Dynamic Information Source Selection for Intrusion Detection Systems

Martin Rehak[†], Eugen Staab[£], Michal Pechoucek[†],
Jan Stiborek[†], Martin Grill[†‡] and Karel Bartos[†‡]
† Department of Cybernetics, Czech Technical University in Prague, Czech Republic
£ Communicative Systems Laboratory, University of Luxembourg, Luxembourg
‡ CESNET, z.s.p.o., Czech Republic
martin.rehak@agents.felk.cvut.cz

ABSTRACT

Our work presents a mechanism designed for the selection of the optimal information provider in a multi-agent, heterogeneous and unsupervised monitoring system. The self-adaptation mechanism is based on the insertion of a small set of prepared challenges that are processed together with the real events observed by the system. The evaluation of the system response to these challenges is used to select the optimal information source. Our algorithm uses the concept of trust to identify the best source and to optimize the number of challenges inserted into the system. The mechanism is designed for intrusion/fraud detection systems, which are frequently deployed as part of online transaction processing (banking, telecommunications or process monitoring systems). Our approach features unsupervised adjustment of its configuration and dynamic adaptation to the changing environment, which are both vital for these domains.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*; C.2.0 [**COMPUTER-COMMUNICATION NETWORKS**]: General—*Security and protection*

General Terms

Algorithms, Security

Keywords

trust, service selection, security, intrusion detection

1. INTRODUCTION

The increasing number of electronic transactions and a growing volume of security surveillance data, motivate a rapid development of intelligent systems that are able to autonomously process the data and autonomously identify cases of anomalous behavior. Typical applications of such systems are fraud detection in electronic transactions [2], analysis of the maritime shipping data provided by the AIS active position-tracking system [12] and network intrusion detection [16].

We are particularly interested in the self-adaptive systems based on the anomaly detection paradigm [7]. Systems based on this paradigm do not rely on a predefined set of rules or attack signatures. Instead, they use the past observation of normal system behavior to build a predictive system model and to compare the predicted state of the system with the actual observations. The discrepancies between the prediction and observation are then considered as anomalous and are reported to supervisors/administrators.

The anomaly detection methods do not extract all features of the data, but concentrate on the features that are characteristic of typical attack cases in a given domain. The industrially-deployed anomaly detection methods (such as those discussed in Section 4) implicitly contain a significant amount of expert knowledge about the problem domain. This expert knowledge compensates for the lack of appropriate (classified) training data which is typically inaccessible due to the high data volumes and thus expensive manual classification. Furthermore, the individual anomaly detection methods suffer from relatively high classification error rates (detailed in Sect. 2) and the system needs to combine several anomaly detection methods to improve their results [1]. This approach is similar to the ensemble classification/learning techniques [17], that use a set of classifiers differentiated either by their training set, their algorithm or used features to provide better results. However, most ensemble classification approaches are based on the use of representative and completely labeled training sets.

We position our work in domains where it is difficult to obtain such training sets, due to the combination of a high number of events in the data, highly dynamic system behavior and relatively high cost associated with labeling the training set elements with ground truth. Our goal is to select the best classification provider from the set of classifier agents, in an unsupervised, open and dynamic system. We explicitly do not address the issues related to *building* the best aggregation function, but our method provides a means of *evaluating* the performance of the agents in the system and *selecting* the best existing agent.

We assume that the system (as shown in Fig. 1) contains several types of agents: all *classifier agents* process the events in the shared input sequence and classify them as normal or anomalous. Some of these agents will use anomaly detection or other techniques to classify the events individually,

Cite as: Dynamic Information Source Selection for Intrusion Detection Systems, Martin Rehaky, Eugen Staab, Michal Pechouceky, Jan Stiboreky, Martin Grillyz, Karel Bartos, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1009–1016

while the others may specialize in aggregating the opinions of the others, embodying an aggregation function and thus defining a specific individual classifier combination. In our work, we do not distinguish between these two subgroups. The *user agents* are agents that represent user preferences towards the system and are tasked with dynamic identification of the optimal system output for a specific user. In order to select the optimal system output, a user agent challenges a classifier agent with events for which he already knows the actual class they belong to. These so-called *challenges* [24] can be inserted between the events processed by the system, evaluated with the other events and the correctness of their classification can be used to evaluate the classifier's effectiveness. The challenges do not completely cover the behavior and characteristics of the event sequence (as would be necessary for a representative training set), but rather cover few specific cases that are of particular interest for a given user agent and that would manifest themselves in a relatively uniform manner in the background event sequences with varying characteristics.

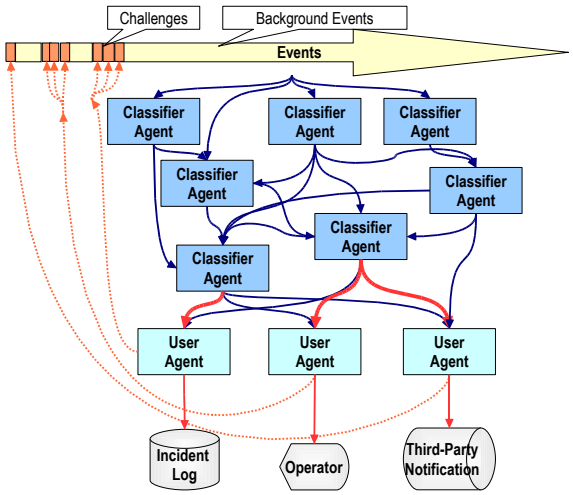

Figure 1: System overview, with underlying events, classifier agent's evaluations (solid blue connectors), optimal evaluations from the user agent's perspective (solid, bold red connectors) and inserted challenges (dashed orange connectors).

The approach to partner evaluation based on response to challenges has several interesting properties that are especially relevant in the intrusion detection context:

- The system protects the **privacy of preferences** of each user agent by hiding the challenges in the real data in a manner which makes them undistinguishable from the background events by classifier agents and protects the identity of the challenge-injecting user agent from the other user agents. Such indirect expression of preferences protects the system from insider attacks against configuration files or other explicit policy specifications.

- Besides the protection of the individual user preferences, the system is also naturally extensible into **competitive environments**. When we postulate the classifier agents provided by several service provider companies, the use of the system presented in this paper allows efficient selection of the best intrusion detection partner (represented by one or more classifier agents) and real-time monitoring of this partner's performance.

The remainder of the paper is organized as follows. In Sect. 2, we formulate the problem of selecting the best classifier agent on an abstract level. In Sect. 3, we present a solution to this problem. This solution is used by the prototype described in Sect. 4, which is evaluated in Sect. 5. In Sect. 6, we discuss literature related to our work before concluding in Sect. 7.

2. ABSTRACT PROBLEM SPECIFICATION

The problem features a set of **classifier agents** $A = \{\alpha, \ldots, \alpha_g\}$ that process a single, shared open-ended sequence $\Phi = \langle \varphi_1, \ldots, \varphi i, \ldots \rangle$ of incoming **events** and use their internal models to divide these events into two categories: *normal* and *anomalous*. The events are inherently of two fundamental types: *legitimate* and *malicious*, and the goal of the classifier agents is to ensure that the normal class as provided by the agent is the best possible match to the legitimate traffic class, while the anomalous class should match the malicious class. The classification thus has four possible outcomes [16] for each event φ, two of them being correct classifications and two of them the errors (see also the *confusion matrix* in Table 1).

Table 1: Confusion matrix

		actual class	
		legitimate	malicious
classification	normal	*true positive*	*false positive*
	anomalous	*false negative*	*true negative*

The classifier agents actually provide more information, as they internally annotate the individual events φ with a continuous "*normality*" value in the $[0, 1]$ interval, with the value 1 corresponding to perfectly normal events and the value 0 to completely anomalous ones. This continuous anomaly value describes an agent's opinion regarding the anomaly of the event, and the agents apply adaptive or predefined thresholds to split the $[0, 1]$ interval into the normal and anomalous classes.

Given that the characteristics of the individual classifier agents α_k are unknown in the dynamically-changing environment, the system needs to be able to identify the optimal classifier autonomously. Furthermore, the system can have several users with different priorities regarding the detection of specific types of malicious events. In the network monitoring use-case, some of the users concentrate on major, infrastructure-type events only (such as denial of service attacks), while the other users seek information about more subtle attack techniques targeting individual hosts. The users are represented by the **user agents** and these agents are assumed to know their users' preferences. Their primary goal is to use their knowledge of user preferences to dynamically identify the optimal information source and to change the source when the characteristics of the environment or user preferences change. To reach this goal in an environment where they have no abstract model of classifier agents' performance, they rely on empirical analysis of classifier agents' response to a pre-classified set of challenges.

In the following, we will analyze the problem from the perspective of a single user agent, which tries to select the best available classification agent, while keeping the number of challenges as low as possible. The **challenges** are events with known classification, which can be inserted into the flow of background events as observed by the system, processed by the classifier agents together with the background events and finally removed before the system reports the results to the users. The processing of the challenges allows the user agents to identify the agent which achieves the best separation of the challenges that represent known instances of legitimate behavior from the challenges that represent known malicious behavior.

The challenges inserted by the user agent are of two different types: the *malicious challenges* correspond to known attack types, while the *legitimate challenges* represent known instances of legitimate events that tend to be misclassified as anomalous. Each classifier agent is then characterized by two probability distributions, empirically estimated from the continuous anomaly values attributed to the two types of challenges, as we can see in Fig. 2. We assume that the anomaly values of both the legitimate and malicious challenges define normal distributions, with the parameters $\bar{x}$ and σ_x for the malicious challenges and $\bar{y}$ and σ_y for the legitimate ones. The distance between the estimated mean values of both distributions ($\bar{x}$ and $\bar{y}$), normalized with respect to the values σ_x and σ_y represents the quality of the classifier agent, defined as its *effectiveness*: the ability to distinguish between the legitimate events and the attacks. The effectiveness of an agent is the value which will be estimated by the trust modeling approach introduced in Sect. 3.

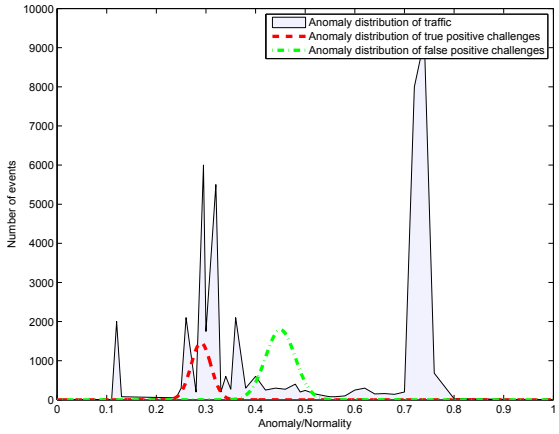

Figure 2: Distribution of challenges on the background of the anomalies attributed to one set of events. The anomalous events are on the left side of the graph, while the normal events are on the right.

The insertion of challenges into the real traffic is not only a difficult problem from the technical perspective (due to the high volume of events processed in near-real-time and hard performance limitations of the system), but can also influence the effectiveness of the classifier agents based on anomaly detection approaches. As these agents are not able to distinguish the challenges from the real events, the challenges are included in their traffic model, making it less representative of the background traffic and therefore reducing its predictive ability.

In the next section, we present a user agent algorithm which dynamically determines the *optimal number of challenges necessary for the reliable identification of the best classifier agent*, while taking into account the: (*i*) past effectiveness of the individual classifier agents, (*ii*) number of classifier agents and the perceived differences in their effectiveness, (*iii*) hard performance limitations imposed by the system and (*iv*) user-defined confidence levels.

The mechanism as presented in this paper only relies on the direct trust experiences [11] of each user agent (interaction trust). Extending the mechanism with other approaches, such as witness reputation or certified reputation is straightforward. On the other hand, the use of such collaborative mechanisms has important security implications, and will be a subject of a dedicated publication in the future.

3. ALGORITHM

In this section we present a simple but adaptive algorithm for choosing the best classifier agent. For each time step $i \in \mathbb{N}$, the algorithm proceeds as follows:

1. Let each classifier agent classify a set of challenges.
2. Update the *trust value* of each classifier agent, based on its performance on the challenges in time step i.
3. Accept the output of the classifier agent with the highest trust value as classification of the remaining events of time step i.

In the following, we first show how the trust values are computed (Sect. 3.1), and secondly, how to determine the number of challenges that is necessary to guarantee accurate trust values (Sect. 3.2).

3.1 Trust Model

As described in Sect. 2, we challenge classifier agents in each time step i with events for which we already know the actual class, i.e. whether they are malicious or legitimate. So, we challenge a classifier agent α with a set of malicious events and a set of legitimate events. For each of these two sets, the performance of the agent is described by a mean and a standard deviation: $(\bar{x}, \sigma_x)$ for the set of malicious challenges and $(\bar{y}, \sigma_y)$ for the set of legitimate challenges. Both means lie in the interval $[0, 1]$, and $\bar{x}$ close to 0 and $\bar{y}$ close to 1 signify accurate classifications of the agent respectively (see again Fig. 2). Based on this performance in time step i, we define the trust experience t_α^i with that classifier agent α as follows:

$$t_\alpha^i = \frac{\bar{y} - \bar{x}}{\sigma_y + \sigma_x} \,. \tag{1}$$

The intention behind this formula is that an agent is more trustworthy, if its classifications are more *accurate* ($\bar{x}$ is low and $\bar{y}$ is high), and more *precise* (the standard deviations are low). Note that t_α^i lies in $(-\infty, \infty)$; however t_α^i will rarely be negative in practice.

To get the final trust value T_α for an agent α, we aggregate the past trust experiences with that agent, as proposed in the FIRE trust model [11]:

$$T_\alpha = \sum_i w_i \cdot t_\alpha^i \,, \tag{2}$$

where w_i are weights that allow recent experiences a higher impact.

The values of the coefficients w_i are determined according to the FIRE model, and their monotonous decrease with increasing time difference ensures that the most recent values are emphasized relative to older ones. In our current system, the weights decrease exponentially. The system (described in Sect. 4) receives the input events in 5 minute batches, and assigns the same weight to all events in each batch. The weight of the challenges from the batch i is determined as:

$$w_i = \frac{1}{W} e^{(j-i)\frac{\ln(0.1)}{4}} , \quad (3)$$

where the j denotes the current time step, and the value of the coefficient $\frac{\ln(0.1)}{4}$ ensures that challenges from the fifth batch (the oldest one being used) are assigned a weight of 0.1 before the normalization. The normalization is performed simply by dividing all the the weights by the sum of their un-normalized values W to ensure that $\sum w_i = 1$. We are currently using the challenges from the last 5 batches, meaning that the $(j - i)$ part of the exponent takes the values between 0 and 4. Please note that the specific assignment of weights w_i is highly domain specific, and is only included as an illustration of the general principle. Deployment of the mechanism in other domains (such as maritime traffic monitoring) would result in other values.

3.2 Determining Number of Challenges

The number of challenges used as basis for the computation of the trust experiences t^i_α should be as small as possible while at the same time providing accurate results for the trust experiences. This means that we want to know the minimum number of challenges n for computing $\bar{x}$ and $\bar{y}$ which gives certain guarantees about the estimation of the actual means μ_x and μ_y (estimated by $\bar{x}$ and $\bar{y}$ respectively). In Sect. 3.2.1, we show how we can choose a number of challenges n such that our estimates are guaranteed to be within a specified margin of error m. In Sect. 3.2.2, we show how to determine such a margin of error m that fulfills our needs, basing the margin of error on the trustworthiness of the classifier agents in the system. Finally, in Sect. 3.2.3, we show how we choose the final number of challenges, in order to respect computational constraints.

3.2.1 Guaranteeing margin of error m

At the outset, let us make two reasonable assumptions. First, we assume that the samples are normally distributed, which is the common assumption if nothing is known about the actual underlying probability distribution. This assumption has been also consistent with our empirical results so far. Second, as suggested in [15], we assume the sample standard deviations which we found in past observations to be the actual standard deviations σ_x and σ_y.

Then, the following formula gives us the number of challenges n that guarantees a specified margin of error m when estimating μ_x (or μ_y analogously) [15]:

$$n = \left(\frac{z^* \sigma_x}{m} \right)^2 , \quad (4)$$

where the critical value z^* is a constant that determines how confident we can be. Table 2 shows z^* for specific confidence levels. More specifically, the integral of the standard normal distribution in the range $[-z^*, z^*]$ equals the respective confidence level. If z^* is for instance chosen for a confidence level of 99%, we know that if we use n challenges for computing $\bar{x}$, the actual mean μ_x will lie in the interval $\bar{x} \pm m$ with the probability of 0.99.

Table 2: Critical values z^*

confidence level	90%	95%	99%
critical value z^*	1.645	1.960	2.576

3.2.2 Choosing margin of error m

Our mechanism chooses the margin of error m in such a way that we can be confident that the pair-wise order between the most trustworthy agent and every other agent is correct. In the following we explain how this is done.

Let us call the currently most trustworthy agent α and let β be any other agent, so that we have $T_\alpha \geq T_\beta$. We now want to make sure that for the next trust experience in step i, this order is not reversed by chance. Recall that a trust experience t^i_α is defined as shown in formula (1). As we use $2n$ challenges to find $\bar{y}$ and $\bar{x}$ respectively, the overall margin of error for the difference of $\bar{y}$ and $\bar{x}$ will not be higher than $2m$. Hence, the largest margin of error m' for which $t^i_\alpha \geq t^i_\beta$ is guaranteed (with the given confidence) when t^i_α takes the lowest and t^i_β the highest possible value within their "guaranteed" range:

$$t^i_\alpha \geq \frac{\overbrace{\bar{y}_\alpha - \bar{x}_\alpha}^{=:a_1} - 2m'}{\underbrace{\sigma_{y_\alpha} + \sigma_{x_\alpha}}_{=:a_2}} = \frac{\overbrace{\bar{y}_\beta - \bar{x}_\beta}^{=:b_1} + 2m'}{\underbrace{\sigma_{y_\beta} + \sigma_{x_\beta}}_{=:b_2}} \geq t^i_\beta . \quad (5)$$

However, we do not want to choose m' to guarantee $t^i_\alpha \geq t^i_\beta$, but to guarantee that the order $T_\alpha \geq T_\beta$ will not be reversed by chance. Therefore, we also have to account for old experiences. Let c_α denote the sum of the old experiences with agent α, weighted as when computing a new T_α (see eq. (2)), i.e., $c_\alpha = \sum_{j<i} w_j \cdot t^j_\alpha$. Let c_β analogously denote the weighted sum of old experiences with β. Then we need to choose m' such that the worst case is:

$$T_\alpha \geq w_i \cdot \frac{a_1 - 2m'}{a_2} + c_\alpha = w_i \cdot \frac{b_1 + 2m'}{b_2} + c_\beta \geq T_\beta , \quad (6)$$

where the inner equation can be solved to give:

$$m' = \frac{b_2(a_1 w_i - a_2 c_\beta) - a_2(b_1 w_i + b_2 c_\alpha)}{2w_i(a_2 + b_2)} . \quad (7)$$

So, a choice of a margin of error m with the constraint $m \leq m'$, guarantees with the specified confidence that for any agent β the order $T_\alpha \geq T_\beta$ will not be reversed by chance, i.e., when it really is the underlying order. To limit the number of challenges, we choose for each agent the maximal margin of error m that fulfills this constraint, which is given by $m := m'$. Since m' is computed several times for α (for every agent β), we choose for α the maximum among these m' to guarantee the order preservation. Finally, we impose an additional lower bound on m, in order to prevent the number of challenges to grow disproportionally when the differences between the agent's trustworthiness are insignificant.

3.2.3 Considering computational restrictions

Let n be the number of challenges that was determined by one user agent using the algorithm specified in Sections 3.2.1 and 3.2.2. Further, let n_u be the upper bound of challenges that can be processed by a specific user agent. In order to determine this value, the system measures the free processing time in the performance bottleneck (typically free CPU cycles between the successive events or event batches), and thus measures the processing time available for the adaptation process. This time is then transformed into the number of challenges (using the known average processing time per challenge), equally divided between the user agents. User agents use n_u as their event quota.

Then the following number of challenges is used for each agent as a basis for $\bar{x}$ and $\bar{y}$ respectively:

$$\min(\lfloor n_u/2 \rfloor, n) \; . \tag{8}$$

The final number of available samples assumes that the n_u samples are divided equally between legitimate and malicious challenges, allowing at most $\lfloor n_u/2 \rfloor$ challenges of each type.

4. PROTOTYPE DEPLOYMENT

In order to evaluate the theoretical model in a production environment, we have used the mechanism as a component of the CAMNEP network intrusion detection system [20], which is used to detect the attacks against computer networks by means of Network Behavior Analysis (NBA) techniques [22]. These techniques are based on the exploitation of NetFlow/IPFIX traffic statistics provided by network routers [5] that contain only a limited amount of data about each connection and do not include any sample of the actual content of communication. Consistent with the problem specification in Sect. 2, the network traffic is represented as a set of discrete events – the *flows*, defined as packet streams with identical source IP address (*srcIP*), destination IP address (*dstIP*), source and destination port (*srcPrt*,*dstPrt*) and protocol (TCP/UDP/ICMP). Besides these basic properties, the system measures only the duration of each flow, the number of packets and their aggregate size.

The NBA systems are not designed to detect stealth attacks against single hosts, but provide a detection capability against attacks that are *significant from network perspective*, such as horizontal scanning (used to map the network for online hosts, typical of malware propagation), vertical scanning (used to determine the services offered by a host), denial of service attacks and other relevant events. Furthermore, the methods outlined in our system also aim to detect the activity of hosts that were taken over by an attacker (typically using zombie networks) and are used to stage further zombie recruiting or exploitation.

The CAMNEP system used to perform the experiments described in this paper incorporates five different *anomaly detection* [7] techniques presented in literature. Each of the methods works with a different traffic model based on a specific combination of aggregate traffic features, such as:

- entropies of flow characteristics for individual source IP addresses [28],
- deviation of flow entropies from the PCA-based prediction model of individual sources [14],
- deviation of traffic volumes from the PCA-based prediction for individual major sources [13],
- rapid surges in the number of flows with given characteristics from the individual sources [9] and
- ratios between the number of destination addresses and port numbers for individual sources [23].

These algorithms maintain a model of expected traffic on the network and compare it with real traffic to identify the discrepancies that are identified as possible attacks. They are effective against zero-day attacks and previously unknown threats, but suffer from a comparatively higher error rate [16], frequently classifying legitimate traffic as anomalous (false positives), or failing to spot malicious flows (false negatives). The *classifier agents* in CAMNEP can be divided to two distinct classes:

- **Detection agents** analyze raw network flows by their anomaly detection algorithms, exchange the anomalies between them and use the aggregated anomalies to build and update the long-term anomaly associated with the abstract traffic classes built by each agent. Each detection agent uses one of the five anomaly detection techniques mentioned above. All agents map the same events (flows), together with the same evaluation of these events, the aggregated immediate anomaly of these events determined by their anomaly detection algorithms, into the traffic clusters built using different features/metrics, thus building the aggregate anomaly hypothesis based on different premises. The *aggregated anomalies* associated with the individual traffic classes are built and maintained using the classic trust modeling techniques (not to be confused with the way trust is used in this work).
- **Aggregation agents** represent the various aggregation operators used to build the joint conclusion regarding the normality/anomaly of the flows from the individual opinions provided by the detection agents. Each agent uses a distinct averaging operator (based on order-weighted averaging [29] or simple weighted averaging) to perform the $R^{g_{det}} \rightarrow R$ transformation from the g_{det}-dimensional space to a single real value, thus defining one composite system output that integrates the results of several detection agents. The aggregation agents also dynamically determine the threshold values used to transform the continuous aggregated anomaly value in the $[0, 1]$ interval into the crisp normal/anomalous assessment for each flow.

The *user agent* functionality is implemented as a collection of the agents. The user agent creates individual *challenge agents*, each of them representing a specific incident in the past, and these temporary, single purpose agents interact with the data-provisioning layers of the system in order to insert the flows relative to the incident into the background traffic and to retrieve and analyze the detection results provided by the classifier agents.

5. EVALUATION

In this section, we empirically verify the two most important properties of the system:

- Ability to dynamically adapt the number of challenges to the actual lower bound necessary for effective system introspection.
- Ability to select the optimal classifier agent from a finite set of agents.

The CAMNEP system can be easily deployed in many different configurations, determined by the number and type of the detection and aggregation agents in the system. For the experimental evaluation detailed below, we have opted for a modification of the configuration used by the authors of [20], adding one supplementary detection agent based on the TAPS algorithm [23] and 30 different aggregation operators represented by aggregation agents, together with the challenge-based self-adaptation infrastructure described above. The system receives the network traffic in batches, each of the batches containing the list of flows observed during the last 5 minutes on the network (the most commonly used interval for NetFlow processing), together with the flows inserted by the challenge agents. In Fig. 4, we can see that the system observes about 80 000 flows every 5 minutes, with roughly 20 000 flows being malicious.

In Fig. 3, we can see the number of challenges as it evolves over time. At the beginning, the system works with a fixed number of challenges, in order to let the anomaly detection methods in the detection agents adapt to the traffic. Once all the detection agents start (at step 5, after 25 minutes), the system starts to progressively insert more challenges, in order to build an initial assessment of all classifier agents. The number of challenges peaks at around the step 14, when it reaches 100 (all challenges combined). Once a user agent has built the initial trustworthiness for all agents, the number of challenges decreases until it levels out at around 40 (legitimate and malicious challenges combined), where it fluctuates until the end of experiment. However, there are two notable increases to explain: between steps 30 and 40, and after step 60.

These increases can be easily explained when looking at Fig. 5, which shows the number of false positives. During these time intervals, we can notice that the choice of an appropriate aggregation agent has a huge impact on the quality of results, and that the adapted system is able to minimize the number of false positives. The number of challenges is lower between steps 40 and 60, when all agents provide similar results, and increases again around 60, where the performance of the aggregation agents varies somewhat more. On the other hand we can see that the user agent did not manage to avoid a spike in false positives around the step 20, when it did not yet have a representative trust model.

In Fig. 6, we show the selected classifier agent for each time step. At the beginning, the system uses a default aggregation agent 0, which implements a simple arithmetic average. It is interesting to note that the system selects from a relatively small subset of agents (numbered between 20 and 26) that are relatively similar to each other.

The experimental results presented above show that the mechanism defined in Sect. 3 is able to effectively select the classifier agents with a low number of false positives (Fig. 5), and that it is able to dynamically adapt to changing environmental characteristics by modifying the number of challenges inserted into the traffic (Fig. 3). Figure 6 also shows that the system effectively identifies a small subset of relevant classifier agents and then selects the active classifier agent from this subset, consistently with the properties of the trust modeling mechanism used by the user agents.

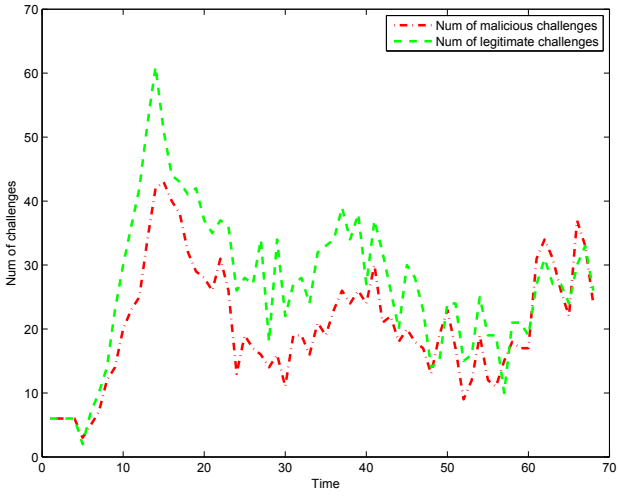

Figure 3: Number of challenges over time, both legitimate (top, green curve) and malicious (bottom, red curve)

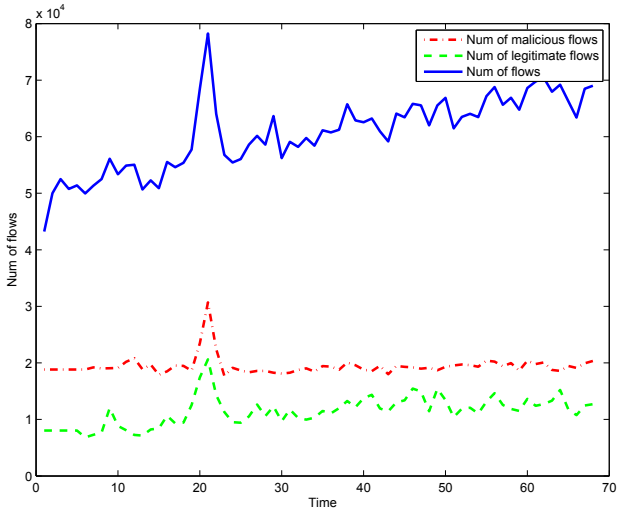

Figure 4: Total number of flows, and a number of known malicious (middle, red curve) and legitimate (bottom, green curve) flows.

6. RELATED WORK

Several trust models base their trust estimation on binary, *positive* or *negative* experiences (e.g. [27, 24, 26]). In our case we deal with continuous assessments, and hence cannot apply these models.

Often, mechanisms that use trust for selecting the "best" service agent out of a set of initially unknown service providing agents, face the *exploration vs. exploitation* dilemma which was investigated in depth in [25, 4, 6]. This dilemma describes the balancing act between choosing the best service agent out of the set of already known agents ("exploit") and looking for currently unknown service agents that might be better than all those that are known ("explore"). However,

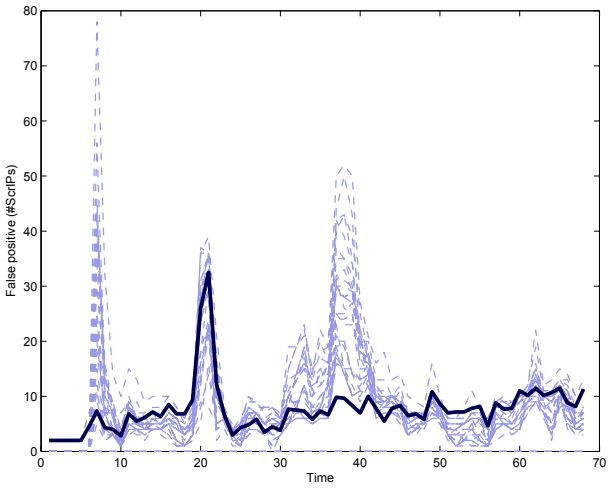

Figure 5: Number of false positives (in number of incidents, identified by unique source IP addresses). Each aggregation (classifier) agent is represented by one thin curve, the solid curve shows the performance of the classifier agent selected by the system.

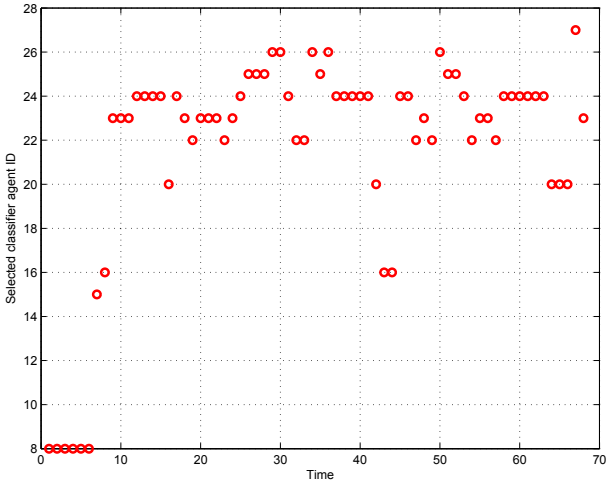

Figure 6: Selected classifier agent (identified by the y axis ID number) for each time step. Note the selection of the default agent 0 at the beginning.

as we use additional challenges for assessing the trustworthiness of the agents, our mechanism avoids the issue of explore vs. exploit. Instead we can – *at the same time* – explore the system with challenges and "exploit" it by means of selecting the best agent.

Reches et al. [18] show how to determine the appropriate amount of (witness-)information about a set of agents in order to chose the best one. In contrast to the lightweight approach presented in our work, their approach involves costly computations. Also, they do not account for the dynamics of a system.

From the related domains, we should mention that some of the ensemble [17, 8] learning approaches, particularly those based on the classifiers diversified by the features of the classified samples, are relevant to our work. However, past attempts to apply these methods in the network security domain [10] are based on the use of pre-classified training data sets to build the optimal aggregation function. They are thus functionally similar to our solution, but are based on an assumption of training data set availability that we do not require (not considering the limited number of challenges).

7. CONCLUSION & FUTURE WORK

In our work, we have introduced a specific use of trust modeling for an internal adaptation of a network intrusion detection system. The technique is applicable to a wide range of domains where classifier agents (considered as information sources) provide a binary classification capability in a dynamic, unsupervised learning environment. Our approach deploys a simplified implementation of an existing trust modeling mechanism in the individual user agents. This mechanism allows the user agents to inject a set of challenges – the events with known classification – into the background defined by the events observed by the system in real time, and use the classification of challenges provided by the individual classifier agents to progressively build and update their trustworthiness, which is a measure of their ability (i.e. competence [3]) to effectively separate the known samples of malicious and legitimate events situated in the current environment.

Besides the simple application of trust modeling techniques for the assessment of other agent's competence, the mechanism is able to determine the optimal number of challenges that needs to be inserted into the system in order to determine the best agent. This number depends on the agent's trustworthiness, classifier characteristics and other dynamically observed data, significantly reducing any human involvement in the adaptation process to design-tie settings of the initial values and admissible boundaries.

The work presented in this paper can be extended in three directions. In the first extension, several user agents can negotiate a joint set of challenges to minimize the negative influence on the quality of the intrusion detection mechanism of the classifier agents. In the second extension, user agents can use an extension of trust modeling which would consider the performance of classifier agents under specific environmental characteristics, implemented by context-sensitive trust modeling techniques [21, 19]. In the final extension, the agents would use the trust modeling mechanism not only to select the best information provider from a predefined set, but they may identify an appropriate combination of existing classifier agents which would further improve their results.

Acknowledgment

This material is based upon work supported by the ITC-A of the US Army under Contract No. W911NF-08-1-0250. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ITC-A of the US Army. This research has been partly supported within the Research Programmes No.MSM6840770038: Decision Making and Control for Manufacturing III (CTU) and MSM6383917201 (CESNET) by the Ministry of Education of the Czech Republic.

8. REFERENCES

[1] P. Barford, S. Jha, and V. Yegneswaran. Fusion and filtering in distributed intrusion detection systems. In *In Proceedings of the 42nd Annual Allerton Conference on Communication, Control and Computing*, 2004.

[2] R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Statistical Science*, pages 235–255, 2002.

[3] C. Castelfranchi and R. Falcone. Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, page 72. IEEE Computer Society, 1998.

[4] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: a bayesian approach. In *Proc. of the 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS '03)*, pages 709–716. ACM, 2003.

[5] Cisco Systems. Cisco IOS NetFlow. http://www.cisco.com/go/netflow, 2007.

[6] R. Dearden, N. Friedman, and D. Andre. Model based bayesian exploration. In *UAI '99: Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence*, pages 150–159, 1999.

[7] D. E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232, 1987.

[8] T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.

[9] L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, and P. Dokas. Minds - minnesota intrusion detection system. In *Next Generation Data Mining*. MIT Press, 2004.

[10] G. Giacinto, R. Perdisci, M. D. Rio, and F. Roli. Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Information Fusion*, 9(1):69–82, 2008.

[11] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI '04)*, pages 18–22. IOS Press, 2004.

[12] F. Johansson and G. Falkman. Detection of vessel anomalies - a bayesian network approach. In *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[13] A. Lakhina, M. Crovella, and C. Diot. Diagnosis Network-Wide Traffic Anomalies. In *ACM SIGCOMM '04*, pages 219–230, New York, NY, USA, 2004. ACM Press.

[14] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies using Traffic Feature Distributions. In *ACM SIGCOMM, Philadelphia, PA, August 2005*, pages 217–228, New York, NY, USA, 2005. ACM Press.

[15] D. S. Moore. *The Basic Practice of Statistics*. W. H. Freeman & Co., New York, NY, USA, fourth edition, 2007.

[16] S. Northcutt and J. Novak. *Network Intrusion Detection: An Analyst's Handbook*. New Riders Publishing, Thousand Oaks, CA, USA, 2002.

[17] R. Polikar. Esemble based systems in decision making. *IEEE Circuits and Systems Mag.*, 6(3):21–45, 2006.

[18] S. Reches, P. Hendrix, S. Kraus, and B. J. Grosz. Efficiently determining the appropriate mix of personal interaction and reputation information in partner choice. In *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS '08)*, pages 583–590, 2008.

[19] M. Rehak and M. Pechoucek. Trust modeling with context representation and generalized identities. In *Cooperative Information Agents XI*, number 4676 in LNAI/LNCS. Springer-Verlag, 2007.

[20] M. Rehak, M. Pechoucek, M. Grill, and K. Bartos. Trust-based classifier combination for network anomaly detection. In *Cooperative Information Agents XII*, LNAI/LNCS. Springer-Verlag, 2008.

[21] A. Rettinger, M. Nickles, and V. Tresp. Learning initial trust among interacting agents. In M. Klusch, K. V. Hindriks, M. P. Papazoglou, and L. Sterling, editors, *Cooperative Information Agents XI, CIA 2007, Delft*, volume 4676 of *LNCS*, pages 313–327. Springer, 2007.

[22] K. Scarfone and P. Mell. Guide to intrusion detection and prevention systems (idps). Technical Report 800-94, NIST, US Dept. of Commerce, 2007.

[23] A. Sridharan, T. Ye, and S. Bhattacharyya. Connectionless port scan detection on the backbone. Phoenix, AZ, USA, 2006.

[24] E. Staab, V. Fusenig, and T. Engel. Towards trust-based acquisition of unverifiable information. In M. Klusch, M. Pechoucek, and A. Polleres, editors, *Cooperative Information Agents XII*, volume 5180 of *LNCS*, pages 41–54. Springer Verlag, September 2008.

[25] W. T. L. Teacy, G. Chalkiadakis, A. Rogers, and N. R. Jennings. Sequential decision making with untrustworthy service providers. In *Proc. of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS '08)*, pages 755–762, 2008.

[26] W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Coping with inaccurate reputation sources: experimental analysis of a probabilistic trust model. In *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*, pages 997–1004, 2005.

[27] Y. Wang and M. P. Singh. Formal trust model for multiagent systems. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI '07)*, pages 1551–1556, 2007.

[28] K. Xu, Z.-L. Zhang, and S. Bhattacharrya. Reducing Unwanted Traffic in a Backbone Network. In *USENIX Workshop on Steps to Reduce Unwanted Traffic in the Internet (SRUTI)*, Boston, MA, July 2005.

[29] R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988.