0.8 < kp \leq 1$) at the same level for every instances, and therefore D_3 is more likely to diverse from D_1 and should be selected for combination. At the end of the first iteration, we only keep D_3 (i.e., K=1), while D_2 is pruned because it is redundant of the base detector, D_1. The final results of this phase consist of two diverse base soft detectors D_1 and D_3. The diversities at the response level for four different thresholds are presented in Fig. 2(c). We can see that the responses of the two selected base soft detectors, D_1 and D_3, diverse at various instances (see Fig. 2(c)) for all threshold points, except for $S \geq 0$.

In Fig. 3 (a), we show a more realistic example, using the ADFA-LD dataset with 20 soft HMM detectors. In this figure, we have eight base soft diverse detectors (green solid ROC

Algorithm 2: $WPBC2(S_1, \dots S_K, T_1, \dots T_K, lab)$: Weighted Pruning Pair-wise Boolean Combination

input: scores of K soft detectors $\{S_1, \dots S_K\}$ on a validation set along with their thresholds $\{T_1, \dots T_K\}$, and true labels lab of size $|lab|$.
output: a new composite $ROCCH$—consturcted by $|P_e|$ (size of P_e) combination responses or $|P_e|$ emerging points. Each point is a combination of two crisp detectors using only one Boolean function.

prune redundant soft and crisp detectors
$(B < S_1, \dots S_L >, \theta < \theta_1, \dots, \theta_L >) \leftarrow PSCDs(S_1, \dots S_K, T_1, \dots T_K, lab)$
// where $L \ll K$ is the number of selected diverse base soft detectors
set *BooleanFunctions* $\leftarrow \{a \wedge b, \neg a \wedge b, a \wedge \neg b, \neg(a \wedge b), a \vee b, \neg a \vee b, a \vee \neg b, \neg(a \vee b), a \oplus b, a \equiv b\}$
let $F \leftarrow$ number of Boolean functions in *BooleanFunctions*
let $m_i \leftarrow$ number of decision thresholds in θ_i
let $M \leftarrow \sum_{i=1}^{L} m_i$ total number of crisp detectors
allocate an array $C[|lab|, M]$
// convert soft detectors to crisp detectors
for $i \leftarrow 1$ **to** L **do**
  for $j \leftarrow 1$ **to** m_i **do**
    $r \leftarrow S_i \geq t_j$ //temporary responses at decision threshold $t_j \in \theta_i$
    push r onto C
allocate an array $P[2, C^2 \times F]$
// temporary store points (*fpr*, *tpr*) of fused responses
foreach $bf \in$ *BooleanFunctions* **do**
  for $i \leftarrow 1$ **to** M **do**
    for $j \leftarrow 1$ **to** M **do**
      $r \leftarrow bf(C[i], C[j])$ // combine responses
      compute $p \leftarrow (tpr, fpr)$ using r and lab
      push p onto P
compute composite $ROCCH$ of all ROC points in P
map each emerging points P_e on $ROCCH$ into a 3-tuples:
$P_e \leftarrow <(S_i, t_j), (S_i, t_j), \text{bf} >$ //where $i = \{1, \dots, L\}$ and $t_j \in \theta_i$
return $ROCCH$ along with all emerging points $\{P_1, \dots, P_e\}$

Algorithm 3: $WPIBC(S_1, \dots S_K, T_1, \dots T_K, lab)$: Weighted Pruning Iterative Boolean Combination

input: scores of K soft detectors $\{S_1, \dots S_K\}$ on a validation set along with their thresholds $\{T_1, \dots T_K\}$, and true labels lab of size $|lab|$.
output: a new composite $ROCCH$—consturcted by $|R_{iter}|$ (size of R_{iter}) combination responses or $|R_{iter}|$ emerging points. Each point is a sequential combination on average of five crisp detectors using four Boolean functions.

call pruning function //prune redundant soft and crisp detectors
$(B < S_1, \dots S_L >, \theta < \theta_1, \dots, \theta_L >) \leftarrow PSCDs(S_1, \dots S_K, T_1, \dots T_K, lab)$
// where $L \ll K$ is the number of selected diverse base soft detectors
set *BooleanFunctions* $\leftarrow \{a \wedge b, \neg a \wedge b, a \wedge \neg b, \neg(a \wedge b), a \vee b, \neg a \vee b, a \vee \neg b, \neg(a \vee b), a \oplus b, a \equiv b\}$
iter $\leftarrow 1$
// combine the first two ROC curves of the first two diverse base soft detectors
let $m_1 \leftarrow$ number of points in first curve $ROC(S_1, \theta_1)$
let $m_2 \leftarrow$ number of points in second curve $ROC(S_2, \theta_2)$
allocate an array $P[2, m_1 \times m_2]$ //temporary store the points of fused responses
foreach $bf \in$ *BooleanFunctions* **do**
  for $i \leftarrow 1$ **to** m_1 **do**
    $r_1 \leftarrow S_1 \geq t_i$ // temporary responses at decision threshold $t_i \in \theta_1$
    for $j \leftarrow 1$ **to** m_2 **do**
      $r_2 \leftarrow S_2 \geq t_j$ //temporary responses at decision threshold $t_j \in \theta_2$
      $r_{12} \leftarrow bf(r_1, r_2)$ // fuse responses
      compute $p \leftarrow (tpr, fpr)$ using r_{12} and lab
      push p onto P
compute $ROCCH_{iter}$ of all combination ROC points in P
map each emerging points p_e on $ROCCH_{iter}$ into a 3-tuples:
$p_e \leftarrow <(S_1, t_i), (S_2, t_j), \text{bf} >$
store all emerging points p_e on $ROCCH_{iter}$ onto $R_{1:2}$
// combine rest of the ROC curves of rest of the L-2 diverse base soft detectors
for $b \leftarrow 3$ **to** L **do**
  let $n_e \leftarrow$ number of emerging points in $R_{1:b-1}$
  let $m_b \leftarrow$ number of points in l $ROC_b(S_b, \theta_b)$ curve
  allocate an array $P[2, n_e \times m_b]$ //temporary storage of fused responses
  foreach $bf \in$ *BooleanFunctions* **do**
    for $i \leftarrow 1$ **to** n_e **do**
      $r_1 \leftarrow R_{1:b-1}(i)$ // responses from immediate previous combinations
      for $j \leftarrow 1$ **to** m_b **do**
        $r_2 \leftarrow S_b \geq t_j$ //temporary responses at decision threshold $t_j \in \theta_b$
        $r_{12} \leftarrow bf(r_1, r_2)$ // fuse responses
        compute $p \leftarrow (tpr, fpr)$ using r_{12} and lab
        push p onto P
  update $ROCCH_{iter}$ of all combination ROC points in P
  map each emerging points p_e on $ROCCH_{iter}$ into a 3-tuples:
  $p_e \leftarrow <R_{1:b-1}(i), (S_b, t_j), \text{bf} >$
  store all emerging points p_e on $ROCCH_{iter}$ onto $R_{1:b}$
store all the emerging points to reach on the final $ROCCH_{iter}$ onto $R_{iter} \leftarrow R_{1:L}$
set *maxiter* and *tol* // maximum number of iterations and tolerance
iter $\leftarrow 2$ to *maxiter*
  repeat steps 2 to 33 with $L + 1$ ROC curves:
  $ROC(R_{iter-1})$ *and* $ROC(S_1, \theta_1), \dots, ROC(S_L, \theta_L)$
  if $(AUCH_{iter} \leq AUCH_{iter-1} + tol)$ **then**
    break // stop further iteration
return $ROCCH_{iter}$ and R_{iter}

curves) and 12 pruned redundant soft detectors (black dotted ROC curves). Similarly, Fig. 3 (b) shows the experiment on CANALI-WD dataset, where we have only three base soft diverse detectors and 17 pruned redundant soft detectors. At the end of Phase1, all the selected base soft diverse detectors $L \ll K$ (stored in B) are then fed into Phase 2 of Algorithm 1.

***Phase2-Pruning Using Unweighted Kappa*:** The second phase of Algorithm 1 leverages the *MinMax-Kappa* pruning method [45], one of the two pruning methods of PBC using unweighted kappa, to select the complementary crisp detectors. Since the base soft detectors selected in Phase1 are diverse, we apply the *MinMax-Kappa* pruning method on each base soft detector individually instead of brute-force search like in PBC. We compute the unweighted kappa coefficient kp between a base soft detector's decision vector (or crisp detector) and the true decision labels (or ground truth), same as in PBC. If n_b is the number of decision levels on a base detector's scores vector S_b, then we obtain n_b crisp detectors. Now, we compute unweighted kappa coefficients of n_b crisp detectors and sorted them in ascending order. According to *MinMax-Kappa*, the accurate crisp detectors should reside close to $kp \approx kp_{max}$ and their complementary crisp detectors should reside close to $kp \approx kp_{min}$. However, we have to set the number of crisp detectors and the ratio of them to be selected close to kp_{max} and kp_{min}. We set the ratio is 50%, same as in *MinMax-Kappa*. Moreover, before selecting the complementary crisp detectors, we have to filter out the trivial crisp detectors (giving always either positive or negative responses) whose kp is close to zero.

In the running example shown in Fig. 2, Phase2 selects two diverse base soft detectors D_1 and D_3 with four different thresholds. Therefore, each base soft detector produces four crisp detectors at four different levels or thresholds. The responses $R(D(S) \geq \theta)$ of each crisp detector for 25 instances and their corresponding true labels (ground truth) are shown in Fig. 2(c). Fig. 2(d) shows the unweighted kappa values sorted in ascending order for each crisp detector of two base soft detectors D_1 and D_3.

Consider a ratio of 50% and the number of crisp detectors to be selected to be two. Therefore, from Fig. 2 (d), we obtain, $kp_{max} \approx 0.62$ and $kp_{min} \approx 0$ for D_1. Similarly, for D_3, $kp_{max} \approx 0.59$ and $kp_{min} \approx 0$. However, the trivial crisp detectors, one for D_1: $R(S \geq 0)$; and two for D_3: $R(S \geq 1)$ & $R(S \geq 0)$ should be filtered out first. Fig. 2 (d) shows the filtered trivial crisp detectors (large diagonal marker with cross sign). Since the ratio is 50%, from each base soft detector, one crisp detector should be selected close to kp_{max} and another one should be selected close to kp_{min}. Fig. 2 (d) shows the four selected complementary crisp detectors (two from each base soft detector, marked with large circle marker).

In general, if the number of selected complementary crisp detectors from a selected base soft detector is m (i.e., *m/2* close to kp_{max} and *m/2* close to kp_{min}), then the total number of selected crisp detectors will be $M = m * L$, where L is number of selected base soft detectors (selected from Phase1). We tested m with different setting (*l=4, 8, 12, 16, and 20*) and obtained best results for *m = 12*.

TABLE II
THE WORST-CASE TIME COMPLEXITY OF PRUNING AND WITHOUT PRUNING BASED BOOLEAN COMBINATION METHODS

Methods	Pruning	Boolean Combination
BBC2	NA	$\mathcal{O}(N^2)$
IBC	NA	$\mathcal{O}(T^2 + N)$
PBC	$\mathcal{O}(N(\log N + 1))$	$\mathcal{O}(U^2)$
WPBC2	Phase1: $\mathcal{O}(K^2)$ Phase2: $\mathcal{O}(K * (T(\log T + 1)))$	$\mathcal{O}(M^2)$
WPIBC		$\mathcal{O}(m^2 + M)$

In Fig. 4, we show a more realistic example, using ADFA-LD dataset. In this figure, we have M complementary crisp detectors (red bold points) selected from L diverse base soft detectors (selected in Phase1) using unweighted kappa-based *MinMax-Kappa* pruning technique. Fig. 4 (a) shows the results under the space of *kp-fpr* and Fig. 4 (b) shows the results under the space of *kp-tpr*. Fig. 5 also shows the selected total $M = 96$ complementary crisp detectors from the $L = 8$ diverse base soft detectors under the ROC space.

Phase3-Boolean Combination Techniques: The third phase combines the selected complementary crisp detectors using Boolean functions. The first combination approach called Weighted Pruning Pair-wise Boolean Combination (WPBC2), shown in Algorithm 2, combines all possible pairs of complementary crisp detectors (selected from Phase1 and Phase2) same as in BBC2. In contrast with BBC2, WPBC2 fuses only the complementary crisp detectors instead of using Brute-force (i.e., all available crisp detectors). The second approach called Weighted Pruning Iterative Boolean Combination (WPIBC), shown in Algorithm 3, combines the complementary crisp detectors of each diverse base soft detectors sequentially same as in IBC. The difference is that WPIBC only combines the most diverse base soft detectors after pruning all the redundant soft detectors. As we will show in the evaluation section, both WPBC2 and WPIBC Boolean combination approaches using only $M \ll N$ complementary crisp detectors of $L \ll K$ diverse base soft detectors improved the true positive rate when the false tolerance is almost close to zero.

B. Complexity Analysis

Suppose, we have K soft detectors with $S_k\{k = 1 \dots K\}$ scores using a validation set V. Let the number of decision thresholds on the scores S_k of each soft detector is constant and the size is T. And let $N = K * T$ be the total number of crisp detectors.

The 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] [43]. The worst-case time complexities of the proposed and existing Boolean combination methods are given in Table II. The pairwise combination of N crisp detectors employed in BBC2,

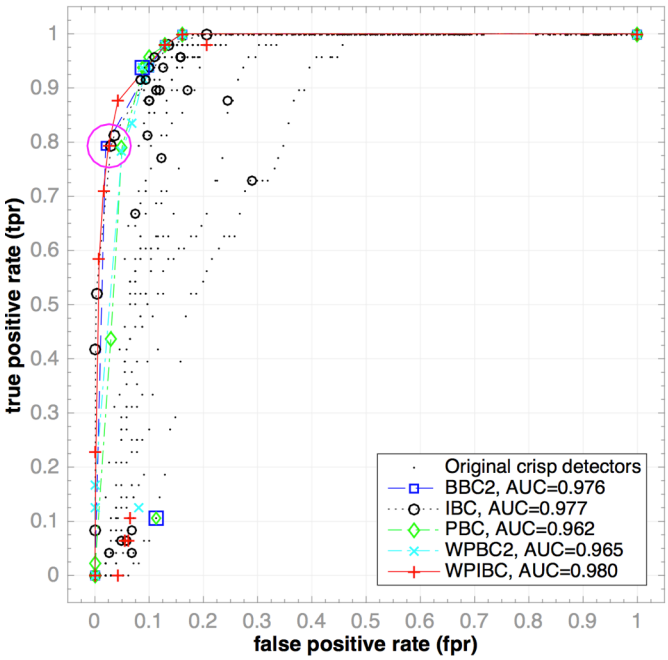


Fig. 6. Algorithm comparisons on ADFA-LD dataset where one fold is used for validation and four folds are used for testing.

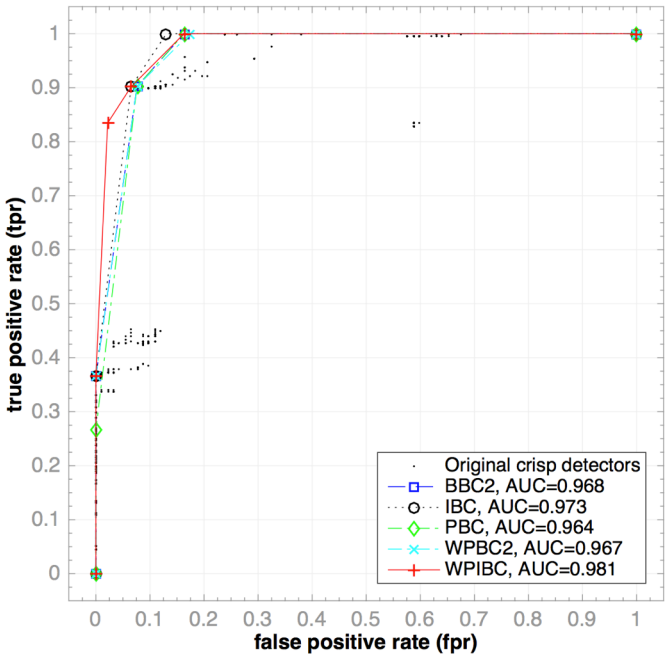


Fig. 7. Algorithm comparisons on CANALI-WD dataset where one fold is used for validation and four folds are used for testing.

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

Our recent pruning approach [43] used the kappa-error diagrams or simply called unweighted kappa coefficient to decide which ensemble members can be pruned with maintaining a similar overall accuracy. Although PBC reduces the impractical exponential computation time for BBC2 to $\mathcal{O}(N(\log N + 1))$, the performance at low false alarm values is also decreased (details in Section VI). This is because PBC selects $U \ll N$ complementary crisp detectors over the whole N converted crisp detectors, it cannot consider the diversity among the individual soft detectors.

The proposed pruning technique is more general as it ensures the diversity among both of the individual soft and crisp detectors instead of using N crisp detectors. As shown above, the total number of crisp detectors, N, depends on two important parameters K and T. Phase1 in the proposed weighted pruning technique reduces the size of the ensemble from K to L diverse soft detectors, by pruning the redundant ones. As shown in Fig. 3 (a), out of K=20 soft HMM detectors, Phase1 selects only L=8 HMMs for ADFA-LD dataset and only L=3 HMMs for CANALI-LD dataset (Fig. 3 (b)). Then, Phase2 optimizes the size of T of each selected base diverse soft detector (L) to $m<<T$ by pruning all the trivial and redundant crisp detectors. Here, m is a user defined parameter and set based on the experimental results using validation set (e.g., in this experiment, $m = 12$ gives the best result for both datasets). At the end, the proposed pruning methods always selects $M = L * m$ complementary crisp detectors.

Therefore, the worst-case time complexity required by the proposed pruning technique to select M complementary crisp detectors is $\mathcal{O}(K^2 + K * (T(\log T + 1)))$; where, Phase1 requires about K^2 operations for computing and sorting the AUC and the weighted kappa of K soft detectors, in order to select L diverse base soft detectors. And in Phase2, each base diverse soft detector (L) requires about $T(\log T + 1)$ operations for computing and sorting the unweighted kappa for T crisp detectors, in order to select $m \ll T$ complementary crisp detectors. Therefore, in case of worst-case, Phase1 selects all K soft detectors (i.e., $L = K$). So, the worst-case time complexity for Phase2 requires about $(K * (T(\log T + 1)))$ operations, in order to select a total of $M = K * m$ complementary crisp detectors. At the end of pruning Phases, Phase3 combines the decisions of M complementary crisp detectors. In Phase 3, the worst-case time complexity for the proposed weighted pruning pairwise Boolean combination (WPBC2) is about $\mathcal{O}(M^2)$ Boolean operations and for the proposed weighted pruning iterative Boolean combination (WPIBC) is about $\mathcal{O}(m^2 + M)$ Boolean operations, where $M \ll N$ and $m \ll T$.

VI. Experiments and Comparison

We experimented with the proposed pruning approach on two system call datasets: ADFA Linux Dataset (ADFA-LD) [6] and CANALI Window Dataset (CANALI-WD) [47]. The experimental results are compared with BBC2 [1] and IBC [21] without pruning. We also compared our approach to PBC that we proposed in previous work [43].

ADFA-LD dataset: ADFA-LD consists of normal and anomalous sequences of system calls collected from Ubuntu [6]. A normal sequence of system calls of a process is collected from the system call traces while it is executed under the normal conditions. An anomalous sequence of system calls of an attack is collected from the system call traces while it is executed against the system. There are in total 5,206 normal

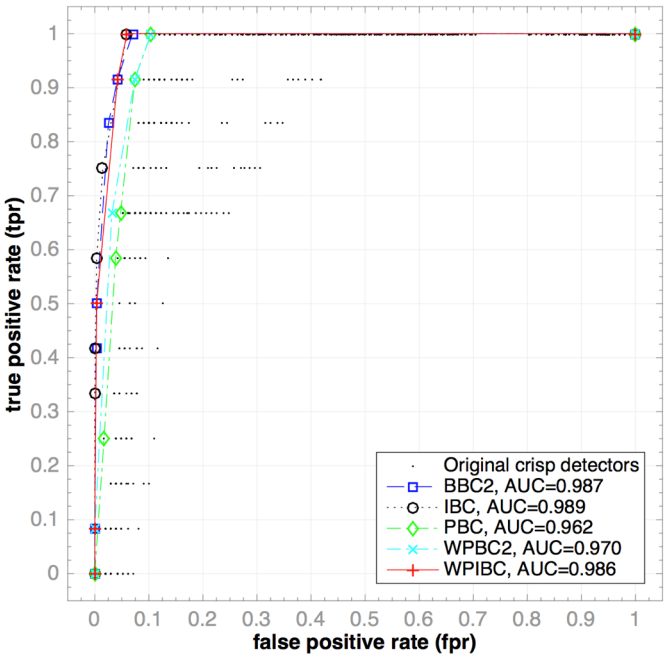


Fig. 8. Algorithm comparisons on ADFA-LD dataset where four folds are used for validation and one fold is used for testing in 5FCV.

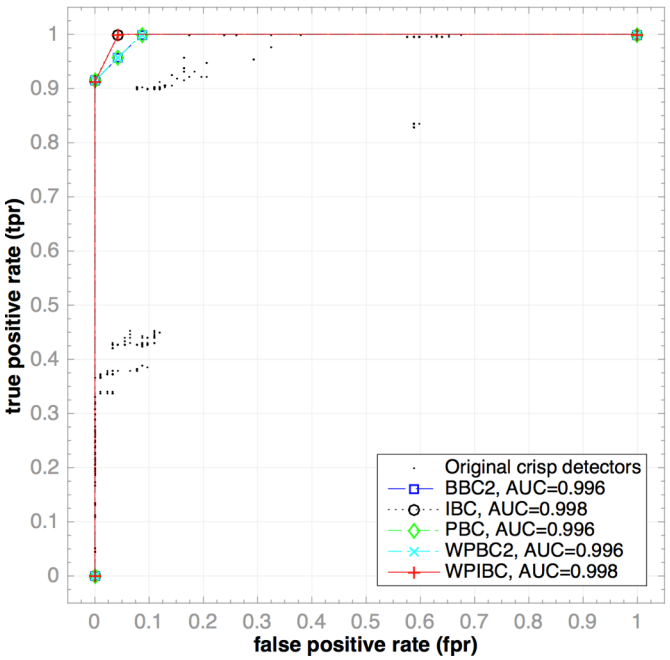


Fig. 9. Algorithm comparisons on CANALI-WD dataset where four folds are used for validation and one fold is used for testing.

traces collected from various normal Unix-based processes such as web browsing and Latex document preparations. The dataset contains 60 attack traces by exercising six different types of attacks: web-based exploitation, simulated social engineering, poisoned executable, remotely triggered vulnerabilities, remote password brute-force attacks, and system manipulation. In training, we use the 833 normal traces same as in [6] to train the 20 discrete-time ergodic HMMs

TABLE III

AVERAGE (AVG), MAXIMUM (MAX), AND MINIMUM (MIN) AUC VALUES AND TRUE POSITIVE RATE (TPR) WITH FALSE POSITIVE RATE (FPR)<=0.002, AND THEIR STANDARD DEVIATIONS (STD) OVER THE 5FCV (TRAIN ON ONE FOLD AND TESTED ON FOUR FOLDS).

		AUC values				***tpr* with *fpr*<=0.002**			
Datasets	**methods**	**avg**	**max**	**min**	**std**	**avg**	**max**	**min**	**std**
	without pruning methods								
ADFA-LD	BBC2	0.98006	0.9852	0.9731	0.0044	0.38334	0.5	0.2292	0.1246
	IBC	0.979	0.983	0.972	0.0042	0.25414	0.4792	0.1665	0.1329
CANALI-WD	BBC2	0.96824	0.9726	0.9601	0.0049	0.36716	0.3739	0.3618	0.0046
	IBC	0.97156	0.9799	0.9612	0.0069	0.36716	0.3739	0.3618	0.0046
	with pruning methods								
ADFA-LD	PBC	0.96762	0.9766	0.9608	0.0078	0.09576	0.2297	0.0208	0.0877
	WPBC2	0.96604	0.9741	0.9602	0.0059	0.11886	0.246	0.054	0.0785
	WPIBC	**0.97762**	0.9788	0.9767	**0.0007**	**0.37498**	0.5208	0.2083	**0.0474**
CANALI-WD	PBC	0.96808	0.9726	0.9601	0.0049	0.24197	0.2639	0.2118	0.0046
	WPBC2	0.96816	0.9729	0.9601	0.0051	0.27716	0.3739	0.2218	0.0071
	WPIBC	**0.96994**	0.9808	0.9541	**0.0021**	**0.34462**	0.3739	0.3225	**0.0034**

TABLE IV

AVERAGE (AVG), MAXIMUM (MAX), AND MINIMUM (MIN) AUC VALUES AND TRUE POSITIVE RATE (TPR) WITH FALSE POSITIVE RATE (FPR)<=0.002, AND THEIR STANDARD DEVIATIONS (STD) OVER THE 5FCV (TRAIN ON FOUR FOLDS AND TESTED ON ONE FOLD).

		AUC values				***tpr* with *fpr*<=0.002**			
Datasets	**methods**	**avg**	**max**	**min**	**std**	**avg**	**max**	**min**	**std**
	without pruning methods								
ADFA-LD	BBC2	0.98918	0.99945	0.9829	0.0043	0.4500	0.5833	0.2500	0.1263
	IBC	0.99112	0.9939	0.9887	0.0021	0.41668	0.5000	0.3333	0.0589
CANALI-WD	BBC2	0.97288	0.9963	0.9469	0.0208	0.58648	0.9142	0.3591	0.2963
	IBC	0.98274	0.9981	0.9679	0.0127	0.60722	0.9142	0.3694	0.2798
	with pruning methods								
ADFA-LD	PBC	0.95648	0.9626	0.9533	0.0037	0.0000	0.0000	0.0000	0.0000
	WPBC2	0.9661	0.9703	0.9643	0.0024	0.16666	0.3333	0.0000	0.1317
	WPIBC	**0.98724**	0.992	0.9827	0.0033	**0.49998**	0.5833	0.3333	0.1020
CANALI-WD	PBC	0.97288	0.9963	0.9469	0.0208	0.58648	0.9142	0.3591	0.2963
	WPBC2	0.9736	0.998	0.9469	0.0217	0.58648	0.9142	0.3591	0.2963
	WPIBC	**0.98028**	0.9981	0.9647	0.0151	**0.5981**	0.9142	0.3591	0.2034

(i.e., K=20 soft detectors) with various values. The rest of the 4373 normal traces and the 60 anomalous traces are used for evaluation.

CANALI-WD dataset: CANALI-WD consists of two normal datasets called goodware and anubis-good and two malware datasets called malware and malware-test [47]. The goodware dataset contains a massive amount of 180 GB execution traces of normal day-to-day operations which are collected from 10 different machines. The anubis-good dataset contains the traces of 36 benign applications executed under Anubis [61]. The malware dataset is a collection of execution traces of 6,000 malware samples including a mix of all the existing categories (botnets, worms, dropper, Trojan horses, etc.), which are randomly extracted from Anubis [61]. The final malware-test dataset is a collection of execution traces of 1,200 malware samples which are collected from a different machine than the normal ones used for Anubis. In training, we use the anubis-good dataset and the traces for nine out of 10 machines in the goodware dataset (same as in [47]) to train 20 soft HMMs detectors with various values. In contrast to [47], however, where the malware dataset was also used to train the models, we only use malware for testing. This is because an anomaly detector mainly models the normal behavior of a system. Therefore, the rest of the 23 traces of the tenth machine in the goodware dataset, 5,855 traces from malware dataset, and 1,133 traces from malware-test dataset are used for evaluation.

A. Experimental Setup

We use a stratified 5-Fold Cross Validation (5FCV) technique, same as in [43], on the testing set for the evaluation of the proposed pruning approach. Since the ratio between the normal and anomalous traces in both datasets is not balanced, we applied stratified 5FCV to partition the normal and anomalous sets separately. This is because we want to keep the same ratio (normal to anomalous) to guarantee that all folds include the normal and anomalies traces. Therefore, for ADFA-LD dataset, each fold contains 874 traces selected randomly from the 4373 normal traces and 12 attacks traces selected randomly from the 60 attack traces. Similarly, for CANALI-WD dataset, each fold contains four traces selected randomly from the 23 normal traces and 1,397 traces selected randomly from the 6,988 anomalous traces. However, as we followed the same setting as in PBC [43] instead the way of standard cross validation, we also used one fold for validation and the remaining four folds for testing on the both ADFA-LD and CANALI-WD datasets.

As described in Section III, we apply the BW algorithm on the validation set to learn the parameters of an HMM with setting the random initialization of A, B and π, and M = 340 distinct system call symbols for ADFA-LD dataset and M = 89 distinct symbols for CANALI-WD dataset. Since a single HMM with a predefined number of states N may have limited chances to fit the underlying structure of the data (as noted in Section III), 20 different discrete-time ergodic HMMs (i.e., 20 soft detectors) are trained with various $N = 10, 20 \ldots 200$ values. For each state value N, we repeated the training process ten times with a different random initialization of A, B and π to avoid the local minima, and the HMM that gives the highest AUC value on the validation set is selected for Boolean combination.

B. Results and Comparison

We mainly focus on how the proposed pruning based Boolean combination approaches can reduce the computation time (as discussed in Section V) of the BBC2 and IBC techniques while maintaining or improving the detection accuracy and reducing the false alarm rate.

Fig. 6 and 7 show the AUC results in the ROC space for the proposed weighted pruning techniques on ADFA-LD and CANALI-WD datasets. We can see that the ROC curve of the proposed pruning based WPIBC shows slightly better AUC than IBC. In particularly, WPIBC is able to ensure the diversity among the fused crisp detectors (selected using unweighted kappa at Phase 2 in Algorithm 1) where each crisp detector comes from the selected diverse base soft detectors (selected using weighted kappa at Phase 1 in Algorithm 1). Therefore, in contrast to IBC, where the order of combination responses in each iteration is the order of all the available soft and crisps detectors, WPIBC maintains the order of combination responses in each iteration among the selected diverse soft and crisp detectors (see details in Algorithm 1 and Algorithm 3). For instance, to achieve the final operating points denoted in Fig. 6 with a large pink circle, WPIBC uses only five selected complementary crisp detectors (red bold plus marker points) and four Boolean operations, whereas IBC uses 17 crisp detectors (black bold circle marker points) and 16 Boolean operations.

Compared to BBC2, although the AUC of WPBC2 is slightly low, WPBC2 maintains the same AUC of PBC shown in Fig. 6 and 7. However, WPBC2 overcomes the exponential time complexity problem of BBC2 by pruning the redundant and trivial crisp detectors, in fact, without pruning, BBC2's time complexity is exponential with respect to the number of detectors (N^2) [1] [43].

Table III shows the maximum detection accuracy (*tpr*) achieved by each technique for a fixed (almost close to zero) *fpr* value of 0.002, all values are averaged over the 5FCV.

For ADFA-LD dataset, although the AUC values of all pruning methods are almost equal, the *tpr* of PBC with MinMax-Kappa pruning technique is the worst. The *tpr* of WPIBC is almost equal to that of BBC2 method, and slightly better than that of IBC method. Moreover, the standard deviation of WPIBC is also good as compared to the other methods. For the CANALI-WD dataset, the *tpr* of WPIBC is still better than PBC and WPBC2 pruning techniques, and almost equal to BBC2 and IBC that do not use pruning techniques. Through this analysis, we observed that when the proposed weighted pruning technique combines the selected complementary crisp detectors iteratively (i.e., called WPIBC), it achieves similar results to that of IBC. Particularly, when we compared the results with the *tpr* where the maximum *fpr* is almost equal to zero (0.002), both WPIBC and WPBC2 outperform PBC. And the results demonstrate

that the proposed pruning approach is more general and applicable to either pair-wise Boolean combinations (WPBC2) and iterative Boolean combinations (WPIBC).

Moreover, we tested the proposed pruning approach by using the standard way of 5FCV that is four folds are used in validation and one fold is used in testing. With this setting, the results of one fold of 5FCV are demonstrated in Fig. 8 for ADFA-LD dataset and in Fig. 9 for CANALI-WD dataset. Table IV shows the average results over the 5FCV with this standard setting of 5FCV. From Fig. 8 and Fig. 9, we can see that for both datasets our proposed pruning based Boolean combination approaches is able to achieve the same performance (in terms of AUC, *fpr* and *tpr*), while reducing the time complexity, the number of crisp detectors, and the number of Boolean combinations. For CANALI-WD dataset, we got almost equal results with the original approaches (which use all crisp detectors), and the highest value of *tpr* = 0.91 when the false alarm rate is zero, given in Table IV. However, for ADFA-LD dataset, we observed a great difference between the proposed pruning approach and the PBC. When the average *tpr* = 0.49 for WPIBC (with the limit of maximum *fpr* is equal to 0.002), it is equal to zero for PBC and 0.17 for WPBC2. For example, from the Fig. 8, we got *tpr* =0.51 (when *fpr*<=0.002) for WPIBC, it is still zero for PBC.

C. Cost Analysis

Table V shows the cost that is the combination time and the number of Boolean operations are required by each method during the validation and testing phases. The values are averaged over the 5FCV on the ADFA-LD dataset. All 5FCV executions are performed on a 3.1 GHz Intel Core i7 CPU machine with 16 GB of RAM and a 17x5400 rpm hard disk.

We can see that although the pruning time of the proposed approach is slightly more than the PBC, WPIBC reduced the combination time and the number of Boolean operations to almost half compared to IBC. The total computation time, including pruning and combination during validation of WPIBC was 7.9 seconds whereas PBC took 16.6 seconds. Furthermore, in testing, WPIBC also reduced the number of combined crisp detectors by almost half than the number required by IBC (5 instead of 11). We can see in Fig. 6 that WPIBC requires on average five crisp detectors while IBC requires 11 crisp detectors to achieve a single point on the final composite ROCCH. Similarly, WPBC2 always requires only two crisp detectors similar to BBC2 and PBC to achieve a single point on the final ROCCH. Therefore, the proposed pruning approach is more general and it can be applicable to both pair-wise and iterative Boolean combinations. However, based on the computation time and the number of combined Boolean operations, WPIBC is more desirable in order to obtain better accuracy while reducing the false alarm rates (as shown in Table III and Table IV).

From the worst-case time complexity given in Table II, we can see that the proposed pruning approach reduces the total number of crisp detectors i.e., $N = K * T$ by optimizing two important parameters of K and T in Phase1 and Phase2 respectively. For example, Phase1 of the proposed weighted pruning approach selects only $L = 3$ diverse ensembles of HMM soft detectors out of $K = 20$ HMM soft detectors (shown in Fig. 3b) for CANALI-WD dataset. As a result, Phase2 computes the unweighted kappa only for about 300 (i.e., $L * T$ and let say $T = 100$) crisp detectors, in order to select only $M = 36$ (i.e., $M = L * m$, where $m = 12$) complementary crisp detectors. Whereas, PBC always computes the unweighted kappa for about $N = 2000$ (i.e., $N = K * T$) crisp detectors, in order to select $U = 50$ complementary crisp detectors. Moreover, since PBC can not ensure the diversity among the ensembles of soft HMM detectors, the probability of selecting the redundant complementary crisp detectors or rejecting the other diverse crisp detectors is also high. In fact, it is reported in Table III and Table IV that PBC significantly reduced the *tpr* with a low false alarm as compared to the other approaches due to the rejection of some diverse complementary crisp detectors.

TABLE V
COST ANALYSIS (VALUES ARE AVERAGED OVER 5FCV) IN TERMS OF PRUNING AND COMBINATION TIME (S), AND NUMBER OF BOOLEAN OPERATIONS APPLIED DURING VALIDATION PHASE, AND THE NUMBER OF COMBINED CRISP DETECTORS REQUIRED TO ACHIEVE EACH VERTEX ON ROCCH DURING TESTING PHASE

Methods	Validation phase			Testing phase
	Pruning time (s)	Combination time (s)	# Boolean operations	# combined crisp detectors
BBC2	NA	16364	4,000,000	2
IBC	NA	11	11,000	11
PBC	1.6	15	19, 701	2
WPBC2	1.9	19	21,701	2
WPIBC	1.9	6	5, 000	5

VII. EFFECTS OF WEIGHTED PRUNING BASED BOOLEAN COMBINATION

For any ensemble based Boolean combination algorithms, increasing the accuracy is highly dependent on the diversity among the fused soft/crisp detectors (i.e., the level of disagreement among the fused soft/crisp detectors should be high). Although the existing ensemble based BBC2 and IBC Boolean combination techniques implicitly fused such diverse soft/crisp detectors and showed higher accuracy, they face the challenges of computation time and complexity because of fusing all the possible pair of crisp detectors from all the available soft detectors (as discussed in Section IV and V; and reported in Table III). In addition, the accuracy of IBC is also dependent on the order of combinations. In fact, with the increase of number of soft detectors, the computation time and complexity increase exponentially for BBC2 and linearly for IBC (discussed in Section IV).

To be clear, we tested the proposed approach using 50 available soft HMM detectors (i.e., on average 5000 crisp detectors), trained with various $N = 5, 10, \ldots 250$ values on CANALI-WD dataset. The results are shown in Fig. 10, where the values are transformed into a logarithmic scale. It is clear that when we apply the proposed weighted pruning approach (top one in Fig. 10), a noticeable improvement can be observed in the reduction of the number of Boolean

operations. Particularly, WIBC significantly reduces the number of Boolean operations as compared to other approaches. For example, BBC2 requires 25 million (7.4 in logarithmic scale) Boolean operations for 50 soft detectors, whereas, WPBC2 uses only 3,600 (3.4) operations. Similarly, when IBC requires 15 thousand (4.2 in logarithmic scale) Boolean operations, WIBC uses only 204 (2.3) operations. As a result, we can state that WPBC2 6944 times faster than BBC2 and WPIBC 73 times faster than IBC for 50 soft detectors. Moreover, from the 30 soft detectors, WPBC2 and WIBC always select five diverse soft detectors with the increase of the number of soft detectors, and thus, reach a constant number of Boolean operations.

The bottom part of Fig. 10 compares the computation time (including pruning and combination time together for pruning based approaches). We can see that WPBC2 and WIBC reported the lowest computation times as compared to other approaches. For example, WPBC2 is ten thousand times (seconds) faster than BBC2 and WPIBC is two times faster than IBC during validation phase using 50 available soft detectors. Moreover, from the 30 soft detectors, although the pruning time for WPBC2 and WIBC increase slightly with the increase of number of soft detectors, the combination time remains same as both are always using only five selected diverse soft detectors. Compared to PBC pruning approach where the pruning and combination time both are increasing linearly with the increase of number of soft detectors.

In fact, PBC shows worst result when the false alarm is almost zero for both ADFA-LD and CANALI-WD datasets (given in Table III and Table IV). On the other hand, the accuracy with almost zero false alarm is the desired expected solution for deploying a ADS in a real world application. The reason is that PBC also uses all the available soft detectors to select a subset of complementary crisp detectors without ensuring the diversities among the use of soft detectors. As a result, the redundant soft detectors produces redundant crisp detectors, and thus it increases the probability of selecting these redundant copies if anyone is selected as a complementary crisp detector by MinMax-Kappa pruning technique of PBC.

The proposed WPBC2 and WPIBC weighted pruning techniques select the most diverse base soft detectors from the available soft detectors using weighted kappa. For instance, from the Fig. 3 (a), eight diverse base soft detectors are selected while 12 are pruned as for redundant copies for ADFA-LD dataset. Similarly, from the Fig. 3 (b), only three diverse base soft detectors are selected while 17 are pruned for CANALI-WD dataset. As the selected base soft detectors are diverse, the converted crisp detectors from them might also be diverse as well. Therefore, when we apply the MinMax-Kappa pruning technique on each selected diverse base soft detectors individually, there has no chance for the selection of redundant complementary crisp detectors. As a result, our proposed pruning technique shows better accuracy when the false alarm is close to zero compared to PBC for both datasets (given in Table III and Table IV). Moreover, the proposed weighted based pruning approach is more general as we can combine the selected diverse soft/crisp detectors either pair-wise or iteratively same as in BBC2 or IBC Boolean combination techniques.

Although the proposed approach is experimentally validated only on HIDS using system call data, it can be applied in other

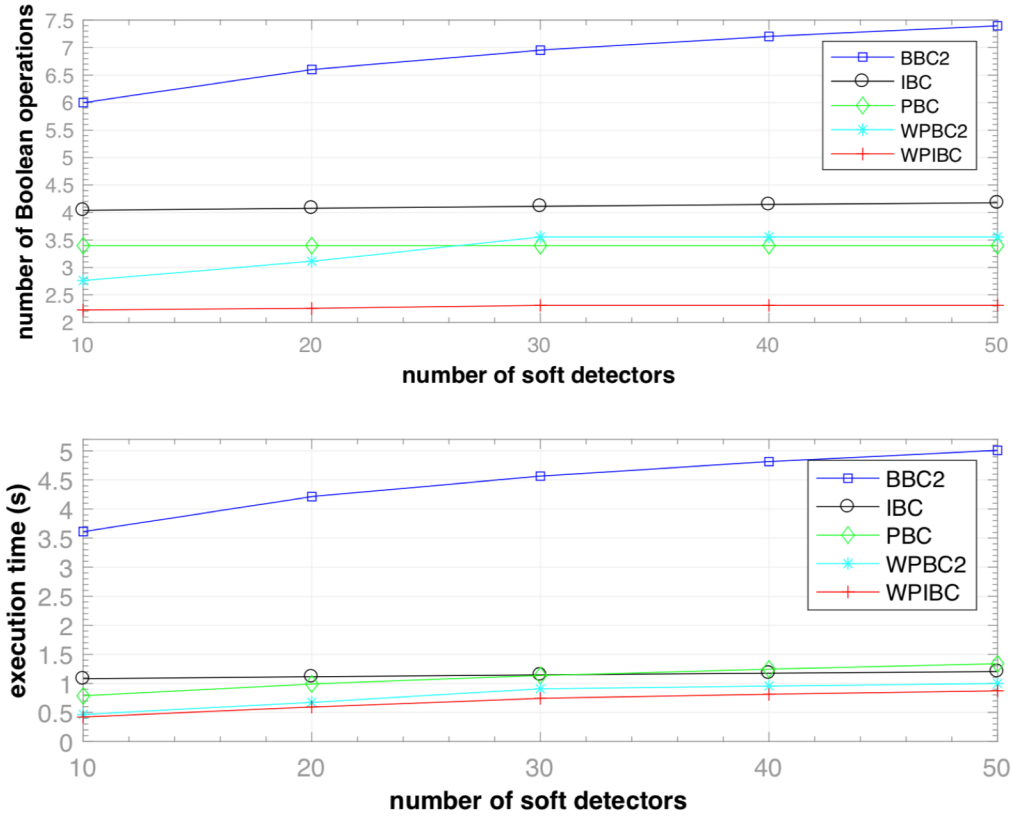


Fig. 10. Algorithm's computation time and complexity analysis on the validation subset of CANALI-WD dataset

application domains particularly, where one model does not formulate the complex normal behaviors of a system. In that case, we can train ensemble detectors with considering various normal behaviors. Then, the proposed method may be a good one for pruning and combining the multiple detector's decisions. For example, detecting programming errors (i.e., software bugs) and root causes in a complex computer programming system [64][65]. Fosdick et al. [66] reported that a computer program is strongly related to the computation patterns of input data and thus useful for detecting the data flow anomalies. The sequences of operations i.e., the flows of data are assumed to be consistent and used them to model ensembles classifiers. A social or cultural event or road accident can also be detected using sensor and user (e.g., users of twitter, Facebook, etc.) generated data. For example, Pramod et al. [63] trained several linear Markov models by segmenting the non-linear traffic data and used them to detect the city events.

VIII. Limitations and Discussion

Our approach is limited to ensemble of homogeneous soft anomaly detectors (i.e., multiple HMMs). However, the input can be ensemble of heterogeneous soft and crisp anomaly detectors (e.g., STIDE [11], SVM [72], etc.). In fact, having different types of detectors should further increase the diversity in the ensemble and allow for improved performance [73]. Heterogenous detectors use different learning techniques and may commit different (and potentially complementary) type of errors, which increases the diversity in the ensemble. For example, OC-SVM models the normal behavior of a system using fixed-size feature vectors instead of sequential features like HMM; STIDE uses the Hamming distance, whereas HMM uses likelihood probability as a matching measure.

To adapt our approach to support heterogeneous detectors, we need to modify Phase1, which assumes the same thresholds of a base soft detector, which are the orders or levels for the weighted kappa for computing the diversity score. It may be more efficient to group them based on each modeling technique. Then, apply the Phase1 pruning technique on each group separately. For example, STIDE with various sliding window sizes can be used to produce many homogeneous soft detectors [11], which can then fed as input to Phase1.

Although the proposed approach significantly reduces the Boolean combination time (see Fig. 10) by pruning the number of combined soft (K) and crisp detectors (N), the worst-case time complexity, particularly, for the pruning phases (given in Table II), will be increased exponentially (K^2) with the increase of K. Therefore, for large values of K, the pruning approach may suffer from scalability problems. To address this limitation, we need resort to parallel processing techniques and platforms such as the Hadoop ecosystem [74][75].

Moreover, the proposed approach is dependent on the ROC space for pruning and combining the decisions of the selected complementary crisp detectors. However, here, the used ROC curves is a binary classification problem. Therefore, to extend the approach for multiclass classification problems, we need to work with a ROC curve for more than two classes and then adapt the Boolean combination and pruning techniques to accommodate multiple classes.

IX. Conclusion

The proposed effective pruning-based Boolean combination techniques analyze the diversities among the available ensemble soft detectors (HMMs) using weighted kappa (measures the agreement/disagreement between two soft detectors). Based on the weighted kappa coefficients, it selects a best subset of diverse base soft detectors while pruned all the redundant soft detectors. Each selected base soft detector is then converted into all the possible crisp detectors (at various decision thresholds) and used them for selecting a subset of complementary crisp detectors using unweighted kappa-based MinMax-Kappa pruning technique. At the end, we merge all the selected complementary crisp detectors and use them for Boolean combinations. The experimental evaluation on the two benchmarking ADFA-LD and CANALI-WD system call datasets verified the validation of the proposed method. We achieved much better results than the recent PBC pruning technique, particularly, when the false alarm is almost close to zero.

Our future plan is to investigate the proposed pruning approach using different diverse detectors and other datasets. Moreover, we also want to leverage Big Data platforms such as Hadoop and the MapReduce programming model in order to further improve the performance of our approach, especially when used with multiple heterogeneous ensemble soft detectors such as HMMs, One-class SVM, STIDE, and so on.

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 Cyber Security 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.

[45] A. Soudi, W. Khreich, and A. Hamou-Lhadj, "An Anomaly Detection System based on Ensemble of Detectors with Effective Pruning Techniques", 2015 IEEE International Conference on Software Quality, Reliability and Security, pp. 109-118, Aug. 2015.

[46] C.A.M. Valiquette, A.D. Lesage, and C. Mireille, "Computing Cohen's Kappa coefficients using SPSS MATRIX", Behavior Research Methods, Instruments, & Computers, 26(1), 60-61, 1994.

[47] D. Canali, A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, E. Kirda, A quantitative study of accuracy in system call-based malware detection, in: Proceedings of the 2012 International Symposium on Software Testing and Analysis, ISSTA 2012, ACM, New York, NY, USA, 2012, pp. 122 132. doi:10.1145/2338965.2336768.

[48] M. Stamp, A Revealing Introduction to Hidden Markov Models, Dec 11, 2015.

[49] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm," Journal of the Royal Statistical Society. Series B (Methodological) 39 (1), pp. 1-38, 1977.

[50] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. Machine Learning Journal, 42(3):203–231, March 2001.

[51] M. J. J. Scott, M. Niranjan, and R. W. Prager, "Realisable classifiers: Improving operating performance on variable cost problems," in Proc. 9th British Machine Vision Conf., P. H. Lewis and M. S. Nixon, Eds., vol. 1, University of Southampton, UK, 1998, pp. 304–315.

[52] J. Neyman and E. S. Pearson, "On the problem of the most efficient tests of statistical hypotheses," Phil. Trans. Royal Society of London A, vol. 231, pp. 289–337, 1933.

[53] M. A. Black and B. A. Craig, “Estimating disease prevalence in the absence of a gold standard,” Statistics in Medicine, vol. 21, no. 18, pp. 2653–2669, 2002.
[54] J. Daugman, “Biometric decision landscapes,” Cambridge U., UK, Tech. Rep. UCAM-CL-TR-482, 2000.
[55] Q. Tao and R. Veldhuis, “Threshold-optimized decision-level fusion and its application to biometrics,” Pattern Recognition, vol. 41, no. 5, pp. 852–867, 2008.
[56] S. Haker, W. M. Wells, S. K. Warfield, I.-F. Talos, J. G. Bhagwat, D. Goldberg-Zimring, A. Mian, L. Ohno-Machado, and K. H. Zou, “Combining classifiers using their receiver operating characteristics and maximum likelihood estimation,” Medical Image Computing and Computer-Assisted Intervention, vol. 3749, pp. 506–514, 2005.
[57] P. A. Flach and S. Wu, “Repairing concavities in ROC curves,” in Proc. 19th Int’l Joint Conf. on Artificial Intelligence, Edinburgh, Scotland, 2005, pp. 702–707.
[58] Cohen, J. (1960). A coefficient of agreement for nominal scales, Educational & Psychological Measurement, 20, 37–46.
[59] Agresti, A (1990). Categorical Data Analysis. New York: Wiley. (p. 367)
[60] Fleiss JL, Levin B, Paik MC (2003) Statistical methods for rates and proportions, 3rd ed. Hoboken: John Wiley & Sons.
[61] http://anubis.iseclab.org, 2011.
[62] V. Chandola, A. Banerjee, V. Kumar, “Anomaly Detection: A Survey” ACM Computing Surveys, vol. 41, no. 3, pp. 1-58, 2009.
[63] A. Pramod, T. Krishnaprasad, M. Surendra, S. Amit, and B. Tanvi, “Understanding City Traffic Dynamics Utilizing Sensor and Textual Observations,” in Proc. of the 13th AAAI Conf. on Artificial Intelligence, pp. 3793–3799, 2016.
[64] S. Hangal and M.S. Lam, “Tracking down software bugs using automatic anomaly detection,” In Proc. of the 24th Int. Conf. on Software Engineering (ICSE), pp. 291-301 ACM, NY, USA, 2002.
[65] A.L. Goel,K. Okumoto, “Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures,”IEEE Transactions on Reliability, vol. R-28, no. 3, August 1979.
[66] L.D. Fosdick, L.J. Osterweil “Data Flow Analysis in Software Reliability,” ACM Computing Surveys, vol. 8 no. 3, pp. 305-330 Sept. 1976.
[67] S.S. Murtaza, N.H. Madhavji, A. Hamou-Lhadj, M. Gittens, "Identifying Recurring Faulty Functions in Field Traces of a Large Industrial Software System," IEEE Transactions on Reliability, 64(1), pp. 269-283, 2014.
[68] W. Sha, Y. Zhu, M. Chen, T. Huang, “Statistical Learning for Anomaly Detection in Cloud Server Systems: A Multi-Order Markov Chain Framework”, IEEE Transactions on Cloud Computing, vol. PP, issue: 99, May 2017.
[69] A. Bovenzi, F. Brancati, S. Russo, A. Bondavalli, “An OS-level Framework for Anomaly Detection in Complex Software Systems”, IEEE Transactions on Dependable and Secure Computing, vol. 12, no. 3, pp. 366 - 372, 2015.
[70] J. Yang, X. Du, L. Zhou, S. Shan, B. Cui, “Research on the Identification of Software Behavior in Anomaly Detection”, 10th Int. Conf. on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 295 - 298, 2016.
[71] D. Gizopoulos, M. Psarakis, S.V. Adve, P. Ramachandran, S.K.S. Hari, D. Sorin, A. Meixner, A. Biswas, Xa. Vera, Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011, pp. 1–6, IEEE, 2011.
[72] W-H. Chen, S-H. Hsu, and H-P. Shen, “Application of SVM and ANN for intrusion detection”, Computers & Operation Research, vol. 32, no. 10, pp. 2617-2634, October 2005.
[73] W. Khreich, S.S. Murtazaa, A. Hamou-Lhadja, and C. Talhi, “Combining heterogeneous anomaly detectors for improved software security”, Journal of Systems and Software, February, 2017.
[74] Apache Hadoop. [Online]. Available: http://hadoop.apache.org/
[75] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, Communications of the ACM. Vol. 51(1), pp. 107–113, 2008.