Community-based Collaborative Intrusion Detection

Carlos Garcia Cordero[1], Emmanouil Vasilomanolakis[1], Max Mühlhäuser[1], and Mathias Fischer[2]

[1] Telecooperation Group,
Technische Universität Darmstadt / CASED
Darmstadt, Germany
Email: [carlos.garcia,manolis,max.muehlhaeuser]@cased.de
[2] Networking and Security Group,
International Computer Science Institute,
Berkeley, USA
Email: mfischer@icsi.berkeley.edu

Abstract. The IT infrastructure of today needs to be ready to defend against massive cyber-attacks which often originate from distributed attackers such as Botnets. Most Intrusion Detection Systems (IDSs), nonetheless, are still working in isolation and cannot effectively detect distributed attacks. Collaborative IDSs (CIDSs) have been proposed as a collaborative defense against the ever more sophisticated distributed attacks. However, collaboration by exchanging suspicious alarms among all interconnected sensors in CIDSs does not scale with the size of the IT infrastructure; hence, detection performance and communication overhead, required for collaboration, must be traded off. We propose to partition the set of considered sensors into subsets, or *communities*, as a lever for this trade off. The novelty of our approach is the application of ensemble based learning, a machine learning paradigm suitable for distributed intrusion detection. In our approach, community members exchange data features used to train models of normality, not bare alarms, thereby further reducing the communication overhead of our approach. Our experiments show that we can achieve detection rates close to those based on global information exchange with smaller subsets of collaborating sensors.

1 Introduction

The continuous growth and sophistication of cyber-attacks poses a serious threat to networked infrastructure. To contest this, IDSs monitor a host or a network for signs of intrusions or security policy violations. Detecting intrusions within IDSs is typically performed through *misuse analysis* or *anomaly detection*. Misuse analysis assumes the availability of fingerprints of previously seen attacks, so that they can be detected upon their next occurrence. Anomaly detection establishes a model of normal system behavior. Each deviation from this model is an anomaly and thus a potential attack. Models of normal behavior can be manually provided or automatically learned [1]. IDSs usually operate isolated from each other. There is no communication or interaction between them and, as a result, isolated IDSs fail to detect distributed attacks as different monitoring points are not exchanging information.

To create a holistic view of the monitored network, collaboration between IDSs is required, which has led to the development of CIDSs [2]. These systems consist of *sensors* and one or several *analysis units* that attempt to detect distributed attacks collaboratively. CIDSs can be either centralized or distributed. In centralized CIDSs, sensors send their monitored information directly to a central analysis unit, while in distributed CIDSs sensors exchange data among each other and do a distributed analysis. Distributed CIDSs provide better scalability than centralized CIDSs while reducing the communication overhead. However, compared to centralized systems, this usually comes at the cost of a decreased detection precision, i.e., the ratio between true alarms (or *true positives*) and the total number of alarms (*true positives* + *false positives*), as there is no component in the system with global information.

CIDS exchange data either on the alarm or detection level. Information exchange on the alarm level, e.g., [3], encompasses the exchange of intrusion alarms for post processing. The main goal of this type of collaboration is to ease the manual task of analyzing all issued alarms by creating summaries and to discover related attacks. In contrast, collaboration on the detection level encompasses the exchange of monitored information (or data features) to collaboratively create or improve mathematical models. These mathematical models aim to improve the detection accuracy and, thus, lower the number of False Alarms (FAs). However, to the best of our knowledge, there is no CIDS that currently supports data exchange on the detection level [2]. We recognize that on the detection level, however, ensemble learning can be applied as a distributed machine learning method [4]. Furthermore, ensemble learning has been demonstrated to be effective in the generic setting of improving anomaly detection [5].

In this paper, we propose a CIDS concept for learning models of normality to detect network anomalies. Our focus is not to introduce a full-fledged CIDS, but rather to demonstrate the applicability of ensemble learning on intrusion detection in a distributed and collaborative setting. We propose the establishment of communities of sensors that exchange data to build anomaly detection models and detect anomalies collaboratively. A sensor is able to participate in multiple communities concurrently, which enables the applicability of ensemble learning techniques. Each sensor shares data with its communities, so that subsets of the entire dataset are created. This allows each community to create an alternative hypothesis from each subset. Each hypothesis represents a particular interpretation of normal behavior and all hypotheses can be used together to determine whether arbitrary network traffic is normal or not. We evaluate our novel CIDS concept with a modified version of the DARPA dataset [6] that reflects a distributed monitoring setting. Our results indicate that a community-based CIDS approach performs better, in terms of detection accuracy and precision, than isolated IDSs in the task of learning models of normality.

The remainder of this paper is organized as follows: Section 2 introduces the related work on anomaly detection and CIDSs. Section 3 presents our community-based CIDS concept. Section 4 evaluates our community concept using anomaly detection. Finally, Section 5 concludes the paper and gives insights into future directions.

2 Related Work

In this section, we give a brief overview of related work for anomaly detection algorithms as well as distributed CIDSs.

2.1 Anomaly Network Intrusion Detection

Discovering anomalies in categorical data is of particular interest to anomaly-based IDSs as they heavily rely on the analysis of categorical attributes [7]. For example, IP addresses are normally represented as categorical rather than numerical attributes. This is an important issue to take into account as not every machine learning technique is able to work well with network data. There are, however, many machine learning algorithms that are well suited for this task.

Rule induction techniques are examples of algorithms suitable for handling categorical attributes. Mahoney and Chan published the Packet Header Anomaly Detector (PHAD) algorithm [8]. It focuses on finding rules describing the normal appearance of the Ethernet, IP, TCP, UDP, and ICMP protocols. Detection of anomalies in this context is limited to packets not adhering to one of the learned protocols. Learning Rules for Anomaly Detection (LERAD) [8], finds rules on its own through a stochastic sampling algorithm. Instead of modeling hand picked rules, LERAD is capable of finding a subset of effective conditional rules that describe normal network data.

Rule learning algorithms, such as LERAD, are prime candidates for building ensembles of learners. An ensemble is a collection of classifiers that come together to classify novel instances as a group. Ensemble learning is comprised of a set of techniques to join the decisions made by different classifiers. The two most common techniques are called Bagging and Boosting [9]. Bagging is the process of sampling, with replacement, instances from a large dataset to create subsets. These subsets are used by many classifiers to learn different models of normality (for anomaly detection). To classify a novel instance, each classifier makes a decision. Multiple techniques can be used to mix all the classification decisions into one final decision. A popular technique is to consider each classifier output as a vote and use the class with the most votes. In this paper, we use a technique where the decision of the classifier with the most confidence in classification is used. LERAD is able to output not only the class, but also the confidence of detection as an *anomaly score*. Therefore, for one particular novel instance, the LERAD classifier with the highest anomaly score is taken as the classification decision.

2.2 Distributed CIDSs

CIDSs can be classified, with respect to their communication architecture, as centralized, hierarchical or distributed [2]. In *centralized CIDSs*, e.g., [10], sensors deliver data to a central analysis unit responsible for performing data analytics. However, centralized CIDSs do not scale with an increasing number of sensors as each additional sensor increases the communication overhead of the central analysis unit. Additionally, the central unit represents a single point of failure. Hierarchical CIDSs employ a hierarchical tree structure of sensors, e.g., [11]. Within this hierarchy and starting from leaf positions, monitored data is correlated, preprocessed, and detection algorithms are employed until the data converges to a central analysis unit at the root of the tree. *Distributed CIDSs* follow a flat P2P architecture and disseminate the functionality of the central analysis unit across multiple sensors. Thus, each sensor also conducts data analysis, so that sensor data is correlated, aggregated, and analyzed in a completely distributed manner. Beside structured CIDS approaches, e.g., [12], several unstructured proposals have been made, e.g., [13, 14]. However, all existing CIDS approaches operate on the alarm level for the exchange of information [2], while our approach of communities establishes collaboration on the detection level.

3 Community-based Collaborative Intrusion Detection

In this section we give insights into our community-based CIDS. We provide a description of our concept followed by a formal model and a discussion on how the parameters of the formal model affect the properties of a CIDS. Subsequently, we describe our community formation algorithms and how the formed communities are used to perform intrusion detection.

3.1 Basic Concept

Sensors are grouped into communities to create samples of the network traffic all sensors are capable of observing. The samples are used to learn models of normality and perform anomaly detection. This idea is inspired by ensemble learning and guarantees the reduction of variance in the process of learning [9]. The overall outcome is an increased detection performance, in contrast to isolated sensors, and the reduction of communication overhead, in contrast to centralized systems.

In each community, one sensor becomes a *community head*. Community heads retrieve monitored data features from all other sensors in their community and perform intrusion detection. Upon detecting attacks, community heads forward alarms to a central administration interface where further correlation may take place. Selecting community heads can be done either stochastically or coupled to specific sensor properties such as their computational capabilities.

This paper focuses on the detection accuracy and precision a distributed CIDS can achieve. We leave out the practical realization of distributed community formation. However, sensors could be grouped together into a P2P network using Distributed Hash Tables (DHTs) or P2P-based gossiping techniques [15]. Afterwards, techniques like flooding can be applied on top of the overlay to establish communities in a distributed way.

3.2 Formal Model

Our community-based CIDS overlay can be modeled as a graph $G = (V, E)$ where the nodes V represent computer systems capable of communicating between each other through an overlay communication links E that exist between them. Let $S \subset V$ be the set of intrusion detection sensors capable of collaborating among each other to detect attacks. Additionally, let $u \in V$ be a central administration interface responsible for collecting the alarms issued by all IDSs $s \in S$ and for generating intrusion reports. A community is a subset $\mathcal{C} \subseteq S$ of sensors, with $n_c = |\mathcal{C}|$ members. The set of all communities is $\mathbb{C}$, and the total number of communities is $n_t = |\mathbb{C}|$. Each community $\mathcal{C}$ has one sensor $s^\star_\mathcal{C} \in \mathcal{C}$ chosen as the community head; responsible for performing data analysis and intrusion detection. Every other member $s \in \mathcal{C}$ is connected by an edge $e = (s, s^\star_\mathcal{C}) \in E$ to $s^\star_\mathcal{C}$. Each sensor s is responsible for sending all features extracted from the data they collect to $\{s^\star_\mathcal{C} | \forall \mathcal{C} \in \mathbb{C} : s \in \mathcal{C}\}$, i.e., all other community heads they are connected to. The community heads of all communities are summarized in the set $\mathcal{S}^\star = \bigcup_{\mathcal{C} \in \mathbb{C}} s^\star_\mathcal{C}$. Each sensor $s \in \mathcal{S}$ may be repeated up to n_s times between different communities.

Fig. 1 shows three different parametrization scenarios. The parameters specify how sensors s and community heads $s^\star_\mathcal{C}$ are grouped together. In Scenario 1, two communities are shown ($n_t = 2$). These communities have four sensors each ($n_c = 4$) and each sensor is allowed to be used only once ($n_s = 1$). Scenario 2 depicts three communities ($n_t = 3$), each having three members ($n_c = 3$), where the sensors are allowed to be repeated at most twice ($n_s = 2$). Lastly, Scenario 3 shows four communities ($n_t = 4$) with two members each ($n_c = 2$) where sensors cannot be repeated more than once ($n_s = 1$).

3.3 Parameters for Building Communities

When doing collaborative intrusion detection with communities, we recognize three dimensions that influence accuracy, scalability and communication overhead. First, we discuss the influence of the size of communities n_c and second, the number of communities n_t. These two parameters allow to model a centralized CIDS, a fully distributed CIDS, or communities. Third, we discuss the impact of the number of times n_s a single sensor can be part of different communities.

Number of Sensors per Community (n_c) The community size n_c significantly influences the detection accuracy. When $n_c = |\mathcal{S}|$, there is one community with all sensors. The sensor head $s^*_\mathcal{C}$ of this single community observes all data

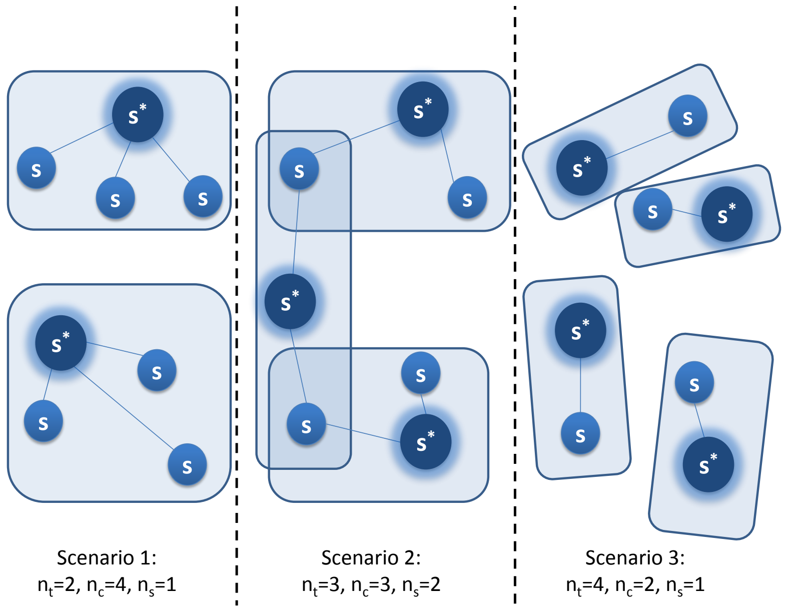


Fig. 1. Two communities (left), three communities (center), and four communities (right), with sensors s and community heads $s^\star$.

in the network and, thus, has full knowledge. This is equivalent to a centralized system that can access all data from one single location. In contrast, when $n_c = 1$, the scenario reflects $|\mathcal{S}|$ isolated sensors learning without any data being shared and no collaboration involved. In this scenario, each community has one sensor that must also be the community head. The size of n_c is bounded by $1 \leq n_c \leq |S|$.

The communication overhead affected by n_c can be expressed as the edges connecting the sensors $s \in \mathcal{S}$ to the community heads $s^\star \in \mathcal{S}^\star$; being inversely proportional to n_c. This overhead is calculated as $|S| - \frac{|S|}{n_c}$ and represents the number of edges required to interconnect all sensors to their respective community heads. Furthermore, with a small n_c, the system as a whole becomes more scalable as communities become responsible for analyzing less data. By increasing n_c, more information becomes available to each community head and a more accurate model can be derived; however, the communities become less scalable as more computational power and memory is required from every community head.

Number of Communities ($\boldsymbol{n_t}$) The second parameter that has an influence on the detection accuracy and precision is the total number of communities n_t. When $n_t = 1$, only one community is established. This is equivalent to $n_c = |\mathcal{S}|$. On the other hand, when $n_t = |\mathcal{S}|$ and $n_s = 1$, all sensors are their own community and no collaboration is involved. This is analogous to the scenario where $n_c = 1$. This shows that both n_t and n_c are inversely related to each other. The number of communities n_t is bounded according to $1 \leq n_t \leq |\mathcal{S}|$.

The parameter n_t affects scalability only in combination with n_c. Having a high number of communities does not imply anything unless n_c is taken into account. The main scalability issue in any distributed environment is the amount of data that needs to be collected and processed. For instance, a large n_t and low n_c implies that there are many communities processing small amounts of data.

Sensor Repetitions in Multiple Communities ($\boldsymbol{n_s}$) We define n_s as the upper bound of the total number of times a sensor can be repeated in different communities. This parameter leverages the impact one specific sensor can have when communities are established stochastically. It is bounded according to $1 \leq n_s \leq n_t$. A sensor cannot be repeated within a community; otherwise, it would introduce bias because of the redundant data being shared.

As this parameter increases, more data is allowed to be repeated among many communities. The availability of all data can be augmented by increasing n_s. However, as this parameter increases, the communication overhead increases as well because sensors must transmit the same information to multiple community heads. The parameter n_s also directly affects the size of each community. As n_s increases, the number of sensors $|\mathcal{C}|$ of each community is increased on average. More members equates to more communication overhead.

3.4 Community Formation

The construction of communities demands criteria for coupling together the set of sensors S into communities $\mathcal{C} \in \mathbb{C}$. The coupling depends on parameters that affect how these are formed, i.e., the community size n_c, the total number of communities n_t, and the maximum sensor repetitions within different communities n_s. The remainder of this section contains a detailed discussion of coupling criteria and the algorithms that implement these criteria.

Coupling Criteria One important design question of our CIDS concept is how to assign sensors to communities, or, more precisely, how the data of all sensors is distributed for analysis. We base our ideas on the bagging ensemble technique. The bagging technique trains a classifier multiple times using different subsets of a dataset. Bagging reduces the variance of the detection accuracy [9]: it reduces the disagreement that might exist when communities are trained on different subsets of a dataset. To create different subsets of the data, data records are sampled with replacement from the entire dataset. To make a decision, every learner classifies the training dataset independently and a combination of all decisions is used to classify each individual training data.

Our proposed community-based CIDS behaves like an ensemble of learners. Each community $\mathcal{C} \in \mathbb{C}$ is a classifier that learns with the data supplied by its members $s \in \mathcal{C}$. Sensors can appear in different communities, which is analogous to sampling batches of data observed by different sensors with replacement. The community size n_c specifies how much will be sampled. The number of communities n_t specifies how many classifiers will be built. Bagging does not usually limit the sampling in any way, we introduce n_s, however, to limit the bias one single community may have in the whole system.

Ensemble methods traditionally split samples of the data randomly (with replacement) among the set of available learners. This is the motivation behind our stochastic creation of communities. We do recognize that in the context of network data more intelligent decisions can be used to split the data. For instance, network traffic can be split according to common network services, IP addresses or other network-related criteria. In this paper, we focus on stochastic community creation and leave other alternatives as future work. We are trying to demonstrate how ensemble methods are able to perform well in the task of anomaly detection when coupling criteria are as general as possible.

Community Construction Algorithms Multiple strategies can be used to form communities by varying the parameters n_t, n_c and n_s. Each parameter can be fixed to a specific value for all communities to share or vary for each individual community. Because of this, we propose two different algorithms to build communities. Algorithm 1 fixes n_c to a particular value such that all communities exhibit the same size. The other two parameters, n_t and n_s, are left to vary for each community. In contrast, Algorithm 2 fixes the parameters n_s and tries to fix n_t whenever it is possible, while leaving n_c to vary for each community.

Algorithm 1: comm$_1$($\mathcal{S}, n_c$)

```
1  $\mathbb{C} \leftarrow \{\emptyset\}$, $T \leftarrow \{\emptyset\}$
2  for $s \in S$ do
3      if $s \notin T$ then
4          $C \leftarrow \{s\}$
5          $T \leftarrow T \cup \{s\}$
6          for $|\mathcal{C}| \leq n_c$ do
7              $s \leftarrow \text{rand}(S - \mathcal{C})$
8              $\mathcal{C} \leftarrow \mathcal{C} \cup \{s\}$
9              $T \leftarrow T \cup \{s\}$
10         $\mathbb{C} \leftarrow \mathbb{C} \cup \{\mathcal{C}\}$
11     return $\mathbb{C}$
```

Given all sensors S and n_c as input, Algorithm 1 outputs a set of communities $\mathbb{C}$. This algorithm consists of two parts: In its first part (lines 2 - 5), the algorithm selects an initial sensor, not belonging to any other community, to start a new community. The list T is used to track sensors that already belong to a community. This restriction ensures that all sensors appear at least once among all communities while forming as few communities as possible. The second part of the algorithm (lines 6 - 9) adds random sensors to C from the set $\mathcal{S} - C$ until $|C| = n_c$.

Given all sensors S, n_t and n_s as inputs, Algorithm 2 outputs a set of communities $\mathbb{C}$ where $|\mathbb{C}| = n_t$ and no sensor is repeated more than n_s times among all communities. In contrast to Algorithm 1, this algorithm creates communities of different sizes. Equally to the n_c parameter of Algorithm 1, n_t has the property of generalizing how the community members collaborate as described in Section 3.3.

Algorithm 2: comm$_2$($\mathcal{S}, n_t, n_s$)

```
1  if $n_s > n_t$ then
2      $n_s = n_t$
3  $C_1, C_2, \ldots, C_{n_t} \leftarrow \{\emptyset\}, \{\emptyset\}, \ldots, \{\emptyset\}$
4  $\mathbb{C} \leftarrow \{C_1, C_2, \ldots, C_{n_t}\}$
5  for $s \in \mathcal{S}$ do
6      $x \leftarrow Uniform(1, n_s)$
7      $T \leftarrow \{\emptyset\}$
8      for $1$ to $x$ do
9          $\mathcal{C} \leftarrow \text{rand}(\mathbb{C} - T)$
10         $\mathcal{C} \leftarrow \mathcal{C} \cup \{s\}$
11         $T \leftarrow T \cup \{\mathcal{C}\}$
12     return $\mathbb{C}$
```

Algorithm 2 follows the following strategy. Lines 3 and 4 initialize the set $\mathbb{C}$ with n_t empty communities. The first loop of the algorithm (line 5) iterates over each available sensor $s \in \mathcal{S}$ to distribute it in the second loop (line 8). Each sensor s is placed, according to a uniform distribution in $[1, n_s]$, in multiple communities. It is possible that some communities are never chosen in line 9 and communities from the initial set $\mathbb{C}$ remain empty. These empty communities are discarded.

3.5 Community-based Intrusion Detection

Each community $C \in \mathbb{C}$ represents an overlay where all sensors $s \in C$ are able to freely communicate with the community head, $s^*_{\mathcal{C}}$. All sensors $s \in \mathcal{S}$ extract features from the network they monitor and forward them to their respective community head where all these are bundled into one *aggregated training dataset*. Each $s^*_{\mathcal{C}}$ $\forall \mathcal{C} \in \mathbb{C}$ learns a model of normality using its aggregated training dataset, performs anomaly detection, and sends all resulting alarms to the central administration interface u. The unit u receives the alarms of all $|S^*|$ community heads, sorts the alarms by anomaly score, and reports the top-most anomalous alarms according to a predefined threshold limited by the FAs.

After establishing a model of normality with the aggregated training dataset, the community heads perform anomaly detection using an *aggregated testing dataset* also gathered within the community. Sensors keep sending the same extracted data features used for creating the aggregated training dataset to the community head. However, the data features are now bundled into an aggregated testing dataset. The outcome of performing anomaly detection is the raising of alarms. Every community head sends these alarms to a central unit where alarm correlation and further analysis takes place.

4 Evaluation

This section presents the results of detecting attacks in a modified version of the DARPA dataset using our novel idea of communities (cf. Section 3) coupled with the anomaly detection algorithm LERAD (cf. Section 2). This evaluation demonstrates how communities outperform isolated sensors in the task of detecting intrusions using anomaly detection.

In our tests we compare the network intrusion detection capabilities of centralized, isolated, and community-based CIDSs. Community-based systems are a variant of centralized and isolated ones that represent a trade-off between scalability and accuracy. Each community analyzes the network traffic of multiple sensors and provide better scalability than centralized systems and better accuracy than isolated systems.

4.1 The DARPA Dataset

The dataset used for evaluation purposes is the DARPA dataset [6]. Regardless of this dataset being outdated and not representing modern traffic patterns, we argue that its usage does not disturb the evaluation results: The dataset is used to compare the performance of three different systems under the same conditions; all of them utilizing the same labeled data. Moreover, the general availability of this dataset and the precisely labeled traffic, without incorrect labels, makes this dataset more useful in this particular context than other alternatives such as the MAWILab [16] or the CDX [17] dataset.

For the evaluation of our approach, we modified the DARPA dataset to reflect the placement of multiple sensors at different points in the network rather than only at one. The description of how this is performed follows.

Modifications to the DARPA Dataset The DARPA intrusion detection dataset [6] is a collection of network traffic obtained from a simulated military computer network with labeled attacks. In this evaluation, only the data records of incoming traffic are taken into account. There are a total of three weeks of training data and two weeks of testing data in the form of packet captures (pcap files). Only the third week of training data and both weeks of testing data are used. The training data does not contain attacks and is used to create models of normality. The testing data contains normal network traffic and 201 attacks ranging from denial of service to exploitation attempts. Due to the modifications described in the following paragraphs, 19 attacks are removed, i.e., traces of these attacks have been dropped as if no sensor was able to pick these up.

In the original dataset all network packets are captured by a single sensor at the ingress point of external traffic. For the purpose of testing the performance of multiple sensors analyzing the data independently of each other and within communities, the DARPA dataset is split according to the visible end-hosts in the local network. The incoming external traffic is split as if only end-hosts captured the traffic. Our modified DARPA dataset emulates multiple sensors, each monitoring a single computer system, gathering data independently of each other. As a consequence, the original testing and training network traffic is split according to the local IPs found in the training set as if captured by multiple sensors instead of only one.

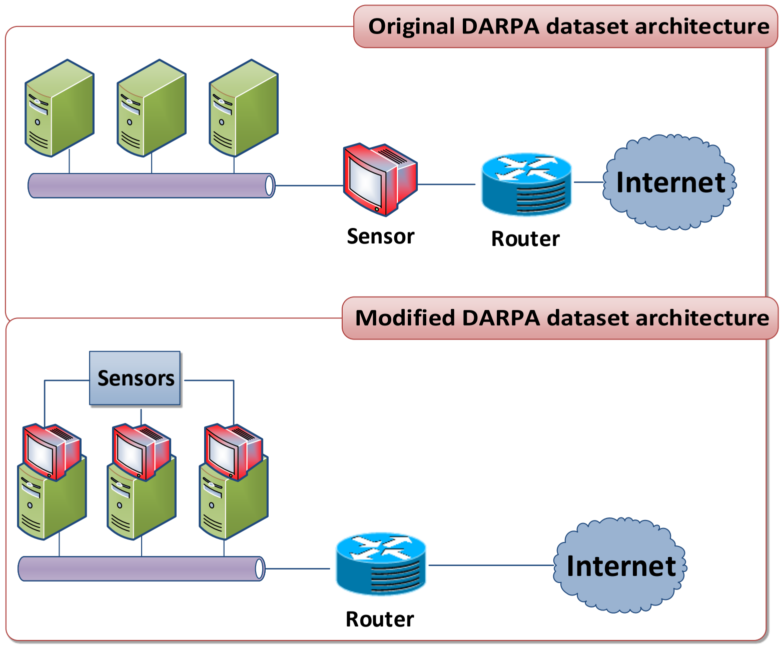

Original DARPA dataset architecture
Sensor
Router
Internet
Modified DARPA dataset architecture
Sensors
Router
Internet

Fig. 2. Modifications made to the 1999 DARPA Dataset.

The DARPA modifications are illustrated in Fig. 2. The red sensor icons indicate the locations where network data is gathered. In the original DARPA dataset, one sensor, at the ingress point, collected all network traffic. Our modified DARPA dataset emulates multiple sensors, each monitoring a single computer system, gathering data independently of each other.

Splitting the original dataset caused two important changes in the resulting dataset. First, all packets targeting an IP address of a non-existent endpoint in the local network are discarded as if no sensor would have seen these. The discarded packets were mostly generated by services that probed a large range of arbitrary IP addresses. Second, we discarded all packets targeting a local IP address in the testing dataset targeted by incoming traffic that is not present in the training dataset. Many packets in the original testing dataset targeted local IP addresses not associated with normal traffic. Hence, for such traffic we cannot derive a model of normality. The end result was a training dataset containing 15 sensors (15 different IP addresses).

4.2 The LERAD Integration

LERAD [8] is used as the detection mechanism of all community heads $s^* \in S^*$. Each community head runs LERAD on its *aggregated training data* to learn rules that describe the network traffic of its community. These rules are the model of normality used for finding anomalies in the *aggregated testing dataset*. Records in the *aggregated testing dataset* are compared with the learned rules and the ones violating these are assigned an anomaly score. The rule violations, or alarms, are sent to the central administration interface u. The role of u is to collect and sort all alarms by anomaly score.

In the process of building the aggregated testing and training datasets, network traffic goes through pre-processing to extract 23 features which are effective for LERAD [8]. For each observed TCP stream we extract the date and time; the destination and source address; the destination and source port; the duration of the TCP stream; the TCP flags of the first, second to last and last packets of the TCP stream; the byte length of the stream; and the first 8 words of the stream.

4.3 Experimental Setup

We evaluate the *accuracy* and the *precision* of detection. Accuracy is defined as the total number of attacks detected over the total number of attacks. The precision equates to the true alarms (or true positives) over the total number of alarms (true alarms + false alarms). Due to the stochastic nature of LERAD, we run each experiment 500 times and average the accuracy and precision of all runs. The confidence intervals of these measurements are omitted in the figures, except for Figure 3(a), as they are insignificant.

The detection accuracy and precision are measured using the alarms the central administrator interface u receives from all community heads. In a pre-processing stage, duplicated alarms within a time-frame of 60 seconds are removed as, according to the original DARPA competition, alarms are deemed true if they detect an attack withing 60 seconds of its occurrence. We analyze each alarm, from highest to lowest anomaly score, assessing if the alarm is a true or false positive. This process continues until a predefined number of FAs is reached and all remaining alarms are discarded. In every experiment, we test the accuracy and precision with different numbers of random communities. Three cases can be distinguished given the size of the community:

Centralized System ($n_c = 15$): All sensors send the extracted features to a single community head.

Isolated System ($n_c = 1$): A community for each sensor ($|\mathbb{C}| = 15$) on its own without any cooperation.

Communities ($n_c = x \mid 1 < x < 15$): Variable number of communities.

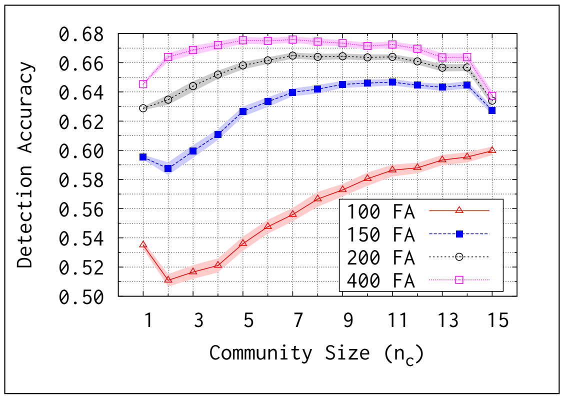
On the one hand, the community of size 15 is expected to outperform all others, in terms of detection accuracy and precision, given that all the features extracted are available in one single location for analysis. On the other hand, it is expected that 15 single independent communities will perform the worst overall as there is no collaboration involved. In the following Subsection, we show that as communities include more sensors, the detection accuracy and precision is improved while at the same time leveraging the communication overhead. In addition, we show that under certain conditions the communities achieve a detection precision similar to the centralized system with a better detection accuracy.

4.4 Results

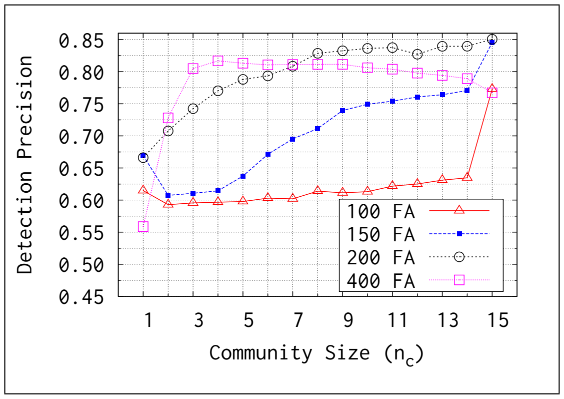
The analysis baseline is shown in Figure 3, where we compare the detection accuracy and precision of every possible community size n_c, as built by Algorithm 1. Figures 3(a) and 3(b) show the outcomes of our experiments varying the FA limit, i.e., changing the threshold for raising alarms. Each anomaly detection experiment is carried out until a predefined number of FAs are issued. At this point, the detection is stopped and the results are recorded. We measure the detection capabilities using 100, 150, 200 and 400 FAs. The testing data corresponds to two weeks (10 total days) of data; as such, 100 FAs equates to an average of 10 FAs per day, 150 to 15 FAs per day, and so on. The shaded area around the solid lines in Figure 3(a) show the confidence intervals of the measurements.

After 100 FAs are found in the sequential analysis of each alarm, from highest anomaly score to lowest, the accuracy and precision of the detections are reported. Figure 3(b) shows that as communities grow in size, the precision is improved. This translates to our hypothesis that the centralized system would have the highest accuracy and precision rates. As seen in both Figures 3(a) and 3(b), if the 100 FA restriction is relaxed, some community sizes are able to improve the detection accuracy in contrast to the centralized system (when $n_c = 15$). At 200 FAs, most community sizes have better detection accuracy than the centralized system. In addition, relaxing the FA restriction allows the detection precision to converge to the one of the centralized system. Lastly, at 400 FAs, a point is reached where every community is able to outperform, in terms of accuracy, both the individual approaches as well as the centralized system. It should be noted that above the 400 FA limitation, no significant changes are observed. However, as seen in Figure 3(b), the precision drops as the FAs are increased. With the 200 FAs limitation, communities with $n_c \in [9, 11]$, quickly approach the precision ratio of the centralized system.

The number of repeating sensors (n_s) has also some interesting properties that impact the detection accuracy of fixed community sizes. We show the experiments of varying $n_s \in [1, 5]$ with Algorithm 2 in Figure 4(a). The graphs being plotted show the impact n_s has on the detection accuracy with respect to the number of communities n_t. As more sensor repetitions are allowed, the overall



(a) Detection accuracy evaluated at different False Alarm (FA) rates.

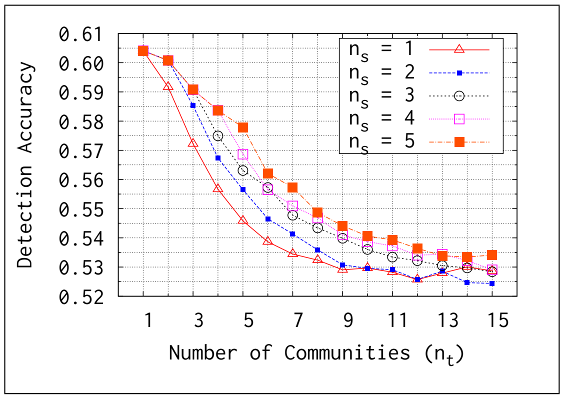


(b) Precision of detections at different False Alarms (FA).

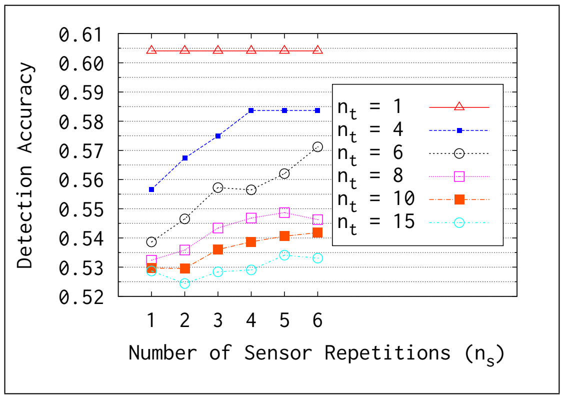
Fig. 3. Detection accuracy and precision at different FAs when communities are built using Algorithm 1.

accuracy is improved. Here we also see the centralized system ($n_t = 1$) still outperforming all others. Furthermore, Figure 4(b) strengthens our aforementioned statement that as n_t increases, the impact of n_s decreases.

To sum up, our results indicate a number of interesting facts. First, as expected, a centralized architecture outperforms all others when the threshold for raising alarms is set high, i.e., when the number of FAs is constrained to low values. Nevertheless, communities provide fair detection and precision ratio and better communication overhead in comparison to a centralized system, while already outperforming individual IDSs at the lowest tolerated FA limit of 10 average alarms per day (100 FAs). Isolated sensors perform no collaboration and, in consequence, create less accurate models of normal traffic than the ones created by collaborating communities. Second, as the threshold for raising alarms



(a) Detection accuracy depending on the number of communities n_t evaluated using different repetitions n_s.



(b) Detection accuracy depending on the number of sensor repetitions n_s.

Fig. 4. Accuracy when the communities are built using Algorithm 2.

is lowered (allowing 200 or more FAs), communities start to perform similarly to the centralized system; finally being able to outperform it (in terms of detection accuracy). This performance can be explained by the fact that, due to the stochastic nature of our algorithm, there is a point where communities are able to gather together enough sensors to generate accurate enough models of normality that explain general network traffic patterns. In addition, these results also comply with our initial argumentation that our community-based CIDS has properties similar to ensemble learning. We are able to improve performance by using different models of normality learned by different communities. Overall, our results in Figures 3(a) and 3(b) indicate that it is possible to find a combination of parameters n_t, n_c, n_s and a particular threshold for raising alarms that enables communities to perform close the a centralized system while reducing the communication overhead. For the particular instance of the modified DARPA dataset, we found that the best results are found when the community size $n_c = 9$, the repetitions $n_s = 3$, the total communities $n_t = 4$ and the FA threshold is set to allow 200 FAs.

5 Conclusion

The continuous sophistication of network attacks urges the development of novel IDSs and architectures. Collaborative IDSs (CIDSs) focus on techniques that group sensors and create a holistic view of the monitored network. In this paper, we presented a CIDS concept that applies the novel idea of communities of sensors that collaborate exchanging features of network traffic to create sufficiently accurate normality models for performing anomaly detection. We developed stochastic algorithms that group sensors into communities and demonstrate how these communities are able to leverage the detection capabilities and communication overhead of CIDSs. Our experimental results indicate that our community-based CIDS concept, performs better than isolated systems in terms of detection accuracy and precision. Furthermore, we demonstrated that communities can perform similarly to centralized systems even though less information is distributed to build normal models for anomaly detection and, as such, less communication overhead is involved. Lastly, we observed that if the threshold for raising alarms is lowered, communities are able to outperform the centralized system.

Future work will comprise additional criteria for community creation. For instance, we will investigate sensor coupling on the basis of exchanged fingerprints of locally monitored traffic, to interconnect sensors with similar traffic patterns. Moreover, we will focus on distributed algorithms for community formation.

References

1. P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, Feb. 2009.
2. E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, “Taxonomy and Survey of Collaborative Intrusion Detection,” *ACM Computing Surveys*, vol. 47, no. 4, p. 33, 2015.
3. Y. C. Cai, Min, Kai Hwang, Yu-Kwong Kwok, Shanshan Song, “Collaborative Internet Worm Containment,” *IEEE Security and Privacy Magazine*, vol. 3, no. 3, pp. 25–33, May 2005.
4. D. Peteiro-Barral and B. Guijarro-Berdiñas, “A survey of methods for distributed machine learning,” *Progress in Artificial Intelligence*, vol. 2, no. 1, pp. 1–11, Nov. 2012.
5. Z.-H. Zhou, “When semi-supervised learning meets ensemble learning,” *Frontiers of Electrical and Electronic Engineering in China*, vol. 6, no. 1, pp. 6–16, Jan. 2011.

6. R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, “The 1999 DARPA off-line intrusion detection evaluation,” *Computer Networks*, vol. 34, no. 4, pp. 579–595, Oct. 2000.
7. V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A Survey,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.
8. M. Mahoney and P. Chan, “Learning rules for anomaly detection of hostile network traffic,” in *IEEE International Conference on Data Mining.* IEEE Comput. Soc, 2003, pp. 601–604.
9. R. Maclin and D. Opitz, “Popular ensemble methods: An empirical study,” *Journal Of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.
10. P. Kannadiga and M. Zulkernine, “DIDMA : A Distributed Intrusion Detection System Using Mobile Agents,” in *International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing.* IEEE, 2005, pp. 238 – 245.
11. Z. Zhang, J. Li, C. N. Manikopoulos, J. Jorgenson, and J. Ucles, “HIDE : a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification,” in *IEEE Workshop on Information Assurance and Security.* IEEE, 2001, pp. 85–90.
12. M. Marchetti, M. Messori, and M. Colajanni, “Peer-to-Peer Architecture for Collaborative Intrusion and Malware Detection on a Large Scale,” *Lecture Notes in Computer Science*, vol. 5735, pp. 475–490, 2009.
13. M. E. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo, “Towards Collaborative Security and P2P Intrusion Detection,” in *IEEE Workshop on Information Assurance and Security.* IEEE, 2005, pp. 333 – 339.
14. C. Duma, M. Karresand, N. Shahmehri, and G. Caronni, “A Trust-Aware, P2P-Based Overlay for Intrusion Detection,” in *International Conference on Database and Expert Systems Applications (DEXA’06).* IEEE, 2006, pp. 692–697.
15. A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, “Peer-to-peer membership management for gossip-based protocols,” *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 139–149, Feb. 2003.
16. R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, “MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking,” in *6th International COnference on - Co-NEXT ’10.* ACM, 2010, pp. 1–12.
17. B. Sangster, T. Cook, R. Fanelli, E. Dean, W. J. Adams, C. Morrell, and G. Conti, “Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets,” in *USENIX Security’s Workshop on Cyber Security Experimentation and Test (CSET)*, 2009.