A Counselors-Based Intrusion Detection Architecture

Silvio E. Quincozes[†], Carlos Raniery[*], Raul Ceretta Nunes[*], Célio Albuquerque[†], Diego Passos[†], Daniel Mosse[‡]

[†]Computer Science Department, Universidade Federal Fluminense
Niterói–RJ, Brazil
[*]Department of Applied Computing, Universidade Federal de Santa Maria
Santa Maria–RS, Brazil
[‡]Department of Applied Computing, University of Pittsburgh
Pittsburgh–PA, EUA

sequincozes@id.uff.br, {csantos,ceretta}@inf.ufsm.br, {celio,dpassos}@ic.uff.br, mosse@cs.pitt.edu

***Abstract*—Intrusion Detection Systems (IDSs) are a fundamental component of defensive solutions. In particular, signature-based IDSs aim to detect malicious activities on computer systems and networks by relying on data classification models built from a training dataset. However, classifiers performance can vary for each attack pattern. A common technique to overcome this issue is to use ensemble methods, where multiple classifiers are employed and a final decision is taken combining their outputs. Despite the potential advantages of such an approach, its usefulness is limited in scenarios where (i) multiple expert classifiers present divergent results or (ii) representative data are missing to detect a specific attack class. In this work, we introduce the concept of counselor networks to deal with conflicts from different classifiers by exploiting the collaboration between IDSs that analyze multiple and heterogeneous data sources. Our empirical results demonstrate the feasibility of the proposed architecture in improving the accuracy of the intrusion detection process.**

***Index Terms*—Intrusion Detection Systems (IDS), Multiple Sources Analysis, Machine Learning, Self-learning IDS**

I. Introduction

Nowadays there are several approaches for attackers to perform intrusive or malicious activities. In order to reduce the damage caused by those attacks, it is fundamental to implement and use IDSs. The purpose of an IDS is to monitor a data source and to identify when a suspicious activity takes place [1]. When an attack is properly identified, the appropriate countermeasures can be taken in order to mitigate or even eliminate its effects on the target system.

To distinguish between suspicious and normal activities, a plethora of data analysis techniques could be employed. Among those, classification techniques are the most used in the literature for detecting anomalies. They use a training database with samples of known attacks to assemble a classification model that can be later used for detecting abnormal behavior. One of the benefits of this approach is that it may be able to detect attacks for which there is no samples available during the training phase [2] [3] [4].

Selecting the most appropriate classification algorithm is a challenging task. There are diverse algorithms available in the literature, but there is no "one size fits all" classifier. In practice, a good classifier for identifying one class of attack is often inefficient for other classes. As a consequence, the development of a single classifier that covers all possible attacks seems unfeasible. This leads to a scenario where the output of an IDS may be unreliable [5].

In order to solve this issue, the literature proposes the use of methods to dynamically select the most appropriate classifier for each attack pattern [6], [7]. Still, the effectiveness of such approaches is limited in scenarios where multiple expert classifiers present divergent results. For example, the chosen classifier can still provide wrong results depending on the data available for analysis.

By considering the limitations of existing solutions, this work proposes an architecture that combines dynamic classifier selection with the analysis of multiple and heterogeneous data sources (*e.g.,* network statistics and application logs). More specifically, we introduce the concept of a counselor network that enables an IDS to solve conflicts based on advice from expert classifiers, thus leading to increased reliability of the output and also incremental learning.

The remainder of the paper is organized as follows. Section II provides an overview of related research efforts available in the literature. In Section III, we introduce and discuss the proposed architecture. Section V evaluates the proposed solution and discusses the obtained results. Finally, in Section VI, we present our conclusions and ideas for future work.

II. Related Work

Existing literature reveals that IDSs that use multiple classifiers usually outperform those based on a single classifier [8], [9]. Furthermore, collaboration among multiple IDSs analyzing different data sources is already being explored [10]. However, to the best of our knowledge, ours is the first proposal that combines both approaches, namely, multiple classifiers and

978-3-903176-23-2

multiple data sources. Therefore, in this section, we discuss separately advances in the usage of multiple classifier systems and collaborative IDS architectures.

A. Multiple Classifier Systems

Sabourin [11] introduces the precursor solutions for the dynamic selection of classifiers. The proposal is based on a classifier ranking method, which evaluates the classifiers' performance through a training database to estimate its overall accuracy and then rank them. A simplified version of this work is proposed in [7], where the classifiers are ranked based on their accuracy in the classification of the known neighboring samples. However, both approaches are prone to fail for unknown samples. Additionally, no decisions criteria are presented to resolve conflict among the classifiers.

In [6], authors present a methodology for estimating the best classifier for a given class, where the competence of each classifier is evaluated according to its performance on specific classes. Thus, the probability of correct classification of an unknown sample is defined from the average of the results given for each evaluated class. However, in real scenarios, there are many attack classes, making this approach costly.

A classifier selection method based on clustering and the weighted average is proposed in [9]. This method consists of clustering known samples and measuring the performance of each classifier for all the formed clusters. Then, the unknown sample is associated to the cluster with the nearest centroid. The classifier with the best performance for that cluster and the classifier with the best performance for the nearest neighboring cluster are selected. Finally, from these selected classifiers, the one with the highest reliability is chosen. Reliability is defined by means of a confusion matrix from known results. However, classification conflicts are not addressed.

Although better than single classifier approaches, the choice of the best classifier remains an open issue, especially on conflicting scenarios. In addition, they do not consider collaboration from multiple sources nor the possibility self-learning.

B. Collaborative Architectures

Although the collaboration between expert systems to analyze multiple and heterogeneous sources is barely investigated in the literature [12], some proposals attempt to consolidate data from different sources for intrusion detection.

In [13], syslog events are correlated with alerts generated by Snort [14]. This correlation is based on the context in which events and alerts are generated — defined by a time window. Therefore, features extracted from both sources allow the detection of more complex attack patterns. However, according to [12], the original architecture structure is prone to Denial-of-Service (DoS) attacks and has performance issues when being executed in a distributed fashion.

A multiple heterogeneous source architecture for intrusion detection is presented in [10]. Network flows, such as HTTP and DNS, are correlated by using their IP addresses. However, a system administrator is required to define rules and make the final decision when multiple correlated alarms are presented.

In [15], a distributed framework is presented to optimize results from homogeneous IDSs. These IDSs are evaluated separately by event injection, where labeled events are used to check their specialty (*i.e.*, ability to accurately detect a certain type of attack). Every IDS node must evaluate its neighbor IDS nodes by sending known samples to check the consistency of the neighbor's output. The neighbor with the best accuracy is chosen as the specialist for the given class. Nevertheless, conflicting scenarios and multiple source are not addressed.

In [16], an architecture for processing heterogeneous alerts is proposed. This architecture has an anomaly detector that works with the Snort IDS. The main purpose of this proposal is to reconfigure Snort to support new signatures. However, those signatures are generated based on manually created rules which often result in false alarms.

An architecture for detecting multiple-step attacks is proposed in [17]. Authors analyze logs in *netflow* and *syslog* formats to identify sequential patterns for the same IP address. This approach aims to detect multistage malicious actions (e.g., correlating authentication errors with failed attempts to access the system's root user). However, monitoring and taking preventive actions lies on the network administrator. Therefore, this approach also presents high dependence on human action for decision making.

Finally, a collaborative IDS [18] deploys IDSs in different parts of the network. IDSs with different specialties or training datasets can consult each other. However, every IDS analyzes the same type of source. Therefore, the results are redundant from a global point of view. Despite the limitations, we consider this work a reference for building cooperative IDSs. In the next section, we improved this network by analyzing multiple and heterogeneous

C. Summary

Although there are efforts focused on employing decision methods for dynamically selecting the most appropriate classifier, those techniques do not leverage different data sources. On the other hand, the integration of heterogeneous sources is also being investigated by the research community. In this trend, however, no efficient methods are presented for the integration of multiple outputs by different classifiers, requiring the human intervention (*e.g.*, network administrator) for decision making.

TABLE I
SUMMARY OF THE CHARACTERISTICS OF SEVERAL RELATED PROPOSALS.

Ref	Souce	Integration	Self-learning	Human Dependency
[16]	Single	No	Yes	High
[15]	Single	Yes	Yes	Low
[18]	Single	Yes	Yes	Low
[13]	Multiple	Yes	No	High
[17]	Multiple	Yes	No	High
[10]	Multiple	No	No	High

Table I summarizes the proposals reviewed in this section in terms of human dependency, cooperation among multiple detectors, self-learning capability and the ability of using

multiple heterogeneous data sources. None of the proposals combines low human dependency, self-learning, and the usage of multiple data sources with the integration of multiple classifiers. Our proposal, detailed in the next section, attempts to fill this void.

III. Proposed Architecture

This section presents the proposed counselors-based IDS architecture. Figure 1 provides an overview of the components, which are explained in the following subsections.

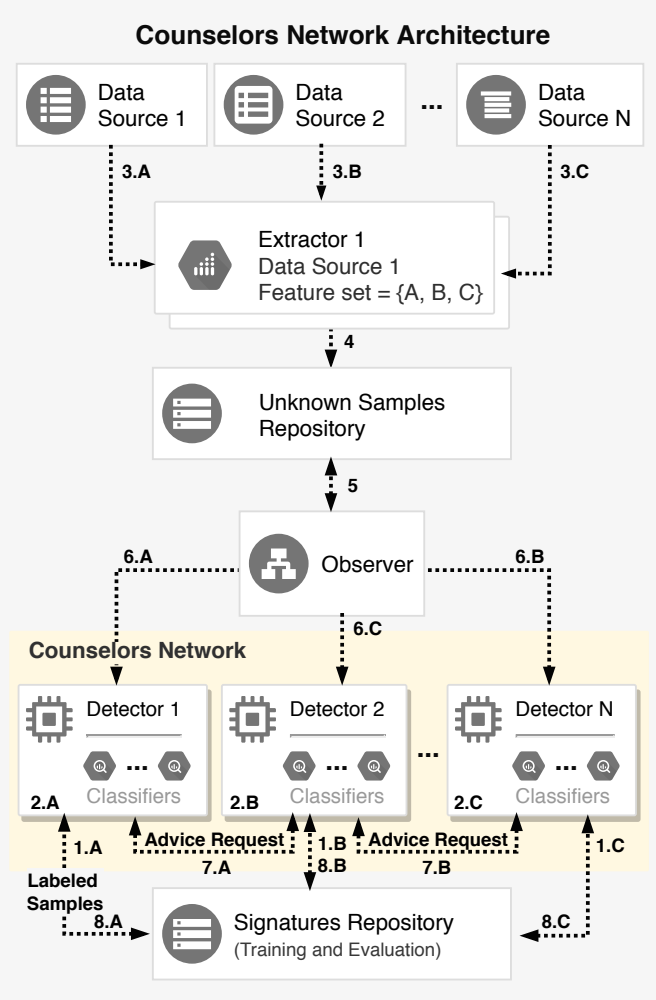

Fig. 1. Architecture Overview.

A. Extractor

The first step in analyzing data is to collect it, such as demonstrated in Figure 1 at 3.A, 3.B, and 3.C. Thus, the *Extractor* component is responsible for reading data from heterogeneous sources and extracting relevant features. The implementation of this component is dependent on the sources that are being monitored. In this paper we consider application logs, network traffic statistics, and connection features.

The *Extractor* then transfers the filtered data to the *Unknown Samples Repository* (step 4 in Figure 1), which stores data samples and provides access to the detection engines that will consume them. In real scenarios, where large volumes of data must be processed, bottlenecks in storage can compromise system performance. Thus, it is important to consider employing tools that provide good performance for retrieving the stored data, such as Firebase Real-time Database [19].

B. Observer

The *Observer* component is based on the publish/subscribe model and serves as a broker for the samples published by Extractors to be delivered to interested subscribers (at step 6.A, 6.B, and 6.C, in Figure 1). *Detectors*, in turn, subscribe to certain topics of interest (*e.g.*, a specific data source). Therefore, any unnecessary processing overhead at the detection component is avoided, because the Observer monitors the *Unknown Samples Repository* (at step 5 in Figure 1), and only forwards those samples of interest for each detector. In a sense, the *Observer*'s role is similar to a load balancer.

C. Detection Engines

The *Detection Engines* contain all the algorithms used for detecting intrusions, such as data mining techniques. These algorithms perform both offline training and evaluation procedures (at steps 2.A, 2.B, and, 2.C) using samples retrieved from a *Signature Repository* (at steps 1.A, 1.B, and, 1.C). Once trained, *Detectors* can perform the online analysis of samples (*i.e.*, retrieved from *Unknown Samples Repository*).

The process performed by the *Detection Engines* is organized in three steps: sample consumption, classifier selection, and data classification. A fourth step is the counsel exchanging, performed only for conflicting cases (steps 7.A and 7.B).

1) Data Consumption: The intrusion detection process occurs in real-time, therefore, it assumes that the training and evaluation phases has already been performed previously (offline). Therefore, data consumption occurs in two different moments: learning/training and intrusion detection phases. First, signatures (*i.e.*, an array of features with regard to the monitored data source) are used to build the classifiers models. Then, during the intrusion detection process, every notification from the Observer triggers a new analysis task that requires consuming the respective new sample. Feature filtering may be performed at this time to retrieve only the relevant resources from the new sample, avoiding network overhead [20].

It is important to note that the offline model can be re-built in parallel to online intrusion detection to periodically update the classifier models with new training data without harming the detection time performance.

2) Classifier Selection: We assume that each detector node employs multiple classifiers in the classification of unknown samples. As related in Section II, there are already different existing methods for combined results from multiple classifiers into a final result. These methods aim to maximize the accuracy of the intrusion detection process, however using inaccurate classifiers can harm the results. Additionally, computational resources may be unnecessarily wasted for additional classifiers without accuracy improving. Therefore,

instead of applying all classifiers to each new sample, we rely on a clustering method during the evaluation phase to choose the most specialized classifiers according to their historical performance for similar samples. During the evaluation phase, every classifier is evaluated for each formed cluster. Then, the most accurate classifiers are selected to the next phase. This choice requires a selection criteria and a comparative metric. In this work, we use the accuracy as a metric to define the potential of a classifier to take a correct decision. Classifiers accuracy can be computed at evaluation time according to Equation 1, where true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) are considered.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

Since multiple classifiers can present good accuracy during the evaluation, we select both the most accurate classifier and those that present it accuracy slightly worst by adopting a maximum threshold. For example, assuming the threshold of 0.1% and the best classifier accuracy is 90%, classifiers with at least 89.9% of accuracy are also selected.

3) Data Classification: In this phase the euclidian distance between the cluster centroids and the unknown sample are computed to verify which cluster this sample has more similarity. Then, only the selected classifiers for this cluster are used to classify the unknown sample. Thus, computational resources are used only for those classifiers that show potential to improve the detection accuracy. Note that evaluation phase occurs offline, but classification phase is online. Thus, no overhead is introduced to real-time intrusion detection.

Due our selection criteria allows multiple classifiers to be classified, if there is no disagreement between their outputs, the confidence in the final result is enhanced. However, it is important to note that as more classifiers are selected for a given cluster, greater is the chance of conflicting decisions. In this case, it's fundamental to define which one is presenting true and reliable results. Even if one of them presents a slightly superior accuracy, the final decision reliability is questionable. The same problem also occurs in the opposite situation, when none of the classifiers has sufficient historical accuracy to make a reliable decision. In both cases any decision can be reckless. Therefore, it is important to define a threshold considering the trade-off between the decision reliability from fewer classifiers and the potential for conflicting cases due to classifiers overlapping.

Our proposal addresses these issues. We introduce a cooperative counselors network composed of detectors that can overcome conflicts by relying on advice from detectors with different expertise and data sources. Further details regard to the proposed counselors network are provided in Section III-D.

D. Counselors Network

A *Counselors Network* is a concept introduced in this paper to deal with specific scenarios where the reliability of the detector's final decision is not ensured by itself. Usually, this can be a consequence of two situations: (i) conflicting outputs from accurate classifiers or (ii) questionable outputs from inaccurate classifiers.

In particular, the first situation can be addressed by requesting advice from a homogeneous counselor detector (*i.e.*, trained with a different knowledge dataset, but from the same source). On the other hand, the second situation may occur when detectors have a limited vision of the network or applications that run on each host, preventing them from accurately detecting attacks. That lack of relevant information to detect specific attacks cab be overcome by requesting advice to expert counselors that analyze complementary data sources.

It is important to note that advice exchanged between detectors that rely on heterogeneous data sources should have the same detecting goal (*i.e.*, detecting the same attack classes). For example, host-based and network-based detectors can cooperate with each other to detect DoS attacks.

Another important factor to be considered in the advice exchange is that the counselors reliability can be established by its historical accuracy for detecting similar samples, in the same way that is performed at classifier selection step. Therefore, a response to advice must include both the counselors result and its historical accuracy. Therefore, the detector that asks for advice can ignore it if the counselor's historical accuracy for similar samples is considered low. In this case, other counselors can be consulted, if available. Otherwise, secondary metrics may be used, such as false alarm rate.

It is important to emphasize that the proposed method of conflict resolution aims at mitigating the need for human intervention, but another important feature is the ability to recognize situations in which current conditions do not allow effective detection. The causes for this limitation may include factors such as insufficient training data or limited vision to application or network information. In any case, those factors are beyond the scope of this proposal.

Whenever a detector receives valid advice (*i.e.*, with acceptable accuracy, according to the established parameters) it generates a new signature that is added to the known sample repository (as shown in Figure 1 at steps 8.A, 8.B, and 8.C). That allows the proposed architecture to improve classifiers' models by means of self-learning, because the new signatures may be used for retraining the classifiers.

IV. Implementation

In order to deploy an IDS, several questions must be considered, such as, choosing the appropriate intrusion detection technique (*e.g.*, data classifiers or cluster algorithms), available data-source, and which features to selected from those data-source. Our proposed architecture is flexible into respect to these project decisions. Bellow, we present a prototype that enables the evaluation of the architecture.

A. Feature Extraction

The Extractor element must be prepared to extract features from a specific data sources in which it is desired to perform intrusion detection. Therefore, due to the multiple existing data sources for intrusion detection, the implementation described

in this section abstracts the data extraction procedure. Thus, from this point it is assumed that the relevant features have been extracted. In particular, CICIDS2017 dataset is composed of network traffic features extracted by CICFlowMeter [21].

B. Model Building

The model building procedure can be split into two essential phases: classifiers training and classifier evaluation. Since both phases should be fed with a different signature dataset (*i.e.*, samples with known labels), we split the entire signature dataset — retrieved at steps 1.A, 1.B, and 1.C of Figure 1 — into a *training dataset* and an *evalution dataset*. Each one with the same number of samples and the same proportion for each existing class. These settings attempt to find a trade-off between the classifiers training level and the confidence in the evaluation phase to estimate classifiers accuracy for future unknown samples. Algorithm 1 details each phase.

```
input : k // number of clusters
        evalData // evaluation signatures
        trainData // training signatures
        α // selection threshold
output: Selected classifiers for each formed cluster
// Training Phase
1  foreach c ← classifiers do
2  |  c ← Build(trainData);
3  end
// Evaluation Phase
4  clusters [k] ← KMeans(k, evalData);
5  foreach cluster ← clusters do
6  |  clusterInstances ← GetInstances(cluster)
   |   foreach c ← classifiers do
7  |  |  foreach i ← clusterInstances do
8  |  |  |  output ← Classify(i);
9  |  |  |  c ← UpdateAccuracy(output);
10 |  |  end
11 |  end
12 |  foreach c ← classifiers do
   |  |  // Select if is accurate
13 |  |  selectedClassfiers ← Select(c, α) ;
14 |  end
15 |  cluster ← selectedClassfiers
16 end
```

Algorithm 1: Offline phases: training and evaluation.

At training phase (Algorithm 1, lines 1—3), the *training dataset* is used to train the following supervised machine learning classifiers: NBTree, Naive Bayes, Random Tree, Rep Tree, and Random Forest. Once trained, these classifiers are evaluated using the *evaluation dataset*. This dataset is first clustered through K-Means algorithm (Algorithm 1, line 4), then each classifier is evaluated for each formed cluster (Algorithm 1, lines 5—11). Since classifiers can perform better for different class patterns, we set the number of clusters (K) based on the number of existing classes from evaluation and training datasets, attempting to form clusters with similar attack types.

After the evaluation, the best classifiers for each cluster are choosen (Algorithm 1, lines 12—15). The classifier selection considers both the most accurate classifier and others with slightly less accuracy. The "slightly" is based on a threshold α, set to 0.1% in our prototype.

C. Detection Phase

Contrary to previous steps, the Detection Phase runs in real time. Therefore, classifiers are assumed to be already trained and selected for each cluster. This phase focus on clustering the unknown instance, by computing the euclidean distance from the sample to each cluster — also previously formed, at evaluation step. Once the selected classifiers are retrieved, they classify the unknown instance (Algorithm 2, lines 1—5).

```
input : centroids // K clusters centroids
        instance // unknown instance
output: Classification of the unknown instance
1  centroid ← KMeans(instance) ;
2  classifiers ← selClassfiers(centroid);
   // Detection Phase
3  foreach classifier ← classifiers do
4  |  Class ← Classify(instance);
5  end
   // Counselors Phase
6  if HaveConflict(classifiers) then
7  |  timestamp ← GetTimeStamp(instance);
8  |  Class ← RequestAdvice(timestamp);
9  end
```

Algorithm 2: Online phase: real-time intrusion detection.

If every selected classifier yields the same result, the detection phase is finished and the result is output. However, a conflicting scenario may occur, defined as satisfying all the following constraints simultaneously: (i) the unknown sample is clustered into a cluster with at least two classifiers in the same Detector elected during its previous evaluation phase, and (ii) there is at least one disagreement between these classifiers. When these conditions are satisfied, a conflict occurs inside the detector, therefore it is necessary to request an advice to an counselor IDS node. Figure 2 details the messages involved involved in getting/providing advice.

The process begins when a detector which found a decision conflict sends an *Advice Request* to the counselors network (*i.e.*, Detector 1 on Figure 2). Then, the counselor which receives the request sends a reply based in it own already analyzed samples. Notice that different data sources may operate under different time frames (*e.g.*, two samples indicating the same attack may have been collected at slightly different times). Therefore, whenever an advice is received by a detector, it must ensure that the information used to base that advice is temporally consistent with the sample being classified. To that end, we consider a maximum two-second

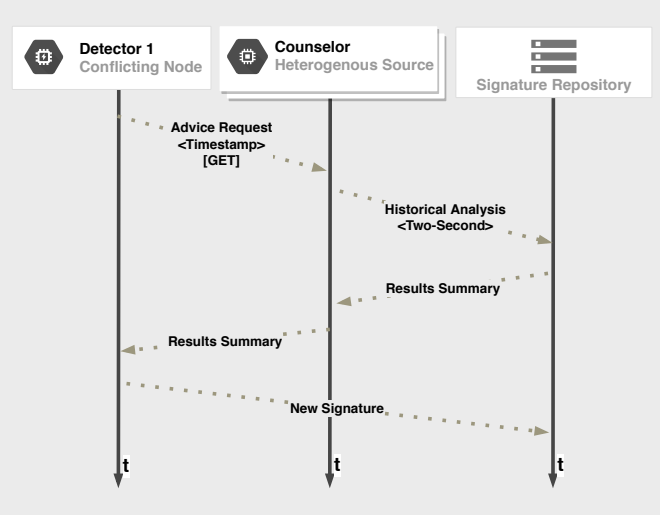

Fig. 2. Advice exchanging flow-model.

time delay between the timestamps (*i.e.*, the past two seconds from the highest timestamp) associated with the advice and the sample being classified. This time-window is based on [22]. Finally, detectors can output the result and retro-feed its signatures dataset without human intervention — at steps 8.A, 8.B, and 8.C in Figure 1. Thus, self-learning can be done.

V. Evaluation

In order to evaluate the proposed architecture, we present two scenarios according to the type of data analyzed by detectors. First, we assess the collaboration between detectors analyzing multiple and heterogeneous data sources. This analysis focuses on detailing the counselors mechanism by looking for specific conflict cases. Then, we appraise the proposed solution into a scenario where multiples homogeneous detectors compose the counselors network — each one with a different knowledge database but processing the same data source. In this second scenario, we focus on recent threats and into whole accuracy based on a larger-scale analysis involving more than one million samples (*i.e.*, 1,116,750 samples).

A. Scenario 1: Heterogeneous Cooperative Network

To evaluate the proposed architecture in a scenario with multiple and heterogeneous sources, we choose the NSL-KDD [23] dataset — an improved version of KDD Cup 99. Although this is not a recent data set, it serves as a baseline since it is widely used in the literature to evaluate recent proposals. From this dataset, three detectors of heterogeneous data sources were configured. The data sources correspond to individual TCP connections features, traffic statistics and application logs, and were used to generate detectors 1, 2, and 3, respectively. To demonstrate the specific cases where our solution can solve conflicts and improve the final output, we selected a reduced set containing 1,000 samples from this dataset.

When analyzing the samples, Detector 1 — which is based on individual TCP connections features — found multiple decision conflicts. One particular case occurred into cluster 3, were two classifiers algorithms (*i.e.*, IBk and NBtree) presented the same historical accuracy of 95.11%. In most cases, these well-rated classifiers present the same results, therefore, improving the confidence in the classification process strengthens. Therefore, according to our conflict definition, these cases are not considered as a conflict. However, during the analysis of a teardrop attack sample — a subclass of Denial of Service (DoS) —, at detection phase, the classifiers presented different results, thus constituting a conflict. While IBk classifier classified this sample as an attack, NBTree classifier identified this sample as being corresponding to a normal behavior, as shown in Table II.

TABLE II
Conflicting situation in Detector 1.

Classifier	Accuracy	VP	FN	VP	VN	Output
IBk	**95,12195%**	**28**	**2**	**0**	**11**	**Attack**
J48	85,36585%	24	6	0	11	-
NBTree	**95,12195%**	**28**	**2**	**0**	**11**	**Normal**
Naive Bayes	60,97561%	15	15	1	10	-
ADTree	85,36558%	24	6	0	11	-
ADABostM1	82,92683%	23	7	0	11	-
RepTree	48,78049%	9	21	0	11	-
KStar	92,68293%	27	3	0	11	-
Random Forest	75,60975%	20	10	0	11	-

Therefore, since Detector 1 cannot make a reliable decision due to the disagreement of its selected classifiers, it must request advice to the counselor's network. In this scenario, Detector 2 — that analyzes traffic statistics — acts as a counselor to Detector 1. Since it is assumed that Detector 2 runs in parallel with Detector 1, the advice response is based on the past two-second analysis. Then, upon receiving the request, Detector 2 checks its already processed results based on the request timestamp. Since Detector 2 unambiguously detected the occurrence of a teardrop attack, as shown in Table III, the result of this request confirms the IBk output from Detector 1. This confirmation was obtained with J48 classification, which historically has 100% of accuracy for this sample class, according to this data source. One of the main reasons each detector is more expert in a different type of attack is due to the specific features set analyzed by them [24].

Therefore, the final result from Detector 1 is based on the advice coming from Detector 2: the unknown sample is classified as an attack. In addition, the signature database is updated with the new sample. From this point on, the IBk classifier will be selected as the best classifier for this cluster, until new information changes the knowledge base. This happens because its updated accuracy will be higher than that of NBTree and others classifiers.

B. Scenario 2: Homogeneous Cooperative Network

In addition to previous analyses, we also evaluate an alternative scenario involving the CICIDS2017 dataset. This is a recent dataset that includes the most common attacks based

TABLE III
ADVICE FROM DETECTOR 2 BASED ON ITS BEST CLASSIFIER.

Classifier	Accuracy	VP	FN	VP	VN	Output
IBk	63,41463%	15	15	0	11	-
J48	**100%**	**30**	**0**	**0**	**11**	**Attack**
NBTree	63,41463%	15	15	0	11	-
Naive Bayes	80,48780%	30	0	8	3	-
ADTree	63,41463%	15	15	0	11	-
ADABostM1	63,41463%	15	15	0	11	-
RepTree	63,41463%	15	15	0	11	-
KStar	63,41463%	15	15	0	11	-
Random Forest	63,41463%	15	15	0	11	-

on the 2016 McAfee report [1], such as Brute force, DoS and DDoS, Web-based, Infiltration, Heart-bleed, Bot, and Scan.

In particular, we focus on analyzing samples containing DoS, DDoS, and PortScan attacks variations. Whereas Detector 1 is trained with 20% of total samples containing four DoS, such as slow loris, slow HTTP test, DoS Hulk, and DoS GoldenEye, Detector 2 is trained with 20% of DDoS (Loit and Ares botnets) and PortScan signatures. Thus, each detector has a different knowledge dataset. As explained in Section IV, these signatures are split into two parts with the same size. Thus, the training dataset is 10% of total available samples.

However, as our goal is to evaluate the cooperation between these detectors, both analyze all classes. Then, we aggregate the remaining 80% samples from each class and use them to test the detectors. Therefore, even if each detector is trained with specific attack classes, both are tested with all classes.

We compare our results in terms of accuracy (Figure 3) and detection rate (Figure 4) with six different approaches. The *Best Local* approach selects only the classifier which demonstrates the best accuracy during the evaluation phase to be used on real-time detection. This approach performed worse than average among all classifiers from Detector 1.

Majority and *Weighted Voting* approaches consider the output of all classifiers to take a final decision. However, whereas *Weighted* approach considers the past accuracy of each classifier to define its weight during the voting, Majority Voting only considers the results from the classifiers that yields the same result as the majority. Both approaches perform better than the best single-classifier (*i.e.*, Random Forest with 81.96% into Detector 1 and 74.57% into Detector 2). The *Majority*

[1] www.mcafee.com/2016ThreatsPredictions

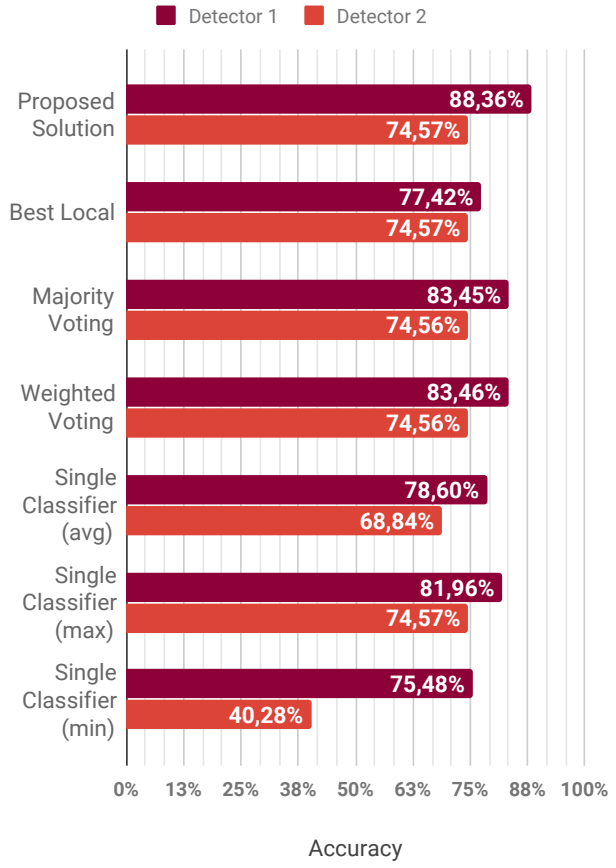

Fig. 3. Accuracy for each approach.

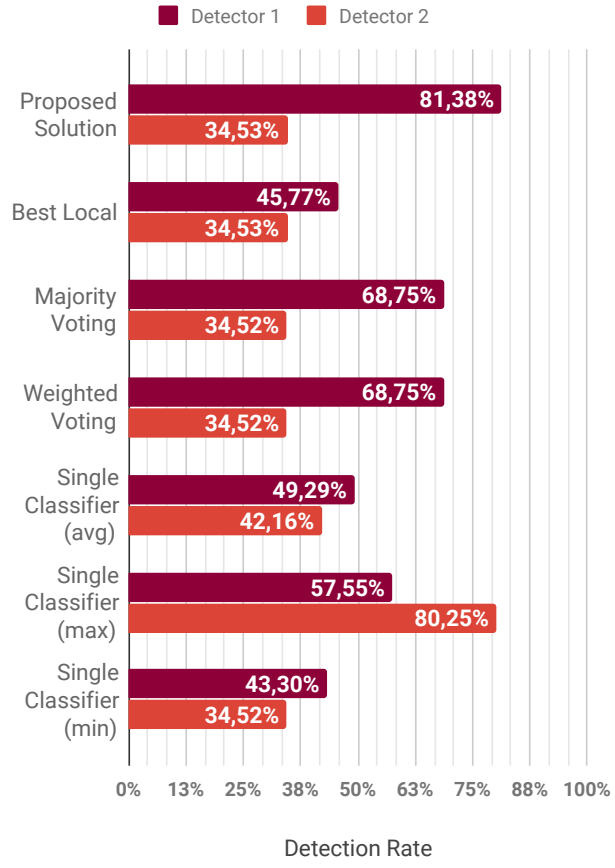

Fig. 4. Detection rate for each approach.

Voting performance is lower than *Weighted Voting*: it presented conflicting cases that can not be solved (*i.e.*, 50% of classifiers present attack outputs and other 50% present normal outputs).

Regarding Detector 2, it presented a poor performance for all methods and classifiers. The main reason may be related to potentially insufficient training data. It is important to note that Naive Bayes presents the best detection rate (81.38%), but was the worst accuracy (40.28%). This is due to its poor performance in detecting normal samples. Therefore, accuracy can be considered more representative than the detection rate.

Finally, our approach can overcome every other, standing out especially in the Detector 1, where the accuracy of 88.36% was achieved. The same can be observed in terms of the Detection rate, in Figure 4, where 81.38% of attacks were detected successfully. In particular, cluster number 2 at Detector 1 resolved 1.485 conflicts (2.36% of total 61.339 clustered samples), thus improving its accuracy. In Detector 2 scenario, our proposal and Best Local achieves the best accuracy — same as the best single classifier accuracy. This low improvement is due the low number of conflicting cases (*i.e.*, only 417 conflicts among all clusters). The detection rate is poor for all approaches except for Naive Bayes. In contrast, NaiveBayes presented many false positives, which makes its accuracy worse than the other classifiers.

VI. Conclusion

The heterogeneity of existing attacks on real computer systems and networks is a very challenging issue that should be carefully addressed. Data classification algorithms can be helpful in dealing with this. The literature demonstrates that the use of multiple classifier algorithms is more efficient than a single algorithm. However, many existing works are only concerned with choosing the best algorithm to correctly classify samples from a single data source, consequently presenting errors or conflicts. On the other hand, works that consider multiple data sources do not present mechanisms to deal with conflict situations when integrating different results.

This work proposed a new architecture that relies on the dynamic choice between multiple classifiers supported by advice exchange from detectors that can analyze multiple and heterogeneous data sources, thus improving the trust in the final decision. Our experiments demonstrate that dynamic classifier selection can be enhanced when multiple sources are analyzed, and conflicts are solved without human intervention.

In future work, we plan to extend our architecture by employing big data technologies to deal with scenarios of a huge amount of traffic. In addition, we will explore other classifiers and cluster algorithms, as well the detectors parameters.

References

[1] R. Singh, H. Kumar, and R. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8609–8624, 2015.

[2] E. W. Ngai, L. Xiu, and D. C. Chau, "Application of data mining techniques in customer relationship management: A literature review and classification," *Expert systems with applications*, vol. 36, no. 2, pp. 2592–2602, 2009.

[3] T. M. Mitchell, "Artificial neural networks," *Machine learning*, vol. 45, pp. 81–127, 1997.

[4] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.

[5] A. S. Britto, R. Sabourin, and L. E. Oliveira, "Dynamic selection of classifiers—a comprehensive review," *Pattern Recognition*, vol. 47, no. 11, pp. 3665–3680, 2014.

[6] T. Woloszynski and M. Kurzynski, "A measure of competence based on randomized reference classifier for dynamic ensemble selection," in *20th International Conference on Pattern Recognition (ICPR)*. IEEE, 2010, pp. 4194–4197.

[7] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[8] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Computing and Applications*, pp. 1–8, 2016.

[9] A. Mi and H. Sima, "Classifier selection method based on clustering and weighted mean," *Journal of Intelligent & Fuzzy Systems*, vol. 31, no. 4, pp. 2335–2340, 2016.

[10] S. Marchal, X. Jiang, R. State, and T. Engel, "A big data architecture for large scale security monitoring," in *International congress on Big data (BigData Congress)*. IEEE, 2014, pp. 56–63.

[11] M. Sabourin, A. Mitiche, D. Thomas, and G. Nagy, "Classifier combination for hand-printed digit recognition," in *Proceedings of the Second International Conference on Document Analysis and Recognition*. IEEE, 1993, pp. 163–166.

[12] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: a survey," *Journal of Big Data*, vol. 2, no. 1, p. 3, 2015.

[13] A. K. Ganame, J. Bourgeois, R. Bidou, and F. Spies, "A global security architecture for intrusion detection on computer networks," *Computers & Security*, vol. 27, no. 1, pp. 30–47, 2008.

[14] M. Roesch, "Snort: Lightweight intrusion detection for networks." in *Proceedings of LISA '99: 13th Systems Administration Conference*, vol. 99, no. 1. USENIX, 1999, pp. 229–238.

[15] K. Bartos and M. Rehak, "Self-organized mechanism for distributed setup of multiple heterogeneous intrusion detection systems," in *Sixth international conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*. IEEE, 2012, pp. 31–38.

[16] K. Hwang, H. Liu, and Y. Chen, "Cooperative anomaly and intrusion detection for alert correlation in networked computing systems," *IEEE Transaction on Dependable and Secure Computing*, 2004.

[17] C.-M. Chen, G.-H. Lai, and P.-Y. Young, "Defense joint attacks based on stochastic discrete sequence anomaly detection," in *Asia Joint Conference on Information Security (AsiaJCIS)*. IEEE, 2016, pp. 74–79.

[18] C. J. Fung and Q. Zhu, "Facid: A trust-based collaborative decision framework for intrusion detection networks," *Ad Hoc Networks*, vol. 53, pp. 17–31, 2016.

[19] Firebase realtime database. [Online]. Available: https://firebase.google.com/docs/database/

[20] A. A. Olusola, A. S. Oladele, and D. O. Abosede, "Analysis of KDD'99 intrusion detection dataset for selection of relevance features," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, 2010, pp. 20–22.

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116.

[22] (1999) KDD-CUP-99 Task Description. [Online]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/kddcup99-mld/task.html

[23] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. IEEE, 2009, pp. 1–6.

[24] S. E. Quincozes, A. Copetti, and J. F. Kazienko, "Avaliação de Conjuntos de Atributos para a Detecção de Ataques de Personificação na Internet das Coisas," in *Brazilian Symposium on Computer Engineering (SBESC)*. SBC, 2018, pp. 1–8.