Reducing false positives of network anomaly detection by local adaptive multivariate smoothing

Article *in* Journal of Computer and System Sciences · April 2016

DOI: 10.1016/j.jcss.2016.03.007

3 authors, including:

Martin Grill
Czech Technical University in Prague
25 PUBLICATIONS **433** CITATIONS
SEE PROFILE

Tomás Pevný
Czech Technical University in Prague
73 PUBLICATIONS **3,059** CITATIONS
SEE PROFILE

Reducing False Positives of Network Anomaly Detection by Local Adaptive Multivariate Smoothing

Martin Grill[a,b], Tomáš Pevný[a,b], Martin Rehak[a,b]

[a] *Czech Technical University in Prague, Faculty of Electrical Engineering*
[b] *Cisco Systems, Inc.*

Abstract

Network intrusion detection systems based on the anomaly detection paradigm have high false alarm rate making them difficult to use. To address this weakness, we propose to smooth the outputs of anomaly detectors by online Local Adaptive Multivariate Smoothing (LAMS). LAMS can reduce a large portion of false positives introduced by the anomaly detection by replacing the anomaly detector's output on a network event with an aggregate of its output on all similar network events observed previously. The arguments are supported by extensive experimental evaluation involving several anomaly detectors in two domains: NetFlow and proxy logs. Finally, we show how the proposed solution can be efficiently implemented to process large streams of non-stationary data.

Keywords: network anomaly detection, regression smoothing, false positive rate reduction

1. Introduction

Increasing number and sophistication of attacks against critical enterprise computing infrastructure drives the need to deploy increasingly more sophisticated defense solutions. An essential component of the defense are Intrusion Detection Systems (IDS) [1] analyzing network traffic that crosses the defense perimeter and looking for evidence of ongoing malicious activities (network attacks). When such activity is detected, an alarm is raised and then analyzed by network administrator who determines the existence and scope of the damage, repairs it and improves the defense infrastructure against future attacks.

IDS can be categorized according to different criteria: by their location (on a host, a wired network, or a wireless network), detection methodology (signature matching, anomaly detection, or stateful protocol analysis), or capability (simple detection or active attack prevention) [1]. In context of our work, the most important criterion is the categorization by detection methodology, which is described in more detail below.

Email addresses: `magrill@cisco.com` (Martin Grill), `tpevny@cisco.com` (Tomáš Pevný), `marrehak@cisco.com` (Martin Rehak)

Signature matching techniques identify attacks by matching packet contents against specific attack signatures. The signatures are created using already identified and well–described attack samples, which is time consuming and can take from couple of hours up to several days, which gives attackers plenty of time for their criminal activities. The biggest weakness of this solution is that it detects only known attacks, which can be due to smart evasion techniques used by malware limiting. With the growing proportion of encrypted traffic, use of self-modifying malware and other evasion techniques, the use of detection techniques tailored to catch predefined known set of attacks is becoming increasingly irrelevant.

Anomaly-based detection tries to decrease the human work (e.g. manual creation of signatures) by building statistical model of a normal behaviour and detect all deviations from it. This enables to detect new, previously unknown attacks provided that their statistical behaviour is different from that of the normal traffic. While anomaly-based methods are attractive conceptually, they have not been widely adopted. This is because they typically suffer of high false alarm rate (not every anomaly is related to the attack) rendering them useless in practice, since network operator can analyse only few incidents per day [2, 3]. Decreasing the false positive rate is therefore important to make anomaly-based IDS competitive with signature matching based solutions.

Rehak [4] divides false positives of anomaly-based IDS into two classes: *unstructured* false positives are essentially a random noise caused by the stochasticity of the network traffic and *structured* false positives caused by a persistent but a very different behavior of a small number of network hosts, for example mail or DNS servers (the precise definition is left to Section 2).

This work proposes a method designed to decrease the rate of unstructured false positives by smoothing anomaly values with respect to time. This causes similar anomalies[1] occurring at different times to receive similar anomaly score, even though one would be otherwise flagged as anomaly. The rate of structured false positives is either decreased or remains the same, which depends on circumstances discussed in detail in Section 3. The method is evaluated with two different anomaly detection based IDSs, in both cases improving their accuracy.

The contribution of the paper is threefold.

- First, it mathematically formulates structured and unstructured false positives and argues why unstructured false positives are more difficult to white-list or remove.
- Second, it propose a theoretically sound method to decrease the rate of unstructured false positives.
- Third, it shows how the method can be implemented to efficiently process data-streams.

[1]Similarity can be arbitrary function $k : \mathcal{X} \times \mathcal{X} \mapsto [0, 1]$.

This paper is organized as follows. The next section first properly defines classes of false positives, describe the proposed solution and mathematically proofs its correctness under mild assumptions. The same section also discusses the modification to efficiently process data-streams. Section 4 shows, how the method was deployed in two Intrusion detection systems. The related work is discussed in Section 5 while Section 6 concludes the paper.

2. Classification of False Positives

Hereafter it is assumed that the network anomaly detection system (AD) observes a stream of network events (e.g., NetFlow [5], HTTP connections, etc.) produced by a set of hosts within the network. Anomaly detection system maintains internal model(s) to assign a score in range $[0, 1]$ to each observed event with zero indicating the normal event and one indicating possible attack, as it is assumed that malicious activities have statistical characteristics different from the normal ones [6, 7, 8] making them rare. As mentioned in the introduction the anomaly detection systems produce false alarms since an overwhelming majority of rare events are not caused by any attack. Rehak [4] divides these false positives into following two classes:

- *Unstructured false positives* are short-term events distributed uniformly over all the network hosts proportionally to the traffic volume. They are typically triggered by widespread, uniformly distributed behaviors (such as web browsing) and we model them as white noise (zero mean and finite variance) added to the anomaly detector's output. Therefore, the observed anomaly score y_i of an event x_i is equal to

$$y_i = g(x_i) + \eta_i, \tag{1}$$

  where $g(x_i)$ is the true anomaly score on event x_i and η_i is the additive white noise. The η_i therefore hides the true value $g(x_i)$.

- *Structured false positives* are caused by a (long-term) legitimate behaviors of a small number of network hosts. These behaviors are different from the background, and because they are found only at a very small portion of network hosts, they are reported as anomalies. Examples are rare applications performing software update, regular calls of unusual network APIs, etc. Since this type of false positive is typically limited to a small number of network hosts and its behavior is very regular, it can be quickly identified and filtered out using white-lists. Nevertheless, these white-lists are specific for a given network and are difficult to create before deployment. We define the structured false positives to be generated by a mixture of distributions

$$x_{sfp} \sim \frac{\sum_{j=1}^{m} \beta_j \epsilon_j(x_i)}{\sum_{j=1}^{m} \beta_j}, \tag{2}$$

  where ϵ_j represent structured false positive with weight β_j. Each component ϵ_j has small variance comparing to that of the unstructured false positives, but means of the components are typically far from each other.

3. Proposed Method

The idea behind the proposed local adaptive multivariate smoothing (LAMS) method is to replace the output of the anomaly detector by average anomaly score of similar events observed in the past, where the similarity between two events is defined as $K_h : \mathcal{X} \times \mathcal{X} \mapsto [0,1]$ (further also called *context*). This effectively smooths the output of the anomaly detector and therefore reduces unstructured false positives. This smoothing can be mathematically formulated as

$$\hat{g}_{\text{nw}}(x) = \frac{\sum_{i=1}^{n} K_h(x, x_i) y_i}{\sum_{i=1}^{n} K_h(x, x_i)}, \tag{3}$$

where $\{x_i\}_{i=1}^{n}$ is the set of observed network events, $\hat{g}_{\text{nw}}(x)$ is the expected anomaly of event x, and $\{y_i\}_{i=1}^{n}$ is the set of corresponding AD outputs. The space $\mathcal{X}$ on which it is defined can be arbitrary (e.g., space of all strings, graphs, etc.), but Gaussian kernel $K_h(x, x') = \exp\left(-\frac{\|x-x'\|^2}{h}\right)$ on Euclidean space is the most common combination (h parameterizes the width of the kernel). Estimator (3) is known as a Nadaraya-Watson estimator [9, 10] which is a non-parametric estimator of the conditional expectation of a random variable.

How does the above smoothing remove false positives according to the division introduced in previous section?

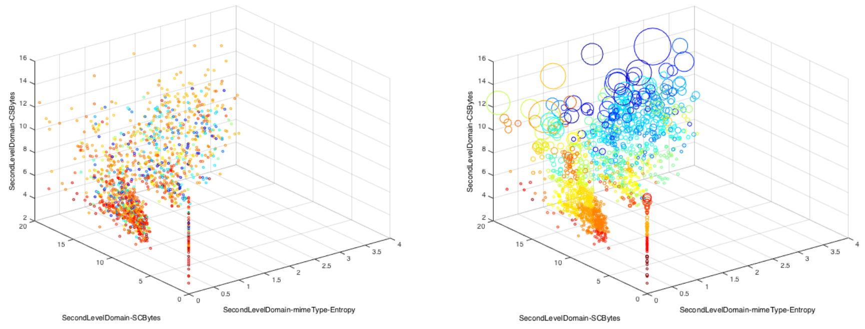
- *Unstructured false positives* are reduced by the Nadaraya-Watson estimator that performs local averaging of the events. Since the unstructured false positives are of the form $y_i = g(x_i) + \eta_i$, it is proven by Devroye *et al.* [11] that it converges to the true value $g(x_i)$ under fairly general assumptions. More formally, Devroye have shown

  $$E|\hat{g}_{\text{nw}}(x) - g(x)| \leq \frac{c}{\sqrt{nh}}, \tag{4}$$

  assuming h is chosen in terms of n, with $h \to 0$, $nh \to \infty$ as $n \to \infty$; $c > 0$ is a constant; the kernel K is absolutely continuous and differentiable, with $K' \in L^1$; g is differentiable and the $\text{Var}(y_i)$ is bounded. Since we are using a Gaussian kernel and the anomaly scores y_i are bounded to $[0,1]$ the estimate converges to the underlying true anomaly score $g(x)$ with a rate of $(nh)^{-1/2}$, when only unstructured false positives are present. However, setting the $h \to 0$ requires infinite computation and memory resources, therefore we restrict the estimator to fixed h.

  The smoothing effect is shown in the Figure 1 where the left figure shows the input to LAMS and right figure the output. It is apparent that the left figure is noisier, as points with different colors are close to each other.

- *Structured false positives* are long-term events confined to subset of network hosts without direct relationship to the background in the sense that AD's output on them does not change with the time. LAMS can remove them in following situations.



Scores of AD without LAMS models.   Scores of AD with LAMS models.

Figure 1: Visualization of score provided by HTTP anomaly detection engine (described in Section 4.2) without (left) and with (right) LAMS models in LAMS 3 context space (see Table 4 for details). Each point represents one HTTP request observed in the network, where its position is defined by the context features (defined in Table 4) and the color denotes the anomaly score, red being the most anomalous and blue the less. The larger the circle the more events share the same context position and anomaly score. The anomaly scores of the anomaly detection engine are shown in left hand side figure. As can be seen there are regions of the context space where most of the points have high anomaly scores and regions of low anomaly scores with additional noise. The right hand side figure shows the same points but the anomaly score is smoothed by the LAMS model effectively reducing the unstructured false positives via smoothing.

If LAMS' similarity measure of alerts K_h is different from that used in the anomaly detector, large number of events with normal AD score can be similar to structured false positives, which decreases the scores output by LAMS on false positives. Example of this type of behavior can be seen on the Figure 2, where we see that the amplitude of LAMS input is higher than that of the output.

If LAMS aggregates outputs of the same anomaly detector deployed on many networks, structured false positives can be normal in other networks. Then, LAMS aggregation eliminates the false positives by aggregating them with these normal activities.

In other situations, structured false positives are confined and there are no similar events receiving lower anomaly score. In this case LAMS fails to remove them, but does not increase their number.

3.1. Complexity Considerations

The complexity of the smoothing of AD's output described by Equation (3) is linear with respect to the number of observed events. This, together with the fact that all observed events need to be stored, makes LAMS useless for practical deployment, since number of network events can be as high as millions per second. We therefore resort to common approach to approximate (3) from

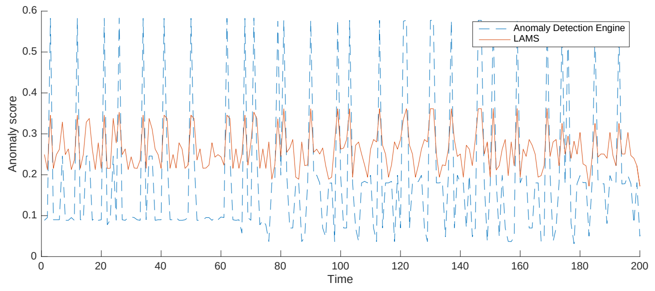


Figure 2: Anomaly score of the HTTP request to google web API in time. There are sudden spikes in AD's anomaly score (blue dashed line) that are reduced by the LAMS model (red solid line).

values maintained in a set of pivots $\Phi = \{\phi_j\}_{j=1}^J$, $\phi_j \in \mathcal{X}$ as

$$\hat{g}_{\text{lams}}(x) = \frac{\sum_{j=1}^J K_h(x, \phi_j)\hat{y}_j}{\sum_{j=1}^J K_h(x, \phi_j)}, \tag{5}$$

where $\{\hat{y}_j\}_{j=1}^J$ are estimates of $\hat{g}_{\text{nw}}(\phi_j)$ calculated according to (3). This reduces the computational complexity of the estimate in arbitrary x to $O(J)$, which is linear with the number of pivots and independent of the number of observed alerts[2]. The same holds for space complexity, because only positions ϕ_j of finite number of pivots and relevant estimates $\hat{y}_k$ needs to be kept. The set of pivots Φ is updated using the modified Leader-Follower algorithm [13], where new pivot is added to the set Φ when an event not similar to any pivot in Φ is received. The update process is described in more detail in Section 3.2.Contrary to the standard definition of Leader-Follower [13, 14] pivots are not allowed to move, because moving pivots towards areas of higher density causes forgetting of rare events, which we want to remember.

3.2. Incremental Update of the LAMS Model

Keeping LAMS model up to date on data-streams requires maintaining estimates $\{\hat{y}_j\}_{j=1}^J$ in $\{\phi_j\}_{j=1}^J$ and alternatively adding new pivot if needed. Upon arrival of a new observation of a network event (y_t, x_t), estimates $\hat{y}_j$ stored in pivots ϕ_j should be updated using the Equation (3) as

$$\hat{y}_j^{\text{new}} = \frac{\sum_{i=1}^n K_h(\phi_j, x_i)y_i + K_h(\phi_j, x_t)y_t}{\sum_{i=1}^n K_h(\phi_j, x_i) + K_h(\phi_j, x_t)}. \tag{6}$$

[2]This statement holds only partially, since with increasing number of observed samples the number of pivots will increase as well [12].

If we denote

$$w_j^{\text{old}} = \sum_{i=1}^{n} K_h(\phi_j, x_i), \tag{7}$$

$$\hat{y}_j^{\text{old}} = \frac{\sum_{i=1}^{n} K_h(\phi_j, x_i) y_i}{\sum_{i=1}^{n} K_h(\phi_j, x_i)}, \tag{8}$$

the Equation (6) can be rewritten as

$$\hat{y}_j^{\text{new}} = \frac{w_j^{\text{old}} \hat{y}_j^{\text{old}} + K_h(x_t, \phi_j) y_t}{w_j^{\text{old}} + K_h(x_t, \phi_j) y_t}. \tag{9}$$

Therefore, for efficient incremental update of the model only $\hat{y}_j$ and w_j need to be stored in each ϕ_j. The update can than be done using the following.

$$w_j^{\text{new}} = w_j^{\text{old}} + K_h(x_t, \phi_j) y_t, \tag{10}$$

$$\hat{y}_j^{\text{new}} = \frac{1}{w_j^{\text{new}}} \left[w_j^{\text{old}} \hat{y}_j^{\text{old}} + K_h(x_t, \phi_j) y_t \right], \tag{11}$$

The above can be further simplified to update only pivots close to x_t, which we define as pivots with similarity $K_h(x_t, \phi_j)$ greater than K_h^{min}. This reduces the complexity of the update process, because only limited number of nearest pivots needs to be updated with each new event.

The update is outlined in Algorithm 1. First, a set of all pivots in $\varepsilon(x_i)$ in the $K_h(x_t, \phi_j)$ vicinity of a new event (y_t, x_t) is found and their estimates $\hat{y}$ are updated as defined in Equation (11). The γ parameter controls the distance (threshold on the similarity) upon which a new pivot is created with $\phi_{J+1} = x_t$, $\hat{y}_{J+1} = y_t$ and $w_{J+1} = 1$.

The algorithm is controlled by several parameters allowing to trade high locality (precision) for generalization and efficiency. The most important parameter is γ, which influences the total number of pivots in LAMS and hence the cost of the algorithm. When this parameter is set to $\max_{x,x' \in \mathcal{X}} K_h(x, x')$, the number of pivots is identical to the number of unique events. Departing from this extreme case the number of pivots decreases with decreasing its value with a possible impact (both positive or negative) on the accuracy of estimates if local variations are strong. An example of its effect is shown in Figure 3. The minimal update weight K_h^{min} is not overly important and it is there more for computational speed-up, as sample's neighborhood are more controlled by the similarity function $K_h(x, x')$.

3.3. Querying the LAMS Model

The query of the model follows the Equation (5) with the difference that the estimate is computed only over similar pivots, controlled by the same parameter γ as used in the update. Formally the estimate $\hat{y}_t$ is calculated as

$$\hat{y}_t = \frac{\sum_{\phi_j \in \varepsilon(x_t)} K_h(x_t, \phi_j) y_j}{\sum_{\phi_j \in \varepsilon(x_t)} K_h(x_t, \phi_j)}, \tag{12}$$

```
**Data**: Stream of events $(y_i, x_i), i = 1 \dots$
**Result**: Set of pivots $\Phi = \{\phi_j = (x_j, \hat{y}_j)\}_{j=1}^{J}$
Start with an empty set $\Phi = \varnothing$;
**while** *there is a new event* $(y_i, x_i)$ **do**
    $\varepsilon(x_i) = \{\phi_k \in \Phi \,|\, K_h(x_i, \phi_k) > K_h^{\min}\}$;
    **for** $\phi_k \in \varepsilon(x_i)$ **do**
        Update $\hat{y}_k$ relevant to pivot $\phi_k$ using Equation (10) and (11);
    **end**
    Find the most similar pivot $\phi_N$ to the $x_i$:
    $\phi_N := \arg\max_{\phi \in \Phi} K_h(x_i, \phi)$;
    **if** $K_h(x_i, \phi_N) < \gamma$ **then**
        Create new pivot $\phi_{J+1} := x_i$;
        $w_{J+1} := 1$;
        $\hat{y}_{J+1} := y_i$;
        $\Phi = \Phi \cup \{\phi_{J+1}\}$;
    **end**
**end**
```

Algorithm 1: LAMS model update algorithm.

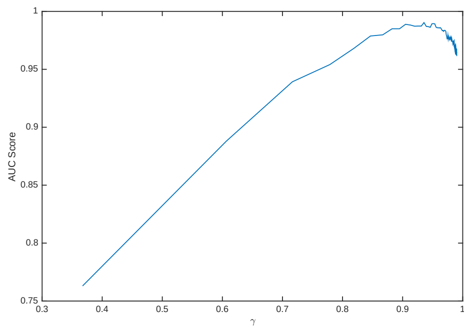


Figure 3: The graph shows effects of γ parameter on the average AUC score (see Section 4 for definition). Too large γ reduces effect of the smoothing causing smaller efficacy. On the other hand, too small γ decreases model's efficacy, since the model cannot capture local variations in the events.

where $\varepsilon(x_t) = \{\phi_j \in \Phi \,|\, K_h(x_i, \phi_k) > K_h^{\min}\}$. Restricting the estimate to be calculated from the neighborhood $\varepsilon(x_t)$ is mainly to be consistent with the update procedure. In practice, $K_h^{\min}$ is so small that the effect of points outside $\varepsilon(x_t)$ on the points would be negligible.

When LAMS is deployed, it always performs an update before the query, which means that there is always at least one pivot from which the estimate can be calculated.

3.4. LAMS Model Feature Selection

As in other machine learning algorithms the biggest influence on LAMS's accuracy has the selection of similarity measure and features defining it. The number of features that can be extracted from network traffic events can be high and defining the context with respect to all of them would lead to poor generalization — a phenomenon known as curse of dimensionality [15]. The usual approach is to reduce dimension by feature extraction algorithms [16] or by feature selection. In experiment described in the next section we have chosen the latter approach.

The features should be selected such that malicious network events are confined in a rather limited part of the space covered by only few pivots. This guarantees that such model will improve the accuracy, since if malicious events fall into a specific region of the feature space the long term anomaly estimate in this region will be consistently high. To select the features in practice, one either has to have examples of network attacks and use feature selection algorithm [17], or use a domain knowledge provided by an expert.

4. Experimental Evaluation

We have evaluated the effectiveness of the proposed LAMS model in two different anomaly detection engines: the first, described in Subsection 4.1, uses NetFlow [18] records while the second, described in Section 4.2, uses HTTP proxy logs. Both anomaly detection engines are based on ensemble of simple yet diverse anomaly detectors allowing to efficiently process large data-streams while providing good detection accuracy.

The accuracy is measured by the area under Receiver Operating Characteristic (AUC) [19], which is frequently used when the operating point of the detector is not known.[3] AUC is equal to the probability that the anomaly score of a randomly chosen malicious sample is higher than a randomly chosen legitimate one. The AUC score one means that the detector is always correct, zero means it is always wrong, and 0.5 means that its accuracy is equal to random guessing.

[3]By an operating point it is understood an upper bound on false positives / false negatives or their costs.

4.1. NetFlow Anomaly Detection

The NetFlow anomaly detection engine [20, 21] processes NetFlow [18] records exported by routers or other network traffic shaping devices. List of all features available in the NetFlow records can be seen in Table A.6 in Appendix. The anomaly detection engine identifies anomalous traffic using ensemble of anomaly detection algorithms, some of them based on Principal component analysis [22, 23, 24], some detects abrupt changes detection [8], and some even uses fixed rules [25]. Furthermore, there are detectors designed to detect specific type of unwanted behavior like network scans [26] or malware with domain generating algorithm [27]. In total the NetFlow anomaly detection engine uses 16 anomaly detectors. The rationale behind the ensemble is that (a) it decreases the computational requirements, since individual detectors and their internal models are simpler, and (b) errors of individual detectors cancels out [28, 29] decreasing the false positive rate in effect.

One of the problems to solve with ensemble systems is the combination of their outputs. Arithmetic mean or majority vote are preferred if individual classifiers have roughly the same accuracy [30]. This is not the case in our engine due to detectors specialized to some attacks. For such systems Evangelista [31] suggested to use average of mean and maximum scores of detectors within the ensemble, as it should be more robust to presence poor detectors without knowing which ones are poor. This combination function (further called Evangelista's) favors the highest anomaly scores and reduces the effect of poor detectors.

The combined output of 16 anomaly detectors is used as an input in five different LAMS models with different contexts. The rationale behind using more than one LAMS model is again the ensemble principle, as the hope is that errors of outputs of different LAMS models will be uncorrelated and they will cancel out. Contexts (features) of individual models are shown in Table 1 and they are named according to the publications which have used them in anomaly detection. Since all features are real numbers, the similarity is defined using the Gaussian kernel introduced in Section 3. The schema of the system is shown in Figure 4. The system consists from two blocks, where the first (left) block represents the anomaly detection part, while the second (right) block represents the part with LAMS models.

4.1.1. Dataset of NetFlow

To evaluate the effects of LAMS model on the NetFlow Anomaly detection engine we have created several datasets from a traffic captured on the network of Czech Technical University (CTU) in Prague. We have used three different approaches to create labels: manual labeling, infecting virtual machines, and attacking our computers within the network by us.

Manual Labeling. A week long capture from the university contains 41 517 828 flows between 19 261 different IP addresses. An experienced network operator has identified 10% as anomalous and 11% as legitimate. The most prevalent

Name	Context features
Xu-source [25]	entropy of source ports for the source IP entropy of destination ports for the source IP number of destination IP addresses for the source IP
Xu-destination [25]	entropy of source ports for the source IP entropy of destination ports for the source IP number of source IP addresses for the source IP
MINDS [8]	number of flows originating from the source IP number of flows originating from the destination IP number of different destination ports for the source IP
Lakhina [22]	number of flows originating from the source IP number of packets transferred from the source IP number of bytes transferred from the source IP
TAPS [26]	number of different destination IPs of the source IP number of unique ports of the source IP entropy of packets size of the source IP

Table 1: Features defining similarity in NetFlow LAMS models.

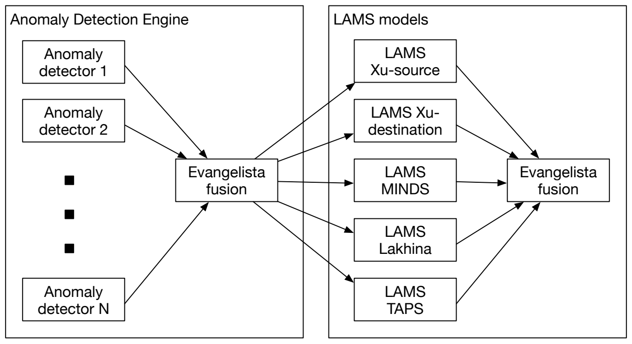


Figure 4: NetFlow anomaly detection system architecture. In the experimental evaluation (Section 4.1.2) we compare the efficacy of the anomaly detection engine alone (depicted in the left box) with the efficacy when using anomaly detection followed by LAMS models (both the boxes).

malicious traffic was ssh scan (55%) followed by p2p traffic (36%), horizontal scan (6.8%), and vertical scan (1.5%).

Malware Infection. The second dataset was created by executing real malware inside controlled virtual machine (VM)[4]. VMs were infected using malicious binaries captured during 24 hours of their traffic together with the traffic of the whole CTU network. The labels were created such that every NetFlow originating from the infected machines was labeled as malicious, divided into requests and responses according to the direction. NetFlows corresponding to normal traffic of Windows XP operating system running on the VMs, such as connection check, Windows Update checks, NTP checks, etc., were also labeled as normal.

In the first capture we have infected only one VM running Windows XP with Neeris [21] malware that has generated a lot of traffic (malware generated on average three Netflows per second). Second, we have infected ten different VMs by FastFlux botnet [21]. Finally, we have infected ten VMs by rbot we could control, and instruct it to perform ICMP flood attack against one of our servers.

Manual Attacks. Third dataset was created by network specialist attacking one computer with an open SSH port inside the university network. She has started her attack by horizontal scan of university network searching for a computer with an open SSH port. Then, she performed brute force cracking of an ssh password with a help of dictionary of 1000 passwords with the last one being correct. She has tried all passwords within 5-minute long window. Finally, she has downloaded 0.5 GB from the attacked computer to simulate stealing the data. Corresponding NetFlows in the dataset were labeled on the basis of known attacker and attacked IP addresses and ports and time of the attack information.

4.1.2. Experimental Results

Table 2 shows AUC scores of the anomaly detection engine after the first aggregation before using multiple LAMS models (denoted without LAMS), and after the second aggregation (denoted with LAMS) on individual datasets and attacks described previously. The results reveals that LAMS models improves the accuracy in detecting attack requests, but decreases the accuracy of responses. To investigate this, we have plot in Figure 5 distribution of points corresponding to flows related to the horizontal scan in the context defined by Lakhina's entropy based features. The figure confirms the assumption that requests are located in a very small and distinct part of the space and therefore LAMS models can well estimate the anomaly score, whereas responses are scattered all over the space mixed with other mostly legitimate traffic. According

[4]The bandwidth of the connection was lowered to 150kb/s to prevent generating huge amount of traffic

Manual labeling	AUC Without LAMS	AUC With LAMS
horizontal scan request	0.82	**0.83**
horizontal scan response	**0.77**	0.66
p2p request	0.87	**0.91**
p2p response	**0.89**	0.75
scan sql	**1**	**1**
scan sql response	**0.92**	0.91
ssh cracking request	0.80	**0.83**
ssh cracking response	**0.84**	0.71
vertical scan	**1**	**1**
vertical scan response	**0.98**	0.96
Malware infection		
Neris bot	0.69	**0.87**
FastFlux botnet	0.81	**0.88**
RBot ICMP flood	0.91	**0.94**
Artificial Attack		
SSH scan	0.93	**1**
SSH scan response	**0.84**	0.77
SSH cracking	**0.83**	**0.83**
SSH cracking response	0.77	**0.78**
Anomalous SSH download	0.95	**0.97**
Anomalous SSH download request	**0.95**	**0.95**

Table 2: Comparison of AUC Scores without and with LAMS model.

to Table 1 Lakhina's entropy features are calculated from a set of flows with the same source IP address receiving the same entropy and also anomaly values. During horizontal scan, each attacked network host will probably have only one response flow to the attack, which will be well hidden among host's own legitimate traffic. This causes (a) the responses to have different entropies and locations in the space and (b) receiving low anomaly values. Contrary, requests of the attack stands out, since they have the same source IP address and the same values of entropies. Moreover, this entropy would be very different from that of other hosts in the network leading to high anomaly. This explanation is supported by the fact that LAMS models do not decrease anomaly values of response to attacks targeting a single host (e.g., vertical scan). Finally note that it is important to detect at least one part of the attack, either the attack, or the responds.

To demonstrate the decrease in false positive rate on structured false positives, we have used the dataset described in Section 4.1.1 and analyzed the traffic

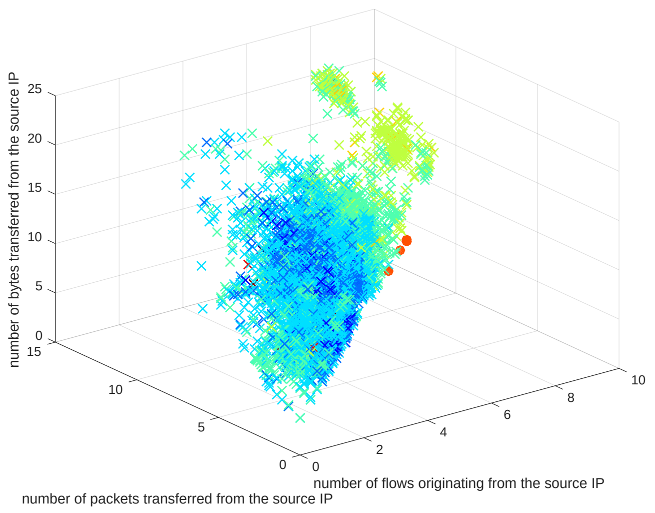


Figure 5: Visualization of the anomaly scores assigned by the Lakhina LAMS model (see Table 1) to the horizontal scan and corresponding responses contained in the manually labeled dataset described in Section 4.1.1. Each point represent one scan request (cross) or scan response (dot). The color corresponds to obtained anomaly score with red being the most anomalous and blue being the least. Visualization of all the other LAMS models defined in Table 1 can be see in the Appendix in Figure A.14.

identified as the most anomalous traffic by the first part of the detection system (combined output of the anomaly detectors without LAMS models). Within we have identified a legitimate traffic meeting the criteria of the structured false positives (being explainable and corresponding to rare events). The complete list is in Table 3 and includes responses of NTP servers, software licenses servers, downloads from local data and database servers, etc. The decrease has been again measured by the AUC score, where the false positives were treated as negative samples and all attacks identified in the dataset 4.1.1 as positive samples. AUCs of identified structured false positives are shown in Table 3. According to it LAMS models almost always reduces the false positive rate and increases the accuracy of detection.

The above investigation of false positives also revealed some unstructured false positives corresponding to excessive web and DNS traffic of some users, which anomaly detectors flagged as suspicious. LAMS models have decreased their anomaly values as is shown in the second part of the Table 3.

4.2. HTTP Anomaly Detection

Cisco Cognitive Threat Analytics (CTA) [32] is an anomaly-based IDS that uses HTTP proxy logs typically produced by the proxy servers located on the network perimeter and used by the network hosts to access web content. Contrary to NetFlow, these logs contain information extracted from HTTP header (time of the request, source IP address, destination IP address, url, MIME type,

	AUC	
Structured false positive	without LAMS	with LAMS
NTP server response	0.98	**0.99**
ICQ servers	0.98	**0.99**
Google crawler	0.99	**1**
Local data server downloads	0.9	**0.93**
Local database server requests	**0.98**	**0.98**
Local database server responses	0.96	**0.98**
Local software license server	0.19	**0.89**
Subnetwork gateway requests	0.83	**0.90**
Subnetwork gateway responses	**0.30**	0.24
HTTP proxy requests	0.75	**0.77**
HTTP proxy responses	**0.76**	0.73

	AUC	
Unstructured false positive	without LAMS	with LAMS
DNS request	0.63	**0.70**
DNS response	0.45	**0.71**
Web browsing user 1	0.99	**1**
Web browsing user 2	**0.99**	**0.99**
Web browsing user 3	0.92	**0.94**
Gmail.com servers	0.98	**0.99**

Table 3: AUC Scores of structured and unstructured false positives in the dataset introduced in Subsection 4.1.1 when LAMS models are used (right column) and not used (middle column). Higher AUC is better.

Name	Context features
LAMS 1	number of unique referrers of the visited domain number of unique operating systems used to visit domain number of unique MIME types hosted on the domain
LAMS 2	number of unique referrers of the visited domain number of unique HTTP status responses of the visited domain amount of bytes downloaded using the User-Agent
LAMS 3	number of unique User-Agents used to visit the domain entropy of the MIME types hosted on the domain number of unique referrers of the visited domain
LAMS 4	URL length number of unique number of unique operating systems used to visit domain number of unique referrers of the visited domain

Table 4: HTTP LAMS models.

downloaded and uploaded bytes, User-Agent identifier, etc.). For the complete list of items see Table A.6 in Appendix.

Anomaly detection part of the CTA consists from more than 30 various anomaly detectors detecting anomalies according to empirical probability estimates and conditional probability estimates of e.g., P(country), P(domain|host), P(User-Agent|second level domain), etc.), time series analysis (models of user activity over time, detection of sudden changes in activity, identification of periodical requests, etc.) and HTTP specific detectors (e.g., discrepancy in HTTP User-Agent field [33]).

The overall architecture of the system is the same as the NetFlow anomaly detection engine shown in Figure 4. Again, there is a first block of 30 anomaly detectors, whose output is combined by Evangelista's fusion function to obtain one anomaly score value for each HTTP proxy log entry. This output is used as an input to the second block with several LAMS models with context features listed in Table 4 and Gaussian similarity function defined in Section 3. Outputs of four LAMS models are again combined by Evangelista's fusion function to the final output.

4.2.1. Dataset of HTTP Proxy Logs

The database of HTTP proxy logs was collected from networks of 30 different companies of various sizes and types with collection period ranging from six days to two weeks. There are more than seven billion HTTP logs in the database. A small team of security researchers has identified in them 2 666 users infected with a malware of 825 various families that have generated in total more than 129 millions HTTP logs. HTTP logs corresponding to malware requests represent less than 2% of the total traffic of the individual networks except for few cases, where the networks were infected by ZeroAccess malware [34]. ZeroAccess is very noisy, generating many HTTP logs which has reached up to

21% of the total number of logs. The other malware families worth to mention were: Cycbot, QBot, SpyEye, BitCoinMiner, Zbot and many others. All the malware was identified using several approaches starting with an analysis of the most anomalous HTTP logs as reported by the CTA, malware reported by the individual network administrators, using blacklists and other public feeds or third-party software. The rest of the unlabeled logs were considered to be legitimate traffic. We are aware that within them there might be another proxy logs corresponding to malware traffic, but we are not aware about any additional method providing the ground truth. Moreover since the most anomalous traffic was always investigated for a malware HTTP logs, the most important part corresponding to low false positive rate was always labeled. This labeling of the most anomalous part has been done for the detection engine using only anomaly detection part and using both parts including LAMS models.

4.3. Experimental Results

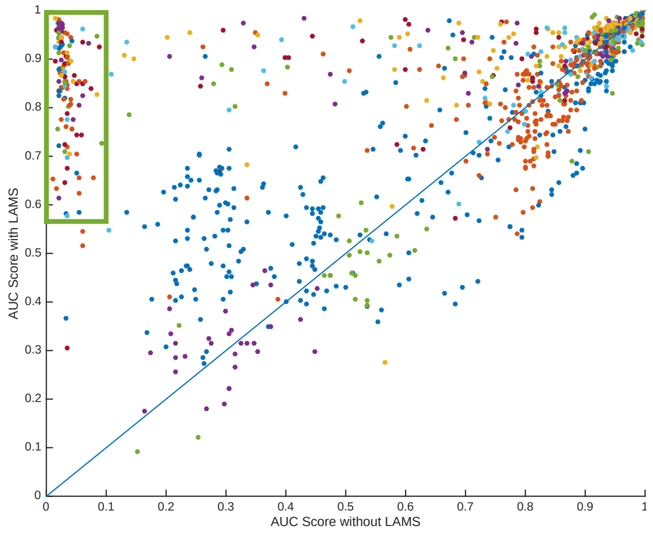


Figure 6: Scatter plot of the AUC scores when running the anomaly detection engine without and with the LAMS model. Each point represent one malware sample and different colors denote different malware families to illustrate variety of the samples. Points that are above the blue line represent an AUC improvement when LAMS is used, whereas point below represent cases where LAMS decreases the efficacy. As can be seen the LAMS model significantly improves the AUC score of the samples with AUC smaller than 0.2, as depicted by the green bar in the figure.

To demonstrate the advantage of LAMS models we have calculated AUC score on sub-datasets with HTTP logs of each malware family considered as positives and all unlabeled HTTP logs as negatives. Since there are 825 different malware families in the dataset the table showing the AUC improvement would be huge. Therefore we have decided to show the effect of the LAMS model in a figure. The Figure 6 shows a scatter plot where x-coordinate of each point is

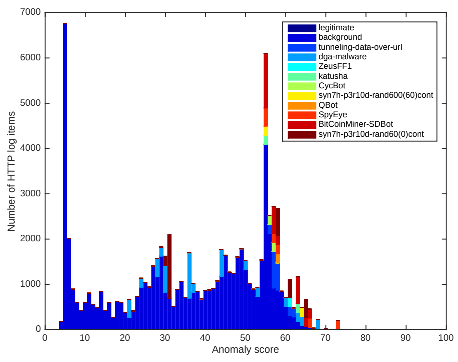


Figure 7: Without LAMS models

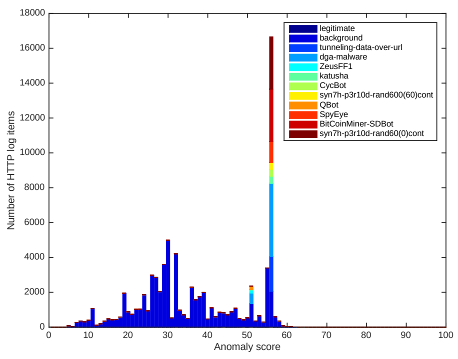


Figure 8: With LAMS models

Figure 9: Distributions of anomaly values of HTTP proxy logs with (right) and without (left) LAMS models.

AUC score of output of the first part of the detection engine using only anomaly detectors and the y-coordinate is AUC score of the output of the second part with LAMS models. This means that if the point is above the diagonal, LAMS models improve the detection accuracy and vice versa. We can observe that LAMS models generally improve the accuracy of detection since majority (75%) of the points is above the diagonal. Particularly noticeable are points within the green rectangle that would not be detectable without LAMS models, but LAMS models significantly increase their anomaly values.

Figures 7 and 8 show the distribution of anomaly scores of the anomaly detection engine without (left) and with (right) LAMS models. We can see that outputs of an anomaly detection engine without LAMS models on malware HTTP logs are more equally spread across the entire range of values. Contrary, when the anomaly detection engine uses LAMS models, the scores of malware's HTTP logs are located in the most anomalous part of the distribution.

5. Related Work

The prior art in reducing the false positives used statistics [35], time series [36], and machine learning algorithms [37] to achieve this goal. Most approaches works with a notion of a network alert, which is a collection of networks flows, rather than with individual flows, which is one of the biggest differentiating factor of our work.

Bolzoni *et al.* [38] propose to correlate anomalies in incoming and outgoing traffic as successful attack should raise alarms in both directions.

Pietraszek [37] has created a classifier that is reducing the workload of the human analyst by classifying the alerts into true positives and false positives. The knowledge of how to classify alerts is learned adaptively by observing the analyst. Similarly, Tian *et al.* [39] use the analyst feedback to generate custom-made filtering rules, that automatically discard similar false alerts encountered in the future. The need of human interaction represent always a bottleneck since the effectiveness of the IDS depends on the human analyst.

Hooper [40] introduced intelligent network quarantine channels technique to get additional information about the suspected hosts. This information is further used to estimate the probability of a host being infected. Although, this technique provides additional information that can reduce the amount of false alarms, it is not always allowed by the organization's security policy to perform such a host checking.

Viinikka and Debar [36] proposed to use Exponentially Weighted Moving Average (EWMA) control charts of the flow of alerts and try to spot abnormalities. Their experimental results concern the production of less and more qualitative alerts.

More recently, Dangelo *et al.* [41] proposed to use U-BRAIN [42] algorithm to infer set of rules from incomplete set of labeled data to separate anomalous and normal network events. The authors shown that the proposed method generalize well, but the resulting accuracy depends directly on the quality of training data. If the training data do not reflect all the aspects of the real network traffic, the risk of false positives and/or negatives increases. This introduces the need of continuous supervised re-training mechanism managed trough human-driven results validation.

The proposed model was inspired by trust models [43, 44, 45, 46] which are specialized knowledge structures designed to maintain information about the trustworthiness of a communication partners, either acquired from interactions, observed from the interactions of others, or received from others by means of reputation mechanism [47]. The design of trust models emphasizes features such as fast learning, robustness in response to false reputation information [48], and robustness with respect to environmental noise.

Extended trust models (ETM) introduced by Rehak *et al.* [49] enhance the original trust models with context representations in order to reflect the context of trusting situation, putting it on par with the identity of trusting parties. The paper [50] took the work further and placed the ETMs in the context of network security problem and use them to aggregate the anomaly scores provided by several anomaly detectors. Each of the network anomaly detectors had its own ETM model with a feature space defined by identity and detector-specific context, serving mainly as a long term memory of past anomaly events. Our current work is an extension of of the original ETM model, described in [50]. Unlike its predecessor, it has been simplified by omitting the fuzzy number representation of the anomaly value. Furthermore, we have discarded the identity, focusing only on the context of the observed network events. This simplifications have allowed us to re-formulate the problem more concisely as LAMS, and generalize its application in context of classification of a stream of events.

Pouzos *et al.* [14] introduced an adaptive kernel smoothing regression which to some extent similar to LAMS models. Contrary to the LAMS model, it updates only the nearest neighbor of the observation and allows the pivots to move towards regions with higher probability density. As discussed in Section 3, this behavior is undesirable, because it will cause LAMS models to forget rare events.

6. Conclusion

We have presented a method reducing false alarm rate of anomaly detection-based intrusion detection systems. The technique smooths detectors output simultaneously over time and space, which improves the estimate of true anomaly score.

We have proved under mild assumptions that the method reduces unstructured false positives caused by stochasticity of the network traffic. The experiments also showed reduction of the structured FPs, while not having major negative effect in the remaining cases.

The method has been evaluated using large-number of samples from two domains with diverse sets of anomaly detectors. Furthermore the method is a critical component of Cognitive Threat Analytics [32] — an online malware detection security-as-a-service product delivered by Cisco, which analyzes more than 10 billions of requests per day. This illustrates one of the key advantages of our method: simplicity and flexibility. It also shows that the method is usable in real-life production IDS systems.

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

[2] K. Julisch, M. Dacier, Mining intrusion detection alarms for actionable knowledge, in: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2002, pp. 366–375.

[3] K. Julisch, Clustering intrusion detection alarms to support root cause analysis, ACM Trans. Inf. Syst. Secur. 6 (4) (2003) 443–471.

[4] M. Rehák, M. Grill, Categorisation of False Positives: Not All Network Anomalies are Born Equal, unpublished Lecture (2014).

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

[6] D. E. Denning, An intrusion-detection model, Software Engineering, IEEE Transactions on (2) (1987) 222–232.

[7] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, J. Zhang, Real time data mining-based intrusion detection, in: DARPA Information Survivability Conference &amp; Exposition II, 2001. DISCEX'01. Proceedings, Vol. 1, IEEE, 2001, pp. 89–100.

[8] L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, P. Dokas, Minds-minnesota intrusion detection system, Next Generation Data Mining (2004) 199–218.

[9] E. A. Nadaraya, On estimating regression, Theory of Probability & Its Applications 9 (1) (1964) 141–142.

[10] G. S. Watson, Smooth regression analysis, Sankhyā: The Indian Journal of Statistics, Series A (1964) 359–372.

[11] L. Devroye, A. Krzy[Zdot]ak, An equivalence theorem for {L1} convergence of the kernel regression estimate, Journal of Statistical Planning and Inference 23 (1) (1989) 71 – 82.

[12] M. Rehák, Multiagent trust modeling for open network environments.

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

[14] F. Pouzols, A. Lendasse, Adaptive kernel smoothing regression using vector quantization, in: Evolving and Adaptive Intelligent Systems (EAIS), 2011 IEEE Workshop on, 2011, pp. 85–92.

[15] J. Friedman, T. Hastie, R. Tibshirani, The elements of statistical learning, Vol. 1, Springer series in statistics Springer, Berlin, 2001.

[16] I. K. Fodor, A survey of dimension reduction techniques (2002).

[17] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, The Journal of Machine Learning Research 3 (2003) 1157–1182.

[18] E. B. Claise, Cisco Systems NetFlow Services export Version 9, 2004.

[19] T. Fawcett, An introduction to roc analysis, Pattern recognition letters 27 (8) (2006) 861–874.

[20] M. Rehak, M. Pechoucek, M. Grill, J. Stiborek, K. Bartoš, P. Celeda, Adaptive multiagent system for network traffic monitoring, IEEE Intelligent Systems (3) (2009) 16–25.

[21] S. Garcia, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, Computers & Security 45 (2014) 100–123.

[22] A. Lakhina, M. Crovella, C. Diot, Diagnosing network-wide traffic anomalies, in: ACM SIGCOMM Computer Communication Review, Vol. 34, ACM, 2004, pp. 219–230.

[23] A. Lakhina, M. Crovella, C. Diot, Mining anomalies using traffic feature distributions, in: ACM SIGCOMM Computer Communication Review, Vol. 35, ACM, 2005, pp. 217–228.

[24] T. Pevny, M. Rehak, M. Grill, Detecting anomalous network hosts by means of pca, in: Information Forensics and Security (WIFS), 2012 IEEE International Workshop on, 2012, pp. 103–108.

[25] K. Xu, Z.-L. Zhang, S. Bhattacharyya, Profiling internet backbone traffic: behavior models and applications, in: ACM SIGCOMM Computer Communication Review, Vol. 35, ACM, 2005, pp. 169–180.

[26] A. Sridharan, T. Ye, S. Bhattacharyya, Connectionless port scan detection on the backbone, in: Performance, Computing, and Communications Conference, 2006. IPCCC 2006. 25th IEEE International, IEEE, 2006, pp. 10–pp.

[27] M. Grill, I. Nikolaev, V. Valeros, M. Rehak, Detecting dga malware using netflow, in: Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, 2015, pp. 1304–1309.

[28] T. G. Dietterich, Ensemble methods in machine learning, in: MULTIPLE CLASSIFIER SYSTEMS, LBCS-1857, Springer, 2000, pp. 1–15.

[29] R. Polikar, Ensemble based systems in decision making, Circuits and Systems Magazine, IEEE 6 (3) (2006) 21–45.

[30] L. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley, 2004.

[31] P. F. Evangelista, M. J. Embrechts, B. K. Szymanski, Data fusion for outlier detection through pseudo-roc curves and rank distributions, in: Neural Networks, 2006. IJCNN'06. International Joint Conference on, IEEE, 2006, pp. 2166–2173.

[32] Cisco Systems, CTA cisco cognitive threat analytics on cisco cloud web security, `http://www.cisco.com/c/en/us/solutions/enterprise-networks/cognitive-threat-analytics` (2014–2015).

[33] M. Grill, M. Rehak, Malware detection using http user-agent discrepancy identification, in: Information Forensics and Security (WIFS), 2014 IEEE International Workshop on, 2014, pp. 221–226.

[34] J. Wyke, The zeroaccess botnet–mining and fraud for massive financial gain, Sophos Technical Paper.

[35] G. P. Spathoulas, S. K. Katsikas, Reducing false positives in intrusion detection systems, computers & security 29 (1) (2010) 35–44.

[36] J. Viinikka, H. Debar, L. M, A. Lehikoinen, M. Tarvainen, Processing intrusion detection alert aggregates with time series modeling, Information Fusion 10 (4) (2009) 312 – 324, special Issue on Information Fusion in Computer Security.

[37] T. Pietraszek, Using adaptive alert classification to reduce false positives in intrusion detection, in: Recent Advances in Intrusion Detection, Springer, 2004, pp. 102–124.

[38] D. Bolzoni, S. Etalle, Aphrodite: An anomaly-based architecture for false positive reduction, arXiv preprint cs/0604026.

[39] Z. Tian, W. Zhang, J. Ye, X. Yu, H. Zhang, Reduction of false positives in intrusion detection via adaptive alert classifier, in: Information and Automation, 2008. ICIA 2008. International Conference on, 2008, pp. 1599–1602.

[40] E. Hooper, An intelligent detection and response strategy to false positives and network attacks, in: Information Assurance, 2006. IWIA 2006. Fourth IEEE International Workshop on, 2006, pp. 20 pp.–31.

[41] G. Dangelo, F. Palmieri, M. Ficco, S. Rampone, An uncertainty-managing batch relevance-based approach to network anomaly detection, Applied Soft Computing 36 (2015) 408–418.

[42] S. Rampone, C. Russo, A fuzzified brain algorithm for learning dnf from incomplete data, arXiv preprint arXiv:1002.4014.

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

[44] J. Sabater, C. Sierra, Reputation and social network analysis in multi-agent systems, in: Proceedings of AAMAS '02, Bologna, Italy, 2002, pp. 475–482.

[45] S. Ramchurn, N. Jennings, C. Sierra, L. Godo, Devising a trust model for multi-agent interactions using confidence and reputation, Applied Artificial Intelligence 18 (9-10) (2004) 833 – 852.

[46] C. Castelfranchi, R. Falcone, 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, p. 72.

[47] A. Josang, E. Gray, M. Kinateder, Simplification and analysis of transitive trust networks, Web Intelligence and Agent Systems 4 (2) (2006) 139–162.

[48] T. D. Huynh, N. R. Jennings, N. R. Shadbolt, An integrated trust and reputation model for open multi-agent systems, Journal of Autonomous Agents and Multi-Agent Systems 13 (2) (2006) 119–154.

[49] M. Rehak, M. Pechoucek, Trust modeling with context representation and generalized identities, in: Cooperative Information Agents XI, no. 4676 in LNAI/LNCS, Springer-Verlag, 2007.

[50] M. rehk, M. pchouek, M. Grill, K. Bartos, Trust-based classifier combination for network anomaly detection, in: M. Klusch, M. Pchouek, A. Polleres (Eds.), Cooperative Information Agents XII, Vol. 5180 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 116–130.

Appendix A. Examples of NetFlow record and HTTP logs

Feature	Example of values
start-time	1440870672
duration	5
protocol	TCP
source ip	192.168.1.2
destination ip	208.80.154.224
source port	1604
destination port	443
TCP flags	.AP.SF
type of service	0
number of packets	1201
number of bytes	1.8 M

Table A.5: Example of one NetFlow record containing information about both communication participants (source and destination IP and port), time of the communication, protocol used, bitwise OR of all TCP flags, type of service (tos), number of packets and bytes transferred in both directions.

Feature	Value example
x-timestamp-unix	1440870672
sc-http-status	200
sc-bytes	16671
cs-bytes	0
cs-uri-scheme	https
cs-host	en.wikipedia.org
cs-uri-port	1604
cs-uri-path	/wiki/Anomaly_detection
cs-uri-query	
cs-username	Martin Grill
x-elapsed-time	5
s-ip	208.80.154.224
c-ip	192.168.1.2
Content-Type	text/html; charset=UTF-8
cs(Referer)	https://www.google.com/
cs-method	GET
cs(User-Agent)	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.157 Safari/537.36

Table A.6: Example of HTTP log. This is one of the HTTP logs created when downloading a wikipedia page. Each page download generates more HTTP logs since all the page resources have to be downloaded. To render the page from the example the browser generated additional 20 HTTP logs, containing page styles, scripts, pictures, etc. In the example the cssc prefixes denote the client to server and server to client communication respectively, so the sc-bytes represent the amount of bytes downloaded by the client and cs-bytes the amount of uploaded bytes to the server. The rest of the features is self-explanatory.

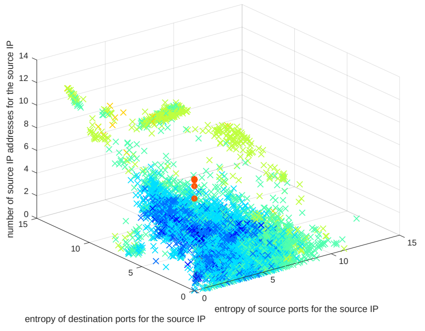


Figure A.10: Xu-source

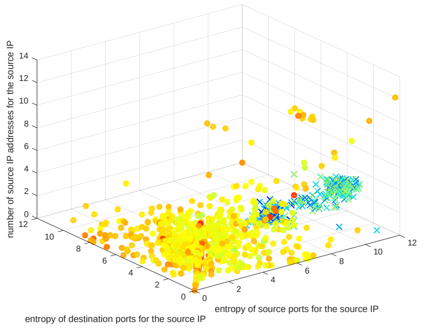


Figure A.11: Xu-destination

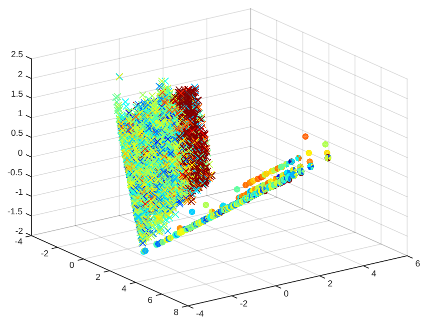


Figure A.12: MINDS

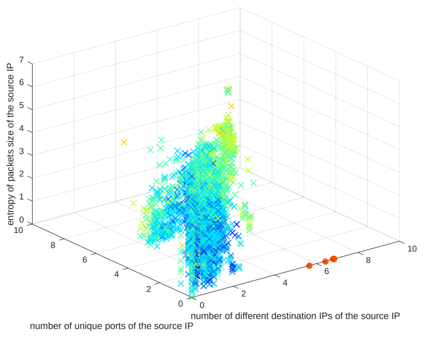


Figure A.13: TAPS

Figure A.14: Additional visualization to the Figure 5 of the horizontal scan and corresponding responses contained in the manually labeled dataset described in Section 4.1.1 in the context space of the rest of the LAMS models defined in Table 1. Each point on the individual scatter plots represent one scan request (cross) or scan response (dot). The color corresponds to obtained anomaly score with red being the most anomalous and blue being the least. Since the MINDS model uses four features to define its context we have used the Multidimensional scaling to be able to show the data in three dimensional plot (therefore there are no axis labels). The figures show that similarly to the Lakhina model (Figure 5), Xu-source and TAPS models have the responses spread across a region of low anomaly score and the requests limited in a small region of high anomaly. For the Xu-destination the requests are spread in the context space but still maintaining high anomaly score. For this particular behavior the MINDS model as the only one increases anomaly score of a part of the responses and reduces the anomaly score of the requests.