Trust-Based Classifier Combination for Network Anomaly Detection

Martin Rehák[1], Michal Pěchouček[1], Martin Grill[21], Karel Bartos[21]

[1] Department of Cybernetics and Center for Applied Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague
Technická 2, 166 27 Prague, Czech Republic
{mrehak|pechouc}@labe.felk.cvut.cz
[2] CESNET, z. s. p. o.
Zikova 4, 160 00 Prague, Czech Republic
{bartosk|grillm}@labe.felk.cvut.cz

Abstract. We present a method that improves the results of network intrusion detection by integration of several anomaly detection algorithms through trust and reputation models. Our algorithm is based on existing network behavior analysis approaches that are embodied into several detection agents. We divide the processing into three distinct phases: anomaly detection, trust model update and collective trusting decision. Each of these phases contributes to the reduction of classification error rate, by aggregation of anomaly values provided by individual algorithms, individual update of each agent's trust model based on distinct traffic representation features (derived from its anomaly detection model), and re-aggregation of the trustfulness data provided by individual agents. The result is a trustfulness score for each network flow, which can be used to guide the manual inspection, thus significantly reducing the amount of traffic to analyze. To evaluate the effectiveness of the method, we present a set of experiments performed on real network data.

1 Introduction

This paper presents a specific application of techniques from agent trust modeling in the domain of Network Intrusion Detection and shows how to apply these techniques to combine several intrusion detection methods and improve the quality of their decisions.

The purpose of the *Network Behavior Analysis* (NBA) systems [1] is to identify the attacks against the network infrastructure and hosts by observing the significant events in the structure and volume of network traffic. Most of the NBA systems are based on *anomaly detection* [2] principles – they build the model of the traffic in the network from the past observations, predict the properties of current traffic and identify the potentially malicious actions by comparing the prediction with observed traffic.

The proposed method is based on system observation using the NetFlow data that may be provided by routers (the protocol was originally defined by Cisco

[3]) or specialized devices [4]. Each **flow** corresponds to one direction of TCP connection (or UDP/ICMP equivalent); all packets with the same source IP address (*srcIP*), source port (*srcPrt*), destination address (*dstIP*), destination port (*dstPrt*) and protocol (TCP/UDP/ICMP) constitute the flow. In addition to these definition parameters, we can observe supplementary data, such as number of bytes, packets, duration of flow and other parameters. The NetFlow data is aggregated over an observation period, typically a 5 minute interval. Once aggregated, the NBA system extracts relevant features of the data and concludes which flows are malicious, and which are part of the legitimate network traffic.

The usefulness of current NBA systems is severely impacted by two major shortcomings: their limited effectiveness, i.e. high error rate [5, 6], and relatively low efficiency, which does not allow their deployment on high bandwidth network links. The effectiveness of an IDS system is typically described by two values, the ratios of false positives and false negatives. The **false positives** are the legitimate, non-malicious flows that are classified as malicious, while the **false negatives** are malicious flows classified by the system as legitimate. We will describe the efficiency of an IDS system in terms of number of network flows per second it can process, as this value is directly associated with the network bandwidth.

Our work uses the techniques developed in the field of agent-based trust and reputation modeling to improve the effectiveness of the system. Furthermore, the deployment within an efficient agent platform supports natural parallelization of tasks and thus easy distribution across multiple processor cores. Specifically, we improve the error rate of the collective detection system by:

- separation of short term anomaly detection and long-term trust modeling
- collaboration between heterogeneous trusting agents, with following processes:
  - anomalies are considered by individuals only if they are consistently identified by anomaly detection models of majority of agents
  - anomalies are considered only if the flows that constitute them fall into one or few traffic classes in the trust models of individual agents
  - reputation mechanism integrates the trustfulness data from several agents

In Section 2, we will discuss the necessary extension of trust modeling techniques, before presenting the anomaly detection algorithms in Section 3 and the core contribution of this work in Section 4. We evaluate our solution on real data in the experiments described in Section 5, and discuss the related work before concluding.

2 Extended Trust Modeling

Trust models [7–10] are specialized knowledge structures designed to maintain information about the trustworthiness of the partners, either acquired from agent's own interactions, observed from the interactions of others, or received from other

agents by means of reputation mechanism [11]. The design of trust models emphasizes features such as fast learning, robustness in response to false reputation information [12] and robustness with respect to environmental noise. Extended trust models [13, 14] are used to address several important assumptions of trust models and to make them more relevant for practical deployment. Extended trust models are able to:

- include the **context** of the trusting situation into the reasoning, making the trust model situational,
- use the similarities between trustees and situations to infer their trustfulness during the **first encounter**, and
- protect the model against trustee **identity changes**.

To achieve these goals, extended trust models are inspired by machine learning [14] and pattern recognition [13] approaches. The models achieve these goals by reasoning not about the performance of specific agents, but about the trustfulness of more general identities situated in a specific context. More specifically, each trustee and trusting situation are described by a set of relevant observable features (feature vector). These features define the feature space, a metric space on which the trust model of each agent operates. Trustfulness is determined for significant clusters in this space (i.e. reflects the behavior of a class of similar agents in a similar situation), and Section 4 describes the update and query operations in more detail.

In the network security domain, low *trustfulness* of the flow means that the flow is assumed to be malicious, i.e. a part of an attack. Trustfulness is determined in the $[0, 1]$ interval, where 0 corresponds to complete distrust and 1 to complete trust. The *identity* of each flow is defined by the features we can observe directly on the flow: *srcIP, dstIP, srcPrt, dstPrt, protocol*, number of *bytes* and *packets*. If two flows in a data set share the same values of these parameters, they are assumed to be identical. The *context* of each flow is defined by the features that are observed on the other flows in the same data set, such as the number of similar flows from the same *srcIP* [15], or entropy of the *dstPrt* of all requests from the same host as the evaluated flow [16]. Identity and context features are used to define the feature space for each specific type of agent introduced below.

3 Detection Agent Types

This section briefly presents the anomaly detection techniques and traffic features used by most important types of detection agents in the system. All agents use the same representation of flow identity, but differ in the context dimensions of the feature space, as we will describe below. The feature space distance function (metrics) is a sum of two components, one covering the identity subspace, the other context subspace dimensions. The identity component is identical for all agent types, and type-dependent context distance is described with each agent type below:

MINDS algorithm [15] builds the context information for each flow using the: number of flows from the same source as the evaluated flow, number of flows towards the same destination host, number of flows towards the same destination from the same source port, and number of flows from the same source towards the same destination port. This makes the context space four dimensional, with logarithmic distance scale in each dimension, combined into the global distance as a sum of their squares. Contrary to the original work, we judge the anomaly from the difference between the floating average of past values and observation in each of the four context dimensions.

Xu *et. al.* [16] actually classifies the traffic sources, which imposes the same context for all flows from the same *srcIP*. For each source, we determine the normalized entropy of the set of source ports, destination ports and destination IPs of all the flows from this source, thus defining a 3D context. Anomalies are then determined by application of static classification rules that divide the traffic into normal and anomalous classes. Distance between the contexts of two flows is computed as a difference between the 3 normalized entropies of each flow, combined as sum of squares.

Volume prediction algorithm [17] uses the Principal Components Analysis to build the model of traffic volumes from individual sources in number of bytes, packets and flows. Then, it identifies the difference between the predicted and real traffic for each source IP and all flows from the source are assigned this value transformed into the $[0, 1]$ interval as anomaly. Again, the context is identical for all flows from the same source, and is defined by the difference between the predicted and real number of flows, packets and bytes from the *srcIP* of the flow. Distance in each of the 3 context dimensions is logarithmic, combined as a sum of squares.

Entropy prediction algorithm [18] works in exactly the same manner as the previous type, but predicts the entropies of *dstIP*, *dstPrt* and *srcPrt* instead of traffic volumes. To aggregate the distance in the context subspace, we determine the distance between the residual entropies as an absolute value of their difference, and add their squares. For detailed discussion of agent types and modifications of original algorithms, please refer to our previous publication [19].

4 Collective Detection Process

The detection agents in the system cooperate to improve the detection performance. Each agent is based on a specific anomaly detection method (as introduced above), and each agent's method also defines the features used to represent the context of the flow in its trust model. Therefore, the models differ between the agents, and we can not perform direct translation between them. Specificity of the trust model and anomaly detection method improves anomaly detection results, reduces the dimensionality and reduces the computational requirements of the trust model. It also directly contributes to elimination of false positives, as we will discuss in Section 4.1. On the downside, model specificity limits the

collaboration to those stages in the process when the agents can use a common language to share the anomalies or reputation values relative to individual flows.

At the end of observation interval j, all detection agents $X \in Ags$ receive the same input set Φ_j of network flows $\varphi_{i,j}$, and this data is the only algorithm input for each observation period j. The processing is performed in three stages:

- anomaly detection,
- trust model update, and
- flow classification by individual and collective trustfulness determination

Before the detailed discussion of these stages, we will introduce the most important terms:

Anomaly $A_X(\varphi_{i,j})$ is a number in the $[0, 1]$ interval describing agent's X opinion about the anomaly of the flow $\varphi_{i,j}$ in the current set Φ_j. One represents the maximal anomaly, and zero no anomaly at all. It is provided by the anomaly detection method embedded in the detection agent X.

Trustfulness $\Theta_X(\varphi_{i,j})$ value can be determined for any feature vector in the feature space of agent X. It falls into the $[0, 1]$ interval as well, and it indicates the estimated level of maliciousness. Flows with trustfulness close to 0 are considered to be malicious, while the flows with high trustfulness are classified as legitimate (trusted).

Feature vector $ix_X(\varphi_{i,j})$ represents the identity and context features determined for the flow $\varphi_{i,j}$ by agent X in the feature space. We use the term *centroid* to denote the permanent feature vectors r_k that are positioned in the feature spaces of trusting agents. The centroids act as trustees of the model, and the trustfulness value $\Theta(r_k)$ of each centroid is updated with relevant observations, and used to deduce the trustfulness of feature vectors in its vicinity.

Metrics $dist_X(ix_X(\varphi_{i,j}), ix_X(\varphi_{k,l}))$ determines the distance of two feature vectors in the feature space of agent X. As mentioned above, each agent type has a metrics defined by its context representation (and the shared identity part), and we shall emphasize that: $dist_X(ix_X(\varphi_{i,j}), ix_X(\varphi_{k,l})) = dist_Y(ix_Y(\varphi_{i,j}), ix_Y(\varphi_{k,l}))$ almost never holds for $X \neq Y$.

After the introduction of terms, we will describe the stages of the algorithm:
Anomaly detection. During the anomaly detection stage, each individual agent A uses its embedded anomaly detection method to determine the anomaly $A_A(\varphi_{i,j})$ of each flow of the set Φ_j. As the features (dimensions) of the feature space of agent A are identical to those used by its anomaly detection method, we can (informally) write $A_A(\varphi_{i,j}) = A_A(ix_A(\varphi_{i,j}))$ to emphasize that the information in the feature vector $ix_A(\varphi_{i,j})$ of the flow is sufficient to determine its anomaly. The anomaly values are shared with other detection agents, and used as an input in the second phase of the processing – all agents thus have the same aggregated anomaly value $A_{Ags}(\varphi_{i,j})$ for each flow, shown in Eq. 1, and this value averages the anomaly opinions of all detection agents:

$$A_{Ags}(\varphi_{i,j}) = \frac{1}{|Ags|} \sum_{X \in Ags} A_X(ix_X(\varphi_{i,j})) \tag{1}$$

Trust update. During the trust update, the agents integrate the anomaly values of individual flows from the set Φ_j into their trust models. As the reasoning about the trustfulness of each individual flow is computationally infeasible and unpractical (the flows are single shot events by definition), the extended model holds the trustfulness $\Theta(r_k)$ of centroids r_k (significant flow samples, e.g. centroids of fuzzy clusters) in the feature space, and the anomaly $A_{Ags}(\varphi_{i,j})$ of each flow $\varphi_{i,j}$ is used to update the trustfulness of centroids in its vicinity. Eq. 2 specifies the operation performed for each flow:

$$\Theta'_A(r_k) = \texttt{trust}((\Theta_A(r_k), W_k),\ (1 - A_{Ags}(\varphi_{i,j}), w_k)) \tag{2}$$

w_k is the weight of the update for the trustfulness associated with r_k, decreasing with the distance between $ix_A(\varphi_{i,j})$ and r_k:

$$w_k = e^{-dist_A(ix_A(\varphi_{i,j}), r_k)} \tag{3}$$

and W_k is the aggregated weight of all past updates to $\Theta(r_k)$. The operation `trust` denotes the weighted update of trustfulness, and depends on the trust model used in the mechanism. The trust model we use represents the trustfulness with triangular fuzzy numbers [20], with the core defined as average value of the trust observations (i.e. $1 - A_{Ags}(\varphi_{i,j})$), and the width of the fuzzy number representing the uncertainty or inconsistence of the past observations. The core of the fuzzy number (also denoted Θ_A in this paper, as it is used as a defuzzyfied value in the subsequent parts of the processing) is thus updated as:

$$\Theta'_A(r_k) = \frac{W_k \cdot \Theta_A(r_k) + w_k \cdot (1 - A_{Ags}(\varphi_{i,j}))}{W_k + w_k} \tag{4}$$

When we update the trust models with the first flow $\varphi_{1,1}$ of the first data set Φ_1, there are no centroids present yet, and the centroids are created progressively as the model processes the input data. Creation is based on a simplified Leader-Follower clustering algorithm [21], which always creates a new centroid with the same position as $ix_A(\varphi_{i,j})$ if a distance to the closest existing centroid is greater than predefined cutoff distance[3].

Collective trust estimation. In the last stage of processing, each agent determines the *trustfulness* $\Theta_A(\varphi_{i,j}\, \Phi_j)$ of each flow $\varphi_{i,j}$ from the current set Φ_j. To determine the trustfulness of individual flow $\varphi_{i,j}$, we aggregate the trustfulness $\Theta_A(r_k)$ associated with the centroids in the vicinity of flow's feature vector $ix_A(\varphi_{i,j})$. This operation is shown in Eq. 5.

[3] Cutoff distance thus determines the density of centroids in the feature space, and consequently the computational cost of the model. When the $ix_A(\varphi_{i,j})$ falls into the proximity of an existing centroid, the L-F algorithm specifies that the position of the closest centroid shall move in the feature space towards the last sample. In our implementation, such behavior would be highly undesirable (and thus is not implemented), as the values $\Theta_A(r_k)$ are relative to their positions, and any shift could impact their relevance.

$$\Theta_A(\varphi_{i,j}) = \texttt{weagg}_{r_k}(\Theta_A(r_k), w_k) \tag{5}$$

The operation `weagg` is a suitable weighted aggregation mechanism, and is determined by the trust model used for trustfulness representation. When we concretize the `weagg` operation as a weighted average, we obtain:

$$\Theta_A(\varphi_{i,j}) = \frac{\sum (\Theta_A(r_k) \cdot w_k)}{\sum w_k} \tag{6}$$

with both sums over the set of centroids in agent's trust model.

All agents provide their trustfulness assessment (which is conceptually a reputation opinion) for all flows to the aggregation and visualization agents, and the aggregated values are then used for traffic filtering. To perform the aggregation, we average the trustfulness opinions for each flow, as shown in Eq. 7. Aggregated trustfulness values for each flow constitute the output of the algorithm:

$$\Theta_{Ags}(\varphi_{i,j}) = \frac{1}{|Ags|} \sum_{X \in Ags} \Theta_X(\varphi_{i,j}) \tag{7}$$

4.1 Algorithm Properties

All three stages of the processing as listed above are designed to reach a joint conclusion between several anomaly detection method. We argue (and also show in the experiments on real networks) that the quality of joint conclusion is higher than the quality of the estimation provided by any single method, and that each of the algorithm stages contributes to quality improvement. In the following, we will discuss the elements that improve the quality of the conclusions reached by the proposed system.

Anomaly detection method integration. In Eq. 1, the algorithm integrates the anomaly values for each flow, and therefore minimizes the impact of traffic irregularities reported by a single method. This approach is not novel per se, but it already minimizes the impact of possible false positives using only the anomaly data, before the start of trust update process (see Fig. 1).

Trustfulness aggregation. During the update of the trust model, each agent creates the centroids r_k in its own feature space, and updates their trustfulness $\Theta_A(r_k)$ with the anomaly of flows in their vicinity. If there is only a **single agent** in the system (thus $A_{Ags}(\varphi_{i,j}) = A_A(\varphi_{i,j})$), this aggregation of anomalies into trustfulness has only limited impact. This is due to the fact that the features of the flow determine its feature vector $ix_A(\varphi_{i,j})$, and the anomaly $A_A(\varphi_{i,j})$ is determined from the vector values and status of the anomaly detection model. Therefore, we only aggregate the anomalies with the anomalies determined for similar flows in the past. However, even this aggregation helps to eliminate the effects of non-systematic irregularities in the traffic model.

When we combine **multiple agents** in the system, the situation becomes more interesting as each agents updates the anomaly value $A_{Ags}(\varphi_{i,j})$ over the

centroids in their trust models. As each agent uses a distinct feature space and metrics, it has a different insight into the problem – the flows are positioned (clustered) according to the different criteria, and the cross correlation implemented by sharing of the anomaly values used to update the trustfulness helps to eliminate random anomalies. Let's assume that some of the flows constitute one anomaly, this anomaly was identified by a majority of agents and that the anomaly $A_{Ags}(\varphi_{i,j})$ of these flows is high. This implies that the flows are adjacent in the feature space of the agents that have identified them (because they are part of one anomaly identified by the anomaly detection model), and that during the trust update phase, their anomaly will significantly influence the trustfulness of one or few centroids r_k (see left segment of Fig. 2). On the other hand, when another agent does not detect these flows as anomalous, they will get dispersed among the clusters (not being recognized by agent's methods neither as anomalous, nor as similar to each other), and they will have only limited influence on the trustfulness of the centroids in their neighborhood (right segment of Fig. 2). This effect further eliminates the false positives, as it requires the untrusted flows to have previously unknown features, therefore creating a new centroid and pushing its trustfulness to low values, or to be similar to existing untrusted cluster(s).

From the computational complexity perspective, the trust update/query/integration phases of the presented algorithm is characterized as $|\Phi_j| \cdot |\{r_k\}|$ for each agent [13], thus $|Ags| \cdot |\Phi_j| \cdot |\{r_k\}|$ for the whole system. It is linear in Φ_j, enabling the deployment of the system on the gigabit grade links using single multi-core PC. Memory requirements of each agent's model are linear in the number of centroids $|\{r_k\}|$. Complexity of individual anomaly detection algorithms is not considered in the estimation, but has not been a concern on real data.

5 Experimental Evaluation

In order to evaluate the effectiveness of our approach and to illustrate the properties discussed in Section 4.1, we have used the data acquired during the tests of our system on a live university network. We have performed two types of tests: in the first series, we have launched a series of scanning and profiling attacks against a selected host inside the surveyed network, and observed whether these attacks (using standard nmap scanner [22]) can be discovered on the background of the real traffic. In the second type of tests, we have tried to detect the third party attacks that were independently manually identified by network administrators.

Before the presentation of aggregated results, we will present several situations that illustrate the algorithm behavior discussed in Section 4.1. In Figure 1, we can see the importance of anomaly aggregation as specified in Eq. 1 – when running alone, the agent is not able to detect the attack traffic (horizontal scan, 1000 flows, represented as surface in the graphs) as anomalous, and most of the attack flows are then classified with trustfulness close to 1. On the other hand, when the agent cooperates with the others and uses the common anomaly value

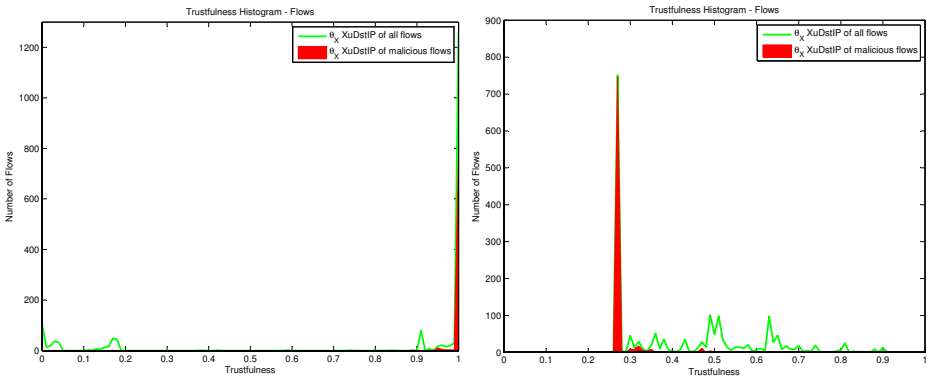

Fig. 1. Histogram of flow number over trustfulness from the results provided by **isolated** run of *XuDstIP* detection agent. (*left*) and the same agent when running with the others (*right*).

$A_{Ags}(\varphi_{i,j})$, it is able to classify the traffic as untrusted. However, the classification is not perfect, as we can see from lower peaks of attack flow trustfulness distribution in the right segment of Fig. 1, between 0.3 and 0.5.

Perhaps the most important cooperative filtering effect is that the flows that are similar for one agent (and fall into the same feature space region) can be dispersed in the feature space of another agent – when they are coherent there as well, and are consistently untrusted by all agents, they are classified as an attack. We can illustrate these two situations in Fig. 2, where we project the attack and false positive flows respectively over the 3D projections of MINDS agent's centroids organized into the tree by similarity. We can see that the attack flows (from a vertical TCP SYN scan) are concentrated in a single centroid, and their high anomaly makes this newly created centroid untrusted. In the false positive case, the flows that were reported by another agent as a possible attack are dispersed over several existing centroids, and their higher anomaly value can not significantly influence the trustfulness of the centroids in their vicinity. Therefore, these flows will be reported with higher trustfulness.

In the experiments presented so far, we have seen the effects of the anomaly aggregation (Fig. 1) and projection on various feature spaces (Fig. 2). However, we should establish whether we need to perform the trust model update and query operations, instead of simply using the aggregated anomaly values directly. In a specific case of a small size vertical scan (300 UDP flows over 5 minutes), we can see that the aggregated anomaly of attack flows (red solid surface in the left segment of Fig. 3) is low, and they are not identified as malicious. On the other hand, the results of trustfulness aggregation provide much better results (Fig. 3, right), as they clearly classify the same traffic as untrusted. This is the effect of *a priori* low trustfulness associated with centroids around the attack flows in feature spaces of detection agents.

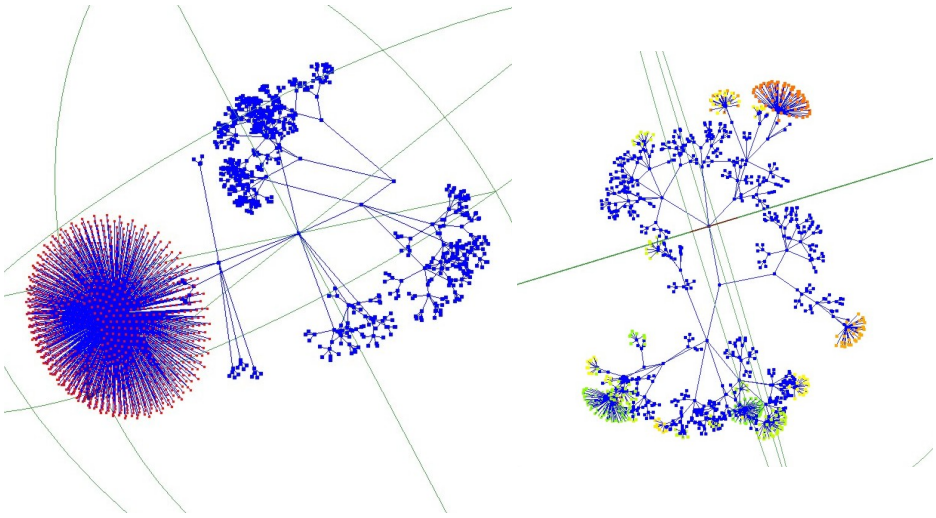

Fig. 2. 3D projection of agent's MINDS trust model. Centroids are organized in a tree, and each flow is attached to the closest centroid and colored by the centroid's trustfulness. The attack flows (*left*) tend to be concentrated around single centroid, while the false positives identified by another agent are spread over the whole model. (*right*)

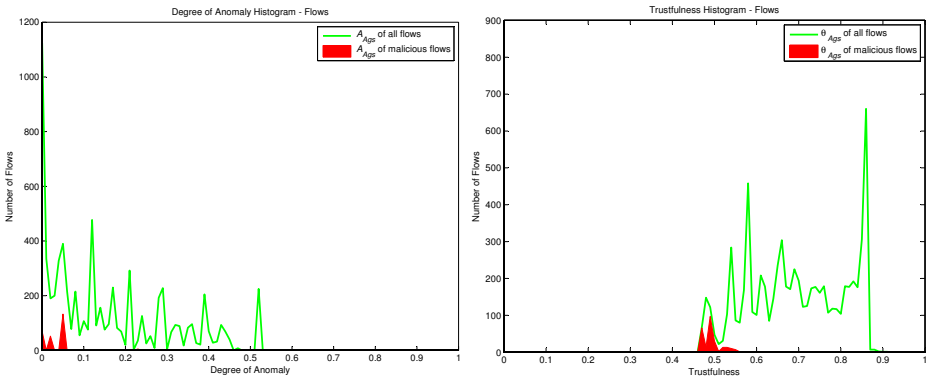

Fig. 3. Histogram of flow number by aggregated anomaly $A_{Ags}(\varphi_{i,j})$ (*left*). The attack (slow vertical scan, 300 flows, red surface) flows are not classified as anomalous. Histogram of flow number by aggregated trustfulness $\Theta_{Ags}(\varphi_{i,j}\,\Phi_j)$ (*right*) correctly classifies the attack as untrusted.

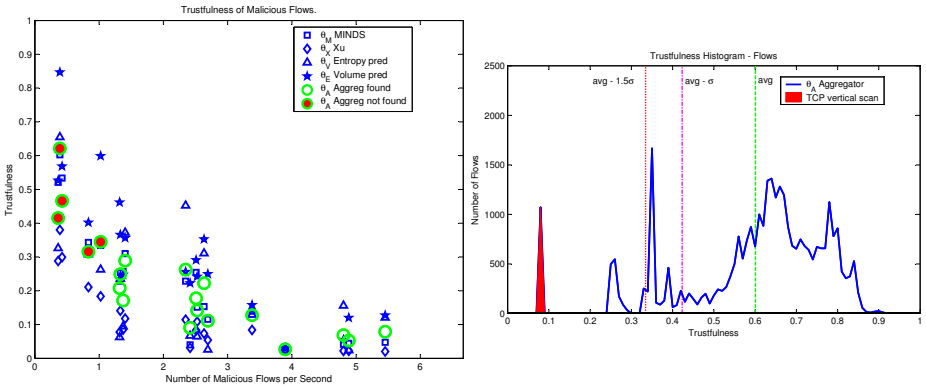

Fig. 4. Trustfulness assigned to vertical scans in function of attack intensity in the set Φ_j (*left*), with a specific example of one attack's classification in the trustfulness histogram (*right*).

In Fig. 4, we can see the trustfulness assigned by individual agents and the whole system to the series of vertical scans that we have launched, in function of the number of flows during the observation period. The scans vary by approach (TCP SYN scan, UDP scan, profiling), port ordering and other parameters, but the results are relatively consistent. Larger scans are reliably detected, opinions of individual agents are relatively consistent, and the aggregation results detect all scans with the with more than 1 flow per second (averaged during the whole period). In this experiment, we say that the attack was detected if the scan flows are classified between the *avg-σ* threshold, as shown in the right section of Fig. 4.

		Anomalous					**Untrusted**				
		$A_{\mathcal{M}}$	$A_{\mathcal{X}}$	$A_{\mathcal{E}}$	$A_{\mathcal{V}}$	A_{Ags}	Θ_M	Θ_X	Θ_E	Θ_V	Θ_{Ags}
flows	detected	6653	3246	13541	12375	**9911**	9149	9975	10704	9518	**9741**
	TP	35	168	5841	5868	**4709**	5242	5712	5833	5864	**5769**
	FP	6618	3078	7700	6507	**5202**	3907	4263	4872	3654	**3972**
	FP[%] all	15.9%	7.4%	18.5%	15.6%	**12.5 %**	9.4%	10.2%	11.7%	8.8%	**9.5%**
srcIP	detected	72.5	322.3	17.2	16.7	**12.5**	7.8	11.3	13.5	10.8	**6.7**
	TP	1.7	0.2	2.5	2.7	**2.3**	2.7	2.7	2.3	2.7	**2.7**
	FP	70.8	322.1	14.7	14.0	**10.2**	5.1	8.6	11.2	8.1	**4.0**
	FP[%] all	1.52%	6.94%	0.31%	0.30%	**0.22 %**	0.11%	0.19%	0.24%	0.18%	**0.09%**

Table 1. Benchmark of anomaly models, aggregated anomaly and partial and final trustfulness. Averaged over 6 data sets, each with 5 minutes of traffic with about 40 000 flows in each dataset.

Once we have determined the system performance limits, we have also evaluated the system on a 30-minute snapshot of real-world traffic, including the activity of two zombie network nodes and one buffer overflow attack. Table 1

presents the performance of the system in terms of false positives/false negatives, evaluated for flows and distinct incident source IP addresses. We present the results for individual anomaly detection agents (MINDS: $A_{\mathcal{M}}$, Xu: $A_{\mathcal{X}}$, Entropy: $A_{\mathcal{E}}$ and Volume: $A_{\mathcal{V}}$, as defined in Section 3), aggregated anomalies $A_{\mathbb{M}}$ as defined by Eq. 1, trustfulness opinions of individual agents (Θ_M, Θ_X, Θ_V, Θ_E) and the final system output Θ_{Ags} defined by Eq. 7. We can see that even a simple aggregation of anomaly models provides far better results than any separate model (low FP values for $A_{\mathcal{E}}$ and $A_{\mathcal{V}}$ are caused by the fact that both models only consider significant sources of traffic $\sim 10\%$ of hosts). The use of trust modeling further improves the results by wide margin – we detect more attacks, with far less false positives. The difference is significant especially in number of detected traffic sources, where we have reduced the rate of false positives more than two fold compared to $A_{\mathbb{M}}$, while detecting the actual attacks more reliably. It shall be noted that the number of suspicious sources is a far better estimator of analysis effort, because of significant variability in incident size.

6 Related Work

The ideas presented in this paper are relevant to three artificial intelligence/computer science domains: trust modeling, pattern recognition and classification, and network intrusion detection. As we have already stated in Section 2, the trust models [10, 7] are specialized knowledge structures that excel in their area of specialization: learning from past observations of trustees behavior, integrating the reputation opinion (and knowledge about social structures) received from other agents and inferring reliable trust value, which can be then used for trusting decisions. In the field of network intrusion detection, they provide a very compelling set of features: they consider trustfulness of reputation [12, 23] or information sources [24], and integrate these methods with robust approaches to reputation integration [11, 8]. Recent models also concentrate on context representation [14, 13] or multidimensionality of trust [25].We currently use the trust model described in [20], which has an advantage of being iterative (i.e. does not hold the history of observations 1 - $A_{Ags}(\varphi_{i,j})$ for each r_k), and thus reduces the computational complexity of the solution.

The field of pattern recognition and classification [21] provides us not only with the formalism used to represent the traffic in the extended trust models, but directly addresses the problem of cooperative classification as such. In [26], the authors introduce a general framework that integrates a major part of previous work on classifier integration strategies, and this work is directly relevant to the last step of the algorithm introduced in Section 4. The integration of key concepts from the classification work into the trustfulness aggregation shall help us to further improve the system. Agent methods have already been suggested for similar purpose, in a cooperative person tracking domain [27].

The multi-agent approaches to intrusion detection problems are mostly used in host-based or hybrid host and network based IDS [28, 29], which perform part

of their sensing on the protected hosts. This allows them to detect several types of local malicious actions, such as suspicious API calls or anomalous application activity. In [28], Valeur *et al.* propose a general framework for alert correlation, which is able to integrate the data from several sensors, associate the activities that are part of single attack and distinguish typical sequences of attack actions. While our approach fits the general theoretical framework introduced in [28], it differs in many crucial aspects. All the sensors use different input data, integrated detection methods use XML-based IDMEF [30] as a common ontology, the events detected by individual methods are associated with each other using time windows, and the system puts a lot of emphasis on the detection of attack sequences specific to composite attacks. The method introduced in our work is based on the fact that all the agents use the same input data Φ_j and collaborate actively (by sharing anomalies) even before submitting their individual trustfulness values, which are subsequently combined.

7 Conclusion

This paper presents a very specific application of collaborative trust modeling in a highly competitive domain of network intrusion detection. The goal of our work is to improve the results provided by state-of-the-art, but still imperfect anomaly detection algorithms by an addition of overlay layer based on extended trust modeling and simple reputation mechanism. The method is able to robustly identify the significant network-level events (scans, Denial of Service attacks, worms, peer-to-peer networks) in the traffic and present them to human supervisor for inspection in a dedicated visualization agent [31].

The application of agent techniques to the problem of network behavior analysis showed that trust modeling techniques distributed over a dynamic community of detection agents can significantly improve the quality of results. Simple integration of anomaly detection algorithms reduces the error rate significantly, but most of the benefit is in the overlay trusting layer, which *reduces the rate of false positives and false negatives simultaneously* and therefore shows the added value that the trust modeling techniques can bring to the highly competitive field of network intrusion detection.

Acknowledgment: This material is based upon work supported by the European Research Office of the US Army under Contract No. N62558-07-C-0001. 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 European Research Office of the US Army. Also supported by Czech Ministry of Education grants 1M0567, 6840770038 (CTU) and 6383917201 (CESNET).

References

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

2. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. **13** (1987) 222–232
3. Cisco Systems: Cisco IOS NetFlow. http://www.cisco.com/go/netflow (2007)
4. Čeleda, P., Kováčik, M., Koníř, T., Krmíček, V., Špringl, P., Žádník, M.: FlowMon Probe. Technical Report 31/2006, CESNET, z. s. p. o. (2006) http://www.cesnet.cz/doc/techzpravy/2006/flowmon-probe/.
5. Lazarevic, A., Ertöz, L., Kumar, V., Ozgur, A., Srivastava, J.: A comparative study of anomaly detection schemes in network intrusion detection. In: Proceedings of the Third SIAM International Conference on Data Mining. (2003)
6. Bragg, R., Rhodes-Ousley, M., Strassberg, K.: Network Security; The Complete Reference. McGraw-Hill Osborne Media (2004)
7. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artif. Intell. Rev. **24** (2005) 33–60
8. Sabater, J., Sierra, C.: Reputation and social network analysis in multi-agent systems. In: Proceedings of AAMAS '02, Bologna, Italy (2002) 475–482
9. Ramchurn, S., Jennings, N., Sierra, C., Godo, L.: Devising a trust model for multi-agent interactions using confidence and reputation. Applied Artificial Intelligence **18** (2004) 833 – 852
10. Castelfranchi, C., Falcone, R.: Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In: Proceedings of the 3rd International Conference on Multi Agent Systems, IEEE Computer Society (1998) 72
11. Josang, A., Gray, E., Kinateder, M.: Simplification and analysis of transitive trust networks. Web Intelligence and Agent Systems **4** (2006) 139–162
12. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An integrated trust and reputation model for open multi-agent systems. Journal of Autonomous Agents and Multi-Agent Systems **13** (2006) 119–154
13. Rehak, M., Pechoucek, M.: Trust modeling with context representation and generalized identities. In: Cooperative Information Agents XI. Number 4676 in LNAI/LNCS, Springer-Verlag (2007)
14. Rettinger, A., Nickles, M., Tresp, V.: Learning initial trust among interacting agents. In Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L., eds.: Cooperative Information Agents XI, CIA 2007, Delft. Volume 4676 of Lecture Notes in Computer Science., Springer (2007) 313–327
15. Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P.N., Kumar, V., Srivastava, J., Dokas, P.: MINDS - Minnesota Intrusion Detection System. In: Next Generation Data Mining, MIT Press (2004)
16. Xu, K., Zhang, Z.L., Bhattacharrya, S.: Reducing Unwanted Traffic in a Backbone Network. In: USENIX Workshop on Steps to Reduce Unwanted Traffic in the Internet (SRUTI), Boston, MA (2005)
17. Lakhina, A., Crovella, M., Diot, C.: Diagnosis Network-Wide Traffic Anomalies. In: ACM SIGCOMM '04, New York, NY, USA, ACM Press (2004) 219–230
18. Lakhina, A., Crovella, M., Diot, C.: Mining Anomalies using Traffic Feature Distributions. In: ACM SIGCOMM, Philadelphia, PA, August 2005, New York, NY, USA, ACM Press (2005) 217–228
19. Rehak, M., Pechoucek, M., Bartos, K., Grill, M., Celeda, P.: Network intrusion detection by means of community of trusting agents. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007 Main Conference Proceedings) (IAT'07), Los Alamitos, CA, USA, IEEE Computer Society (2007)
20. Rehák, M., Lukáš Foltýn, Pěchouček, M., Benda, P.: Trust Model for Open Ubiquitous Agent Systems. In: Intelligent Agent Technology, 2005 IEEE/WIC/ACM International Conference. Number PR2416 in IEEE (2005)

21. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. 2nd edn. John Wiley & Sons, New York (2001)
22. Lyon, G.: Nmap. (http://insecure.org/nmap/)
23. Yu, B., Singh, M.P.: Detecting deception in reputation management. In: AAMAS '03, ACM Press (2003) 73–80
24. Barber, K.S., Kim, J.: Belief revision process based on trust: Agents evaluating reputation of information sources. In: Trust in Cyber-societies. Volume 2246 of Lecture Notes in Computer Science., Springer (2001) 73–82
25. Vu, L.H., Aberer, K.: A probabilistic framework for decentralized management of trust and quality. In Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L., eds.: Cooperative Information Agents XI, CIA 2007, Delft. Volume 4676 of LNCS., Springer (2007) 328–342
26. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. **20** (1998) 226–239
27. Meshulam, R., Reches, S., Yarden, A., Kraus, S.: Mlbp: Mas for large-scale biometric pattern recognition. In: AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, ACM Press (2006) 1095–1097
28. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A.: A comprehensive approach to intrusion detection alert correlation. IEEE Transactions on Dependable and Secure Computing **01** (2004) 146–169
29. Shyu, M.L., Quirino, T., Xie, Z., Chen, S.C., Chang, L.: Network intrusion detection through adaptive sub-eigenspace modeling in multiagent systems. ACM Trans. Auton. Adapt. Syst. **2** (2007) 9
30. : IETF Intrusion Detection Message Exchange Format. http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-16.txt (2007) Accessed in January 2007.
31. Rehak, M., Pechoucek, M., Celeda, P., Krmicek, V., Moninec, J., Dymacek, T., Medvigy, D.: High-performance agent system for intrusion detection in backbone networks. In: Cooperative Information Agents XI. Number 4676 in LNAI/LNCS, Springer-Verlag (2007)