Network Intrusion Detection by Means of Community of Trusting Agents

Martin Rehak, Michal Pechoucek, Karel Bartos, Martin Grill
Department of Cybernetics, Center for Applied Cybernetics
Czech Technical University in Prague
{mrehak,pechouc,bartosk,grillm}@labe.felk.cvut.cz

Pavel Celeda
Institute of Computer Science
Masaryk University
{celeda}@ics.muni.cz

Abstract

We apply advanced agent trust modeling techniques to identify malicious traffic in computer networks. Our work integrates four state-of-the-art techniques from anomaly detection, and combines them by means of extended trust model. Deployment of trust model ensures interoperability between methods, allows cross-correlation of results during various stages of the detection and ensures efficient evaluation of current traffic in the context of historical observations. The goal of the system, which is designed for online monitoring of high-speed network, is to provide efficient tool for targeted runtime surveillance of malicious traffic by network operators. We aim to achieve this objective by filtering out the non-malicious (trusted) part of the traffic and submitting only potentially malicious flows for subsequent semi-automatic inspection.[1]

1 Introduction

This work presents a reasoning core of a high-performance network-based intrusion detection system designed for deployment on high-speed backbone links of corporate networks and internet. The whole system consists of several layers, each operating on different speed: the lowest level, *traffic acquisition layer* must work on wire speed, perform packet inspection and traffic statistics aggregation. This contribution presents the second, *detection layer*, designed as a multi-agent system, where individual agents perform anomaly detection using state-of-the-art approaches and process the traffic anomaly data using extended [10] trust modeling methods [13]. In the top layer, we deploy an advanced user interface agent that allows the analyst to efficiently explore the significant anomalies identified by multi-agent system in the detection layer.

The overall "business" goal of the system (presented in Section 2) is to enable cost-efficient surveillance of anomalous traffic in high-speed networks. Combination of vast amount of transmitted data with relatively high false-positive rate of existing anomaly detection techniques makes their efficient deployment almost impossible; the number of false alarms, together with relatively big amount of time required for incident analysis discourages the administrators. We will introduce existing anomaly detection techniques that we use in Section 3.1, discuss the details of trust modeling algorithm in Section 3.2, and terminate with simple experimental evaluation.

2 System Architecture

As we have stated above, the whole system is composed of three layers, roughly defined by required processing speed and used technologies. This section will provide a very brief overview of all the layers, in order to present the specialized techniques of detection layer in context of the whole system.

The main purpose of the **traffic acquisition** layer is to analyze the data transmitted on the network, while gathering the real time statistics about current network flows. Software or hardware probes gather NetFlow [2] compatible data from one or more probes on the network, possibly with transformations and pre-processing applied by dedicated *Collectors*, essentially a high performance, domain-specific databases [6]. NetFlow format was initially introduced by Cisco [2] to gather statistical information about the nature of flows in the network.

Each network *flow* is an unidirectional component of TCP connection or UDP transmission – packets that share the same source IP address (srcIP), destination IP address (dstIP), source Port (srcPrt), destination Port (dstPrt) and protocol (TCP, UDP, ICMP or others) [15] are considered as a single flow. For each such flow, the probes gather important characteristics – the number of packets/bytes transmitted, aggregated values of TCP flags, transmission speed,

[1]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 and 6840770038.

duration, and other characteristics (see Table 1 for the features used in the system). Besides these observed values, we compute a set of aggregates over the whole set of flows acquired during one acquisition timeframe – presented anomaly detection methods deployed in detection agents rely on flow counts, traffic volumes and entropies computed over relevant subsets of all flows. This layer is based on several standard software components: modified NetFlow collector [6], complemented by advanced programable traffic acquisition cards [1].

Traffic acquisition layer provides the data to **detection layer** periodically, at the end of each observation period. The goal of the detection process (elaborated in Sec. 3) is to identify potentially malicious flows and to report these flows to **user interface layer** for analysis and possible reaction. User interface layer also gathers supplementary data from third party systems to facilitate the analysis, using `DNS` servers, `whois` and other standard protocols. Both the detection and user-interface layers are implemented in A-Globe multi-agent platform [14].

3 Cooperative Detection Process

At the end of each observation period, the detection layer receives a snapshot of network flows transmitted during the last observation period j. These flows form a set denoted Φ_j, and i-th flow of the set is denoted $\varphi_{i,j}$. The set Φ_j is received by all detection agents almost simultaneously, and the agents start processing of flows to extract anomalies.

We assume that there are several detection agents in the system, denoted $A, B, C, D, ...$, that belong to set *Ags*. Each agent performs two principal roles: it uses its own anomaly detection algorithm to detect anomalies in current dataset and shares these anomalies with other agents from the set *Ags*. In order to infer the anomaly $A_A(\varphi_{i,j,}\,\Phi_j)$ of the flow $\varphi_{i,j}$, as assessed by agent A, we use a selection of suitable existing anomaly detection techniques. Please note that some formulas can be simplified by dropping the explicit agent reference if there is no risk of ambiguity.

In the second stage of processing, presented in Section 3.2, the agents use the anomaly data to update their trust models and infer the conclusions regarding the maliciousness of current traffic flows.

3.1 Anomaly Detection Techniques

This work is based on four existing anomaly detection techniques, with smaller or bigger modifications that were necessary to adapt the technique to our use-case and trust framework. We actually use two components of each technique: **flow representation**, either with directly observed features, or using an underlying model, and **anomaly assessment**, a function that provides mapping between flows and perceived level of their anomaly.

Feature	Description
srcIP	Source IP Address.
dstIP	Destination IP
srcPrt	Source Port
dstPrt	Destination Port
Protocol	Protocol (TCP/UDP/ICMP)
Packets	Number of Packets in the Flow
Bytes	Number of Bytes Transferred

Table 1. NetFlow features subset that defines the identity of each flow $\varphi_{i,j}$, unique over each specific NetFlow aggregation period.

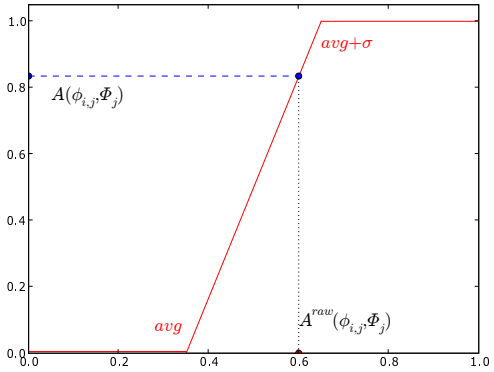

Figure 1. Normalization of raw anomaly $A^{raw}(\varphi_{i,j,}\,\Phi_j)$ to final value $A(\varphi_{i,j,}\,\Phi_j)$ by intersection with function A_{norm}.

Before the presentation of four state-of-the-art anomaly detection techniques, we will address one of the principal problems of their interoperability. Depending on the context of deployment and applied method, the range of returned anomaly values can differ significantly – the value that is highly anomalous when returned by one method can be completely normal for another. To address this problem, we will apply a normalization step which will turn method-specific anomaly assessment into a universal value relative to the past traffic, which is shared by all agents.

To perform the conversion, the agents maintain a normalization function defined by the data: more specifically the average *avg* and standard deviation σ of anomalies. These values define normalization function A_{norm}, which is shown in Figure 1. A_{norm} is defined as a fuzzy interval on the same support as anomaly values. It is zero under *avg*, linearly grows to 1, which is reached at $avg + \sigma$. Final anomaly values $A(\varphi_{i,j,}\,\Phi_j)$ are obtained as the value of $A_{norm}(A^{raw}(\varphi_{i,j,}\,\Phi_j))$ for the untransformed anomaly value $A^{raw}(\varphi_{i,j,}\,\Phi_j)$, as shown in Figure 1.

The **MINDS** system [5] represents the flow by basic NetFlow aggregation features (srcIP, srcPrt, dstIP, dstPrt, protocol) and complements them by the number of the flows from

the same srcIP, to the same dstIP and their combinations with dstPrt and srcPrt respectively. These properties are assessed both in time and number of connections-defined windows, to account for slow scanning. However, in our implementation, we have simplified the technique considerably to be able to integrate it with the remainder of our solution. We only observe time window-defined features, and we don't use the anomaly detection stage based on advanced outlier factor modeling. The original system was not designed for backbone networks, and this technique does not scale well enough for our purposes (as claimed in [5]). Therefore, we use a simple comparison with aggregated history data to determine the anomaly of each flow for each of the characteristics listed in Table 2. Computed anomalies are then normalized (see below) and combined into one anomaly coefficient per flow.

In order to determine the anomaly of the individual flow $\varphi_{i,j}$ with respect to each of the four individual dimensions listed in Table 2, we retrieve the number of flows corresponding to to the selection criteria – for example having their srcIP equivalent with the $\varphi_{i,j}$: $\{\Phi_j|\varphi_{i,j}[srcIP]\}$. We retrieve the history value for the same attribute, and find the difference[2]. This difference is then transformed into $[0, 1]$ interval, and undergoes normalization process described above (Figure 1). The history value is updated using a very simple floating average approximation - we average the current history value with the actual observation, with 0.8 and 0.2 weight applied respectively. This process allows us to follow natural activity trends, and in the same time identifies rapid surges of traffic. The value $A_{\text{MINDS}}(\varphi_{i,j,}\,\Phi_j)$ is then determined as an aggregation of all four marginal anomalies of relevant to the flow $\varphi_{i,j}$.

The system proposed by **Xu** *et al.* [16] for traffic analysis on backbone links also uses the NetFlow based identity 5-tuple. The context of the single connection is defined by the normalized entropy of srcPrt, dstPrt and dstIP dimensions (see Table 2) of the set of all connections from the set $\{\Phi_j|\varphi_{i,j}[srcIP]\}$ (sharing the srcIP of the flow $\varphi_{i,j}$ in the current flow set Φ_j). As the original work was intended for deployment on backbone links, our implementation is much closer to original than in the MINDS case discussed above. There is no need for history management, as we only use the current flow and data set to determine the anomaly.

More formally, we identify a set $\{\Phi_j|\varphi_{i,j}[srcIP]\}$ as in one of the MINDS cases, and subsequently determine the entropy of the dstIP, srcPrt and dstPrt of the set. In the next step, entropy is normalized by dividing it with the $log(|\{\Phi_j|\varphi_{i,j}[srcIP]\}|)$, in order to obtain a number between 0 and 1, denoted as relative uncertainty. Once we have the uncertainties, we use a set of rules on the relative uncertainty values to identify the hosts (and consequently flows) with normal, server and exploit (e.g. anomalous) traffic characteristics, and we assign the anomaly values according to these rules [16]. Once we perform the normalization of results, we obtain a set of $A_{\text{XU}}(\varphi_{i,j,}\,\Phi_j)$ values for further processing.[3]

Feature	Description
count-dst	Number of flows to unique destinations from the $\varphi_{i,j}$ source IP.
count-src	Number of flows from the unique sources toward the same destination as $\varphi_{i,j}$.
count-serv-src	Number of flows from the $\varphi_{i,j}$ source IP to the $\varphi_{i,j}$ destination port.
count-serv-dst	Number of flows to the same destination IP as $\varphi_{i,j}$ using the same source port as $\varphi_{i,j}$.

Feature	Description
ent-dstIP	Entropy of destination IP address over the set of all the flows from the $\varphi_{i,j}$ source IP.
ent-srcPrt	Entropy of source port over the set of all the flows from the $\varphi_{i,j}$ source IP.
ent-dstPrt	Entropy of destination port over the set of all the flows from the $\varphi_{i,j}$ source IP.

Table 2. Context features extracted by our (simplified) implementation of MINDS [5] algorithm (top) and Xu et al. algorithm [16]. for each flow $\varphi_{i,j}$ from the set Φ_j.

Two methods adopted from the work of **Lakhina** *et al.* share several common properties: they were originally designed for detection of anomalous *origin-destination flows*, defined as a sum of all flows arriving from one network and leaving to another network. In order to identify the anomalous behavior, both variants use *Principal Component Analysis* (PCA) to model the normal traffic, using the past flow sets $\{\Phi_{j-k}...\Phi_{j-1}\}$ to predict relavant features: traffic volumes [7] and entropies [8]. Differences between predicted and observed values are then used to identify the anomalies.

However, our deployment and intention differs from original publication. We don't need to identify problematic networks, but rather well defined flows or their groups. Furthermore, we don't deploy several simple sensors on an interconnected mesh of links, but a full-fledged (and possibly hardware accelerated) NetFlow sensors working on individual flow level. In order to use the PCA, we need to aggregate the observed flows into larger, predictable entities. For simplicity, we aggregate the flows by srcIP, predict the total number of packets of these flows (from the set $\{\Phi_j|\varphi_{i,j}[srcIP]\}$) and compare the prediction with actual observations in Φ_j. We are currently investigating other aggregation possibilities, like subnets, protocol, ports and var-

[2]Due to the efficiency considerations, we actually only work with flows from aggregated groups with more than three flows. This restriction theoretically opens a window for slow scanning techniques, but the limitation is not sever as such techniques would be difficult to discover by our time-window dependent implementation of the algorithm.

[3]In another work of the same authors [17], the method is applied on wider selection of data, aggregated also by dstIP and ports.

ious combinations.

In the baseline work, Lakhina *et al.* [7] uses the above described general approach to model the volumes of traffic. In our instance, it means predicting the number of all packets from the srcIP where the flow $\varphi_{i,j}$ has originated. Technically, we transform each set Φ_j into a single vector, which contains the number of packets originating from each srcIP. Agent maintains a PCA model which is defined as a transformation operator (*i.e.* matrix). Agent subtracts the normal (PCA predicted) component of the traffic from the observed vector of packet counts aggregated by srcIP, and leaves a residual (potentially anomalous) packet count for each source IP. This number is then transformed into the $[0, 1]$ interval, normalized and returned as $A_{\mathrm{LV}}(\varphi_{i,j}, \Phi_j)$ for each flow $\varphi_{i,j}$ - note that its value is identical for all flows from the same IP address. Due to the statistical nature and high computational requirements of the PCA method (the transformation matrix is data dependent and must be updated periodically), we only model the srcIP addresses with more than 20 originating packets. We argue that this threshold is comfortably below the modeling capabilities of PCA in our case.

In our other approach, based on another work of the same authors [8], the PCA method is used to model the normal and residual entropy of the destination IP and source and destination ports of all the flows originating from the $\varphi_{i,j}$ srcIP. Subtracting the modeled values from the observed data subtracts normal traffic entropies and allows us to concentrate on anomalous residual entropy. Clustering then also emphasizes the residual traffic, as the residual values of entropies are used to define the context of the flow. Technically, the approach is analogous to the previous method, the only difference is in the fact that we model three parameters instead of one, and that these parameters are based on entropies, rather than volumes.

3.2 Trusting Agent Algorithm

Trust modeling intervenes in the second stage of Flow processing, when the agents have finished their anomaly assessment and have shared their opinions regarding flow anomaly with each other. The application of trust modeling to anomaly detection follows three main goals: (i) to provide a uniform history management mechanism for anomaly detection methods, (ii) to combine the data produced by anomaly detection algorithm, and to (iii) normalize the decision regarding the anomaly of individual flows.

Each agent maintains its own trust model, based on trust management of significant reference elements (called **reference idexts** as defined below) in the Identity-Context spaces of flow features, rather than on direct attachment of trustfulness to individual observed flows. There are two principal reasons for this design; working with flow classes represented by reference idexts allows natural generalization, and consequently the reuse of trustfulness data between similar flows in different batches Φ_j, Φ_{j+k}, and it also significantly improves the efficiency of the mechanism. Authors of the MINDS system even directly claim that working with complete history of flows (albeit in different manner in their case) is not feasible for large flow sets [5].

In order to identify classes of flows and associated reference elements, we have analyzed available classification methods [4] and identified necessary requirements on flow representation, as well as appropriate clustering method.

In order to represent each flow $\varphi_{i,j}$ in the Identity-Context space [12] of agent A, we need to identify its associated **Idext**, a formal representation of **ide**ntity and conte**xt** $ix_A(\varphi_{i,j})$. Idexts are the elements of Identity-Context space $\mathbb{IC}$, which fulfills the properties of metric space. $\mathbb{IC}$ definition differs between agents, following the anomaly detection method used to define the context of each flow. Identity features (see Table 1) are shared by all agents, while the agent-specific features are discussed in Section 3.1.

Identity and context features are used to define the metrics of the space $\mathbb{IC}$, which describes similarity between flows (and their idexts) as perceived by the evaluating agent. Metrics shall comply with its mathematical properties[4]. Distance function is used during trust *model update* and *query* to determine existing **reference idexts** relevant to the current flow (*i.e.* within short distance d), determine the weight of each reference idext with respect to this observation and perform a query or update operation for each reference idext within domain-dependent distance limit.

Trust model is formally represented as a fuzzy set [9] Θ of reference idexts r_k, where the membership $\Theta(r_k)$ of each reference idext r_k in the set Θ determines its level of trustfulness. The set of all reference idexts is denoted $\mathcal{R}$. When evaluating the trustfulness, it is necessary to include the quality of trustfulness data in the reasoning. Such aspects include consistency of observations relevant to each reference idext r_k, their number and possibly other factors. We address this problem by making the set Θ a type-2 fuzzy set, where the membership function $\Theta(r_k)$ does not return a real number, but a fuzzy number [3] in the $[0, 1]$ interval. The width of the fuzzy number increases with uncertainty [11], and the use of Mamdani inference [9] in trusting assessment includes this information in the decision.

Formally, when an agent A updates its trust model with flow $\varphi_{i,j}$, it aggregates the anomaly assessments $A_X(\varphi_{i,j}, \Phi_j)$ from all agents X in the set *Ags*. It can use a wide range of aggregation function - from $\min$, through average to $\max$, depending on operator preference between false positives/false negatives. However, at this level, we

[4] Any distance function/metrics $d : \mathbb{IC} \times \mathbb{IC} \rightarrow R$ must respect following properties: **non-negativity**: $d(c_1, c_2) \geq 0$, **symmetry**: $d(c_1, c_2) = d(c_2, c_1)$, **zero distance $\Leftrightarrow$ identity**:$d(c_1, c_2) = 0 \Leftrightarrow c_1 = c_2$, **triangle inequality**: $d(c_1, c_3) \leq d(c_1, c_2) + d(c_2, c_3)$.

prefer functions closer to max (*i.e.* increasing false positives and limiting false negatives), because false positives can be addressed in the subsequent stages of mechanism. Aggregation returns anomaly estimate $A(\varphi_{i,j}, \Phi_j)$, a real number in $[0, 1]$ interval, and we use $\tau_{i,j} = 1 - A(\varphi_{i,j}, \Phi_j)$ as a trustfulness observation for the flow $\varphi_{i,j}$.

With the flow idext $ix_A(\varphi_{i,j})$ and its aggregated anomaly $A(\varphi_{i,j}, \Phi_j)$, we use the following formula (Eq. 1) to update the trustfulness $\Theta(r_k)$ of relevant reference context(s) r_k. Note that d denotes a distance function, and weight w is a monotonously non-increasing function of distance between $ix_A(\varphi_{i,j})$ and r_k, defined as $w_k = e^{-2d(ix_A(\varphi_{i,j}), r_k)}$ and W_k is an aggregate weight of all past observations used for $\Theta(r_k)$ update.

$$\Theta'(r_k) = \mathit{WeAg}((\Theta(r_k), W_k), (\tau_{i,j}, w_k)) \quad (1)$$

The operator *WeAg* performs an update of the data that determine the shape of the fuzzy number $\Theta(r_i)$, as defined in [11]: average, average of squares (for σ estimate using the formula $\sigma^2(X) = \mathit{avg}(X^2) - \mathit{avg}^2(X)$), min and max. Note that first two aggregations (*avg* and avg^2) in Eqs. 2-5 are weighted, while the min/max are evaluated regardless of the weights.

$$\Theta'_A(r_k)[avg] = \frac{W_k \cdot \Theta_A(r_k)[avg] + w_k \cdot \tau_{i,j}}{W_k + w_k} \quad (2)$$

$$\Theta'_A(r_k)[avg^2] = \frac{W_k \cdot \Theta_A(r_k)[avg^2] + w_k \cdot \tau_{i,j}^2}{W_k + w_k} \quad (3)$$

$$\Theta'_A(r_k)[\mathit{min}] = \min(\Theta_A(r_k)[\mathit{min}], \tau_{i,j}) \quad (4)$$

$$\Theta'_A(r_k)[\mathit{max}] = \max(\Theta_A(r_k)[\mathit{max}], \tau_{i,j}) \quad (5)$$

During the query phase of the model, we use an analogous method to construct a fuzzy number representing the trustfulness of the flow $\varphi_{i,j}$. Once we construct its idext $ix_A(\varphi_{i,j})$, we can determine $\Theta_A(ix_A(\varphi_{i,j}))$ by aggregating its value from the Θ values over the reference contexts:

$$\Theta_A(ix_A(\varphi_{i,j})) = WeAg_{r_k \in \mathcal{R}}(\Theta_A(r_k), w_k) \quad (6)$$

Detailed equations for each component of the fuzzy number $\Theta_A(ix_A(\varphi_{i,j}))$ are analogous to Eqs. 2-5. The aggregation process is also highlighted in Figure 2.

3.2.1 Reference set update

Until now, we have not addressed an important question of reference context creation and maintenance: when and how to create/modify the reference contexts. In our past research, we have shown that the size of the set $\mathcal{R}$ is mainly an efficiency concern - when we place the reference contexts dense enough, the quality of the decisions provided by the model remains comparable [10]. However, the efficiency is one of the primary concerns in our domain, as

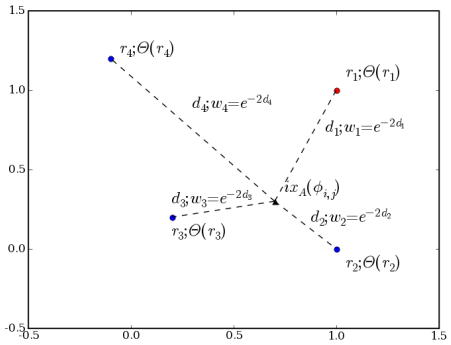

Figure 2. Model query: aggregation of trustfulness $\Theta_A(ix_A(\varphi_{i,j}))$ from reference idexts.

the size of the typical set Φ_j can easily reach millions of flows on gigabit links. Therefore, it is preferable to minimize the number of reference contexts, while keeping them close to the flow positions in the $\mathbb{IC}$. Another requirement on the method is high efficiency, ability to work without previous knowledge of target $\mathcal{R}$ size and the ability to work online, in the same time with Θ update. To comply with the requirements, we have selected a Leader-Follower algorithm [4], which has an additional advantage of simple implementation. Leader-Follower algorithm simply creates a new cluster (whose centroid is used as a reference idext r_k) if it encounters a new observation (*i.e.* flow idext) farther away from than a predetermined threshold from existing centroids. When the observation falls into an existing cluster, the position of cluster centroid might be updated, as outlined in Alg. 1.

Input: *flow,situat,trustObs*
closest ← *nil*;
mindist ← ∞;
$ix(\varphi_{i,j})$ ← identityCx(*flow*,*situat*); $\tau_{i,j} \leftarrow 1 - A(\varphi_{i,j}, \Phi_j)$
foreach $r_k \in \mathcal{R}$ **do**
  dist ← $d(r_k, ix(\varphi_{i,j}))$
  if *dist* < *mindist* **then**
    closest ← r_k
  $w_k = weight(dist)$
  if $w_k >$ *threshold* **then**
    $\Theta(r_k) = \mathit{WeAg}((\Theta(r_k), W_k), (\tau_{i,j}, w_k))$
end
if *mindist* > *clustsize* **then**
  $\mathcal{R}$.append($ix(\varphi_{i,j})$)
  $\Theta(ix(\varphi_{i,j})) = \mathit{WeAg}((\mathit{undef.}, 0), (\tau_{i,j}, w_k))$
else
  closest.updatePosition($ix(\varphi_{i,j})$)

Algorithm 1: Processing of the flow $\varphi_{i,j}$ into the model.

3.2.2 Trustfulness Assessment

As we can see in the algorithm, there are several parameters that can be altered to make tradeoffs between high locality (precision), and generalization ability and efficiency. The most important parameter is the clustsize, which directly influences the size of the set $\mathcal{R}$. When this parameter is set to 0, the number of clusters is identical to number of unique flows, and we have the largest model possible. When we depart from this extreme case and increase the distance, the size of the $\mathcal{R}$ decreases, computational and memory efficiency increases, but with possible impact on trust data quality if the trustfulness values exhibit strong local variations that can not be captured by coarse-grained model. This effect can be reduced by appropriate definition of $\mathbb{IC}$. The threshold parameter influences the generalization capacity of the model – higher we set the threshold, more local the effects of trust update become. Again, increasing threshold values can improve adaptation to local irregularities of trustfulness, but model learning becomes less efficient.

Even if the anomalies provided by anomaly detection parts of all agents undergo a normalization process to make their scales as method-independent as possible, we need to convert the trustfulness $\Theta_A(ix_A(\varphi_{i,j}))$ assessed by individual trusting agents into an agent and environment-independent value. To perform this conversion, we use the concept of self-trust (*i.e.* agent's estimate of mean performance in the current environment) [11]. Self-trust is defined by two fuzzy intervals on $[0,1]$ domain: High-trust HT_A has membership equal to 1 on the interval between average trustfulness of all flows before $\varphi_{i,j}$: $avg_{l<j \cup (l=j \wedge k<i)}(\tau_{k,l})$, and then decreases linearly to reach 0 in $avg_{l<j \cup (l=j \wedge k<i)}(\tau_{k,l}) - 2 \cdot \sigma_{l<j \cup (l=j \wedge k<i)}(\tau_{k,l})$. Low trust is defined as its complement, and together, they define a partitioning of unity on the trustfulness interval $[0,1]$. The above definition shows that LT_A and HT_A are constructed from the same data as the trustfulness of all flows, but regardless of their position. They are used to transform the trustfulness value $\Theta_A(ix_A(\varphi_{i,j}))$ into a degree of trust $DOT_A(\varphi_{i,j,}\,\Phi_i)$, and its complement, degree of distrust, $DOD_A(\varphi_{i,j,}\,\Phi_i)$ (both are real numbers). To perform the transformation, we use a Mamdani Inference [9][5]. To determine the degree of trust $DOT_A(\varphi_{i,j,}\,\Phi_i)$, we calculate the inference D_{min} between $\Theta_A(ix_A(\varphi_{i,j}))$ and HT_A (shown in Figure 3):

$$DOT_A(\varphi_{i,j,}\,\Phi_i) = hgt(\Theta_A(ix_A(\varphi_{i,j})) \cap_{min} HT_A) \quad (8)$$

and we obtain the degree of distrust $DOD_A(\varphi_{i,j,}\,\Phi_i)$ as:

[5] $$D_T(X^*, A_i) = hgt(X^* \cap_T A_i) \quad (7)$$
where T is a selected *t-norm* (see [9]).We have selected the Standard (Gödel, Zadeh) t-norm, defined as $T(A,B) = \min(A,B)$, and therefore, we can informally define the inference as a height of the intersection of two fuzzy numbers (intervals).

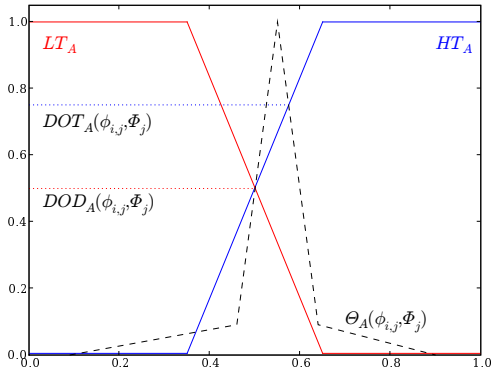

Figure 3. Obtaining $DOT_A(\varphi_{i,j,}\,\Phi_i)$ and $DOD_A(\varphi_{i,j,}\,\Phi_i)$ values from trustfulness $\Theta_A(ix_A(\varphi_{i,j}))$.

$$DOD_A(\varphi_{i,j,}\,\Phi_i) = hgt(\Theta_A(ix_A(\varphi_{i,j})) \cap_{min} LT_A) \quad (9)$$

Degrees of trust and distrust can be conveniently used to obtain a binary result regarding the trustfulness of the flow $\varphi_{i,j}$. Such flow would be considered as *trusted* iff

$$DOT_A(\varphi_{i,j,}\,\Phi_i) \geq DOD_A(\varphi_{i,j,}\,\Phi_i) \quad (10)$$

Note that the use of fuzzy arithmetics in the degree of trust assessment naturally incorporates the uncertainty (which is represented as a "width" of Θ values) of trust values into the decision.

4 Experimental Evaluation

Our experiments evaluate the ability of trusting agent to select a suitable subset of anomalous traffic for subsequent inspection by human operators. Therefore, we are interested by distribution of reference idexts over the trustfulness range, and the response of the system against the known threats. We have performed a series of experiments on live network to establish the characteristics of the system when it encounters a `TCP SYN` vertical scan against a single host using the `nmap` tool.

In Figure 4, we can see how the distribution of the currently observed flows $\varphi_{i,j}$ from the set Φ_j over the trustfulness range $[0,1]$ (x axis). The y axis is simply defined as the number of flows with given trustfulness $\Theta(\varphi_{i,j})$. In our graphs, we can observe that the traffic covers wide range of trustfulness values, and that the distributions of trustfulness from different agents are roughly in the same area of trust range, thanks to the normalization of anomalies introduced in Section 3.1. The major peak on the left side of the distribution, centered roughly on $\Theta = 0.12$, corresponds

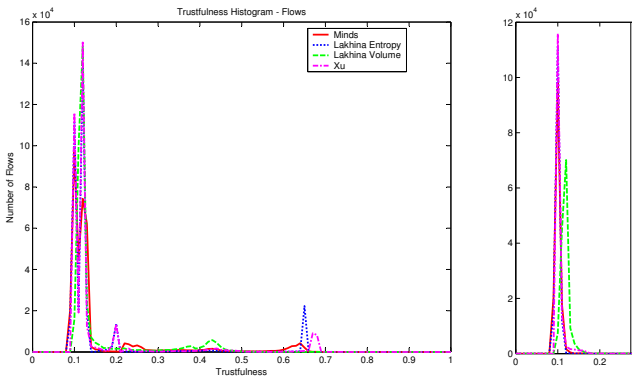

Figure 4. Distribution of traffic trustfulness in the traffic with major scan, with all traffic (left) and attack traffic only (right).

entirely to the attack traffic (which is restricted to this spot) – the histogram for attack traffic only is shown in the right-hand size of the figure. We should also point out that due to the small size of scanning packet payload, the scanning is barely visible when we follow the volume of data in bytes, becomes more apparent when we count packets and dwarfs the other traffic in number of flows.

Our simple example illustrates the ability of the proposed solution to separate malicious and normal traffic, and to single-out the malicious traffic for detailed inspection. More detailed and rigorous evaluation of system efficiency, including the estimate of false negatives/positives will be presented in a dedicated publication due to the lack of space.

5 Conclusion

This contribution presents a novel approach to cross-correlation of various anomaly detection methods by means of trust modeling, a specific security technique researched in the field of multi-agent systems. We present an extension of trust modeling technique with efficient flow identity and context representation, thus addressing the problems related to number of flows and their persistence. Differences in the anomaly values distribution of various trust anomaly detection methods is addressed using a normalization step, and the trustfulness values themselves are normalized again before the trusting decision, using the data-dependent low and high trust intervals.

The presented technique has been successfully validated in the real traffic, and our current work focuses on improvements in classification effectiveness, algorithm efficiency, robustness evaluation and other crucial aspects. We will also emphasize the cooperation between trusting agents in the later stages of decision-making process, and autonomous adaptation of the mechanism to current network traffic.

References

[1] CESNET, z. s. p. o. Family of COMBO Cards. http://www.liberouter.org/hardware.php, 2007.

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

[3] D. Dubois and H. Prade. Fuzzy Real Algebra:Some Results. *Fuzzy Sets and Systems*, 2(4):327–348, 1979.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2nd edition, 2001.

[5] 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.

[6] P. Haag. NfSen - NetFlow Sensor. http://nfsen.sourceforge.net/, 2007.

[7] 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.

[8] 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.

[9] W. Pedrycz and F. Gomide. *An Introduction to Fuzzy Sets : Analysis and Design*. Complex Adaptive Systems. Bradford Book, Cambridge, London, 1998.

[10] M. Rehak, M. Gregor, M. Pechoucek, and J. M. Bradshaw. Representing context for multiagent trust modeling. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006 Main Conference Proceedings) (IAT'06)*, pages 737–746, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[11] M. Rehák, Lukáš Foltýn, M. Pěchouček, and P. Benda. Trust Model for Open Ubiquitous Agent Systems. In *Intelligent Agent Technology, 2005 IEEE/WIC/ACM International Conference*, number PR2416 in IEEE, 2005.

[12] 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.

[13] J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artif. Intell. Rev.*, 24(1):33–60, 2005.

[14] D. Šišlák, M. Rehák, M. Pěchouček, M. Rollo, and D. Pavlíček. A-globe: Agent development platform with inaccessibility and mobility support. In R. Unland, M. Klusch, and M. Calisti, editors, *Software Agent-Based Applications, Platforms and Development Kits*, pages 21–46, Berlin, 2005. Birkhauser Verlag.

[15] W. Stallings. *Data and computer communications (5th ed.)*. Prentice-Hall, Inc., 1997.

[16] 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.

[17] K. Xu, Z.-L. Zhang, and S. Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *SIGCOMM '05*, pages 169–180, New York, NY, USA, 2005. ACM Press.