ILAB: An Interactive Labelling Strategy for Intrusion Detection

Anaël Beaugnon, Pierre Chifflier, Francis Bach

ILAB: An Interactive Labelling Strategy for Intrusion Detection

Anaël Beaugnon[✉1,2], Pierre Chifflier[1], and Francis Bach[2]

[1] French Network Security Agency (ANSSI), Paris, France
[2] INRIA, École Normale Supérieure, Paris, France
{anael.beaugnon,pierre.chifflier}@ssi.gouv.fr
francis.bach@ens.fr

Abstract. Acquiring a representative labelled dataset is a hurdle that has to be overcome to learn a supervised detection model. Labelling a dataset is particularly expensive in computer security as expert knowledge is required to perform the annotations. In this paper, we introduce ILAB, a novel interactive labelling strategy that helps experts label large datasets for intrusion detection with a reduced workload. First, we compare ILAB with two state-of-the-art labelling strategies on public labelled datasets and demonstrate it is both an effective and a scalable solution. Second, we show ILAB is workable with a real-world annotation project carried out on a large unlabelled NetFlow dataset originating from a production environment. We provide an open source implementation (https://github.com/ANSSI-FR/SecuML/) to allow security experts to label their own datasets and researchers to compare labelling strategies.

Keywords: Intrusion Detection · Active Learning · Rare Category Detection

1 Introduction

Supervised learning is adapted to intrusion detection and has been successfully applied to various detection problems: Android applications [11], PDF files [7,35], botnets [2,5], Windows audit logs [4], portable executable files [19]. However, supervised detection models must be trained on representative labelled datasets which are particularly expensive to build in computer security. Expert knowledge is required to annotate and data are often confidential. As a result, crowdsourcing [37] cannot be applied as in computer vision or natural language processing to acquire labelled datasets at low cost. Some labelled datasets related to computer security are public (Malicia project [22], KDD99 [41], kyoto2006 [39], etc.) but they are quickly outdated and they often do not account for the idiosyncrasies of each deployment context.

Experts are essential for annotating but they are an expensive resource, that is why the labelling process must use expert time efficiently. Active learning methods have been proposed to reduce the labelling cost by asking the expert

to annotate only the most informative examples [32]. However, classical active learning methods often suffer from sampling bias [29, 34]: a family (a group of similar malicious or benign examples) may be completely overlooked by the annotation queries as the expert is asked to annotate only the most informative examples. Sampling bias is a significant issue in intrusion detection: it may lead to missing a malicious family during the labelling process, and being unable to detect it thereafter. Moreover, the labelling strategy must scale to large datasets to be workable on real-world annotation projects.

Finally, active learning is an interactive process which must ensure a good expert-model interaction, i.e. a good interaction between the expert who annotates and the detection model [33, 43]. The expert annotations improve not only the detection model but also the relevance of the following annotation queries. A low execution time is thus required to allow frequent updates of the detection model with the expert feedback. A labelling strategy with a high execution time would alter the expert-model interaction and is unlikely to be accepted by experts.

In this paper, we introduce ILAB, a novel interactive labelling strategy that helps an expert acquire a representative labelled dataset with a reduced workload. ILAB relies on a new hierarchical active learning method with binary labels (malicious vs. benign) and user-defined malicious and benign families. It avoids the sampling bias issue encountered by classical active learning as it is designed to discover the different malicious and benign families. Moreover, the scalable algorithms used in ILAB make it workable on large datasets and guarantee a low expert waiting time for a good expert-model interaction.

Our paper makes the following contributions:

- We present a novel active learning method called ILAB designed to avoid sampling bias. It has a low computation cost to ensure a good expert-model interaction, and it is scalable to large datasets.
- We compare ILAB with two state-of-the-art active learning methods for intrusion detection [14, 40] on two detection problems. We demonstrate that ILAB improves the scalability without reducing the effectiveness. Up to our knowledge, [14, 40] have never been compared. We provide an open source implementation of ILAB and of these two labelling strategies to foster comparison in future research works.
- We show that ILAB is a workable labelling strategy that scales to large real-world datasets with an annotation project on NetFlow data originating from a production environment. We provide an open source implementation of the graphical user interface deployed during the annotation project to allow security experts to label their own datasets.

The rest of the paper is organized as follows. Section 2 presents the sampling bias issue in active learning and related works. The problem being addressed and the notations are detailed in Section 3. Section 4 explains ILAB labelling strategy. Finally, Section 5 compares ILAB with state-of-the-art labelling strategies through simulations run on public fully labelled datasets, and Section 6 presents

a real-world annotation project carried out with ILAB on a large unlabelled NetFlow dataset.

2 Background and Related Work

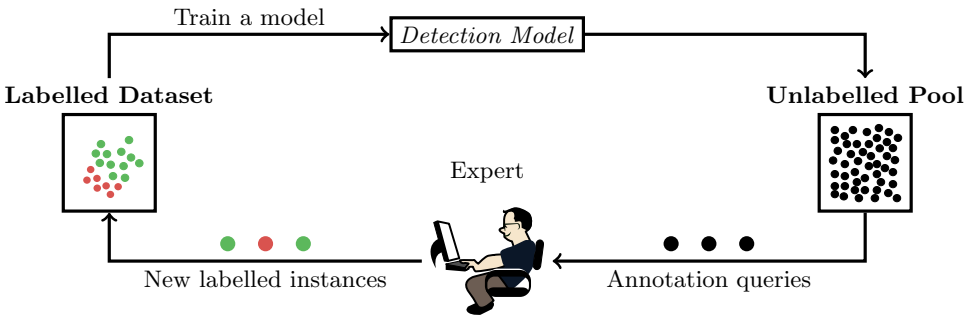


Fig. 1: Active Learning: An Interactive Process

Active Learning. Active learning [32] methods have been developed in the machine learning community to reduce the labelling cost. A labelling strategy asks the expert to annotate only the most informative instances, i.e. the ones that lead to the best detection model. Active learning methods rely on an interactive process where the expert is asked to annotate some instances from a large unlabelled pool to improve the current detection model and the relevance of the future annotation queries (see Figure 1). However, annotating only the most informative instances may cause a family of observations to be completely missed by the labelling process (see [8, 29] for theoretical examples) and, therefore, may have a negative impact on the performance of the detection model.

Sampling Bias. Figure 2 provides an example of sampling bias in one dimension with uncertainty sampling [20] which queries the closest instances to the decision boundary. Each block represents a malicious or a benign family. With this data distribution, instances from the family M_1 are unlikely to be part of the initial training dataset, and so the initial decision boundary is likely to lie between the families B_2 and M_3. As active learning proceeds, the classifier will gradually converge to the decision boundary between the families B_2 and M_2 and will only ask the expert to

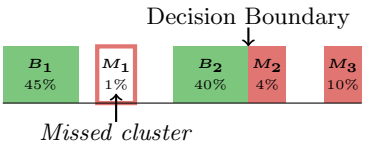


Fig. 2: Sampling Bias Example

annotate instances from these two families to refine the decision boundary. The malicious family M_1 on the left is completely overlooked by the query algorithm as the classifier is mistakenly confident that the entire family is benign. As the malicious family M_1 is on the wrong side of the decision boundary, the classifier will not be able to detect this malicious family thereafter.

Sampling bias is a significant problem for intrusion detection that may lead to malicious families remaining completely undetected. Besides, the risk of sampling bias is even higher for intrusion detection than for other application domains because the initial labels are not uniformly distributed. Uniform random sampling cannot be used to acquire the initial labelled instances as the malicious class is too under-represented. The signatures widely deployed in detection systems can provide initial labels but they likely all belong to the same family or to a small number of families.

Related Work. Online active learning [21,30,31,44,45] is well-suited to follow the evolution of the threats: experts perform annotations over time to update the detection model that is already deployed. In this setting, the detection model in production has been initially trained on a labelled dataset representative of the deployment environment. In our case, such a representative labelled dataset is unavailable and the objective is to acquire it offline to train the initial detection model.

Some works focus on offline active learning to build a labelled dataset for intrusion detection. First, Almgren et al. [1] have applied plain uncertainty sampling [20] to intrusion detection before the sampling bias issue has been discovered. Then, Aladin [40] and Görnitz et al. [14] have proposed new labelling strategies for intrusion detection that intend to discover the different malicious families. Aladin applies rare category detection [26] on top of active learning to foster the discovery of the different families, and Görnitz et al. use a k-nearest neighbour approach to detect yet unknown malicious families. However, both [40] and [14] deal with sampling bias at the expense of the expert-model interaction. These labelling strategies require heavy computations to generate the annotation queries that cause long waiting-periods that cannot be exploited by the expert. ILAB relies on rare category detection to avoid sampling bias, as Aladin, but with a divide and conquer approach to ensure a good expert-model interaction. Aladin [40] and Görnitz et al. [14] labelling strategies have never been compared to our knowledge. We compare ILAB with these two labelling strategies in the simulations presented in Section 5 and we provide open source implementations in order to foster comparison in future research works.

Finally, active learning is an interactive process where a user interface is required for the expert to annotate. Almgren et al. and Görnitz et al. have only run simulations on fully labelled datasets with an oracle answering the annotation queries and they have not mentioned any user interface. Aladin has a corresponding graphical user interface, but [40] provides no detail about it. As an ergonomic user interface can definitely reduce the expert effort [9, 33], ILAB comes up with an open source graphical user interface briefly described in Section 6.

3 Problem Statement

Our goal is to acquire a representative labelled dataset from a pool of unlabelled instances with a reduced human effort. Both the number of annotations asked from the expert and the computation time for generating the annotation queries must be minimized to reduce the workload and ensure a good expert-model interaction. We assume that there is no adversary attempting to mislead the labelling strategy as it is performed offline before the detection model is deployed in production.

Notations. Let $\mathcal{D} = \{x_i \in \mathbb{R}^m\}_{1 \leq i \leq N}$ be the dataset we want to label partially to learn a supervised detection model $\mathcal{M}$. It contains N instances described by m real-valued features. For example, each instance x_i could represent a PDF file, an Android application, the traffic of an IP address, or the activity of a user. Such unlabelled data are usually easy to acquire from the environment where the detection system is deployed (files, network traffic captures, or logs for example).

To represent an instance with real-valued features the expert must extract discriminating features and transform them into real values. Many research works focus on feature extraction for given detection problems: Android applications [11], PDF files [7,35], Windows audit logs [4], portable executable files [19]. In this paper, we do not address feature extraction and we focus on reducing the cost of building a representative labelled dataset with an effective labelling strategy. Instances are represented by real-valued features regardless of the detection problem thanks to feature extraction. As a result, labelling strategies are generic regarding the detection problems.

Let $\mathcal{L} = \{\texttt{Malicious}, \texttt{Benign}\}$ be the set of labels and $\mathcal{F}_y$ be the set containing the user-defined families of the label $y \in \mathcal{L}$. For example, malicious instances belonging to the same family may exploit the same vulnerability, they may be polymorphic variants of the same malware, or they may be emails coming from the same spam campaign.

Our aim is to create a labelled dataset

$$\mathcal{D}_L \subseteq \{(x, y, z) \mid x \in \mathcal{D},\ y \in \mathcal{L},\ z \in \mathcal{F}_y\}$$

maximizing the accuracy of the detection model $\mathcal{M}$ trained on $\mathcal{D}_L$. $\mathcal{D}_L$ associates a label $y \in \mathcal{L}$ and a family $z \in \mathcal{F}_y$ to each instance $x \in \mathcal{D}$. The labelled dataset $\mathcal{D}_L$ is built with an iterative active learning strategy. At each iteration, a security expert is asked to annotate, with a label and a family, $b \in \mathbb{N}$ instances selected from the pool of remaining unlabelled instances denoted by $\mathcal{D}_U$. During the annotation process, the expert cannot annotate more instances than the annotation budget $B \in \mathbb{N}$.

Objective. The objective of the labelling strategy is to build $\mathcal{D}_L$ maximizing the accuracy of the detection model $\mathcal{M}$ while asking the expert to annotate at most B instances. In other words, the labelling strategy aims to ask the expert to annotate the B instances that maximize the performance of the detection model $\mathcal{M}$. Besides, the labelling strategy must be scalable to work on large datasets while keeping a low expert waiting time.

4 ILAB Labelling Strategy

ILAB is an iterative annotation process based on active learning [32] and rare category detection [26]. At each iteration, the expert is asked to annotate b instances to improve the current detection model and to discover yet unknown families. Active learning improves the binary classification model raising the alerts while rare category detection fosters the discovery of new families to avoid sampling bias. First, we describe how we initialize the active learning process and then we explain the labelling strategy, i.e. which instances are selected from the unlabelled pool to be annotated by the expert.

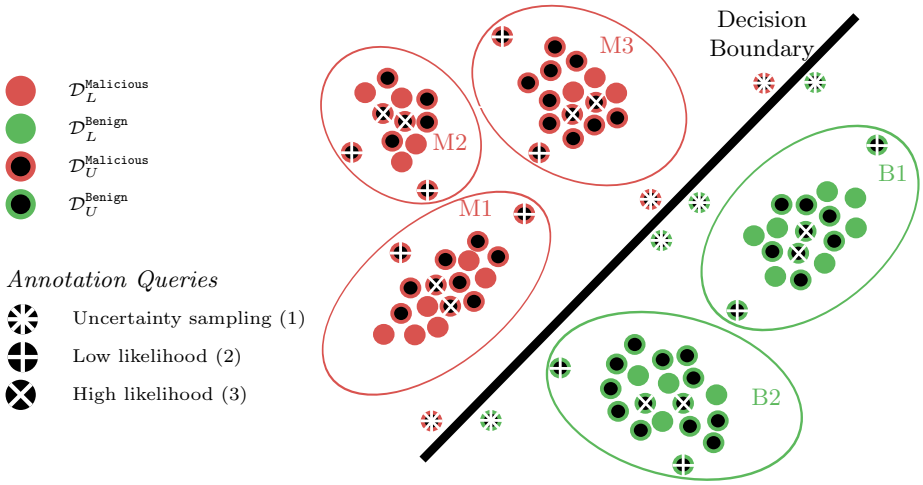


Fig. 3: ILAB Labelling Strategy

Initial Supervision. The active learning process needs some initial labelled examples to learn the first supervised detection model. This initial supervision can be difficult to acquire for detection problems. The `Malicious` class is usually too under-represented for uniform random sampling to be effective at collecting a representative labelled dataset.

If a public labelled dataset is available for the detection problem considered, it can be used for the initial supervision. Otherwise, the signatures widely deployed in detection systems can provide `Malicious` examples at low cost, and random sampling can provide `Benign` examples. In both cases, the initial labelled dataset does not contain all the malicious families we want to detect, and it is not representative of the data in the deployment environment. ILAB enriches the initial labelled dataset across the iterations to make it representative of the environment where the detection system is deployed.

The iterations are performed until the annotation budget B has been spent. At each iteration, $b_{\texttt{uncertain}}$ annotation queries are generated with *uncertainty sampling* to improve the detection model and $b_{\texttt{families}} = b - b_{\texttt{uncertain}}$ instances are queried for annotation with *rare category detection* to avoid sampling bias (see Figure 3).

4.1 Uncertainty Sampling

A binary probabilistic detection model $\mathcal{M}$ is learned from the annotated instances in $\mathcal{D}_L$. We use a discriminant linear model, i.e. logistic regression [10]. Linear models are highly valued by computer security experts who do not trust black box detection models [27]. These detection models can be interpreted because the coefficients associated with each feature represent their contribution to the detection model. Besides, discriminant models are known to be better than generative ones in active learning settings [47]. Finally, learning a logistic regression model and applying it to predict the label of new instances is fast so the expert does not wait a long time between iterations. Our approach is generic, the expert can choose to use another model class particularly suited for her application.

The rare malicious families are often the most interesting in intrusion detection, hence the impact of the training instances from rare families is increased. The logistic regression model is learned with sample weights inverse to the proportion of the family in the training dataset:

$$\beta(x, y, z) = \frac{|\mathcal{D}_L|}{|\{(x', y', z') \in \mathcal{D}_L \mid y' = y \wedge z' = z\}|}.$$

The weights are capped, $\hat{\beta} = \min(\beta, 100)$, to avoid giving too much weight to very rare families. Learning the logistic regression detection model with these weights is crucial to ensure a good detection of the rare malicious families.

The model $\mathcal{M}$ is used to compute the probability $p(x)$ that an unlabelled instance $x \in \mathcal{D}_U$ is `Malicious` according to $\mathcal{M}$:

$$\forall x \in \mathcal{D}_U,\ p(x) = P_{\mathcal{M}}(y = \texttt{Malicious} \mid x).$$

Annotation Queries. The $b_{\texttt{uncertain}}$ unlabelled instances which are the closest to the decision boundary of $\mathcal{M}$ are annotated by the expert:

$$\arg\min_{x \in \mathcal{D}_U} |p(x) - 1/2|. \tag{1}$$

The detection model is uncertain about the label of these instances, that is why their annotations allow to improve the detection model. This step corresponds to uncertainty sampling [20], a classical active learning method applied in [1]. Uncertainty sampling suffers, however, from sampling bias [29]. We also perform rare category detection to foster the discovery of yet unknown families.

4.2 Rare Category Detection

Rare category detection is applied on the instances that are more likely to be `Malicious` and `Benign` (according to the detection model $\mathcal{M}$) separately. Not all families are present in the initial labelled dataset and rare category detection [26] fosters the discovery of yet unknown families to avoid sampling bias. One might think that we could run rare category detection only on the malicious instances since it is the class of interest in intrusion detection. However, a whole malicious family may be on the wrong side of the decision boundary (see the family M_1 in Figure 2), and thus, running rare category detection on the predicted benign instances is necessary. Hereafter, we only detail the rare category detection run on the `Malicious` predictions since the analysis of the `Benign` ones is performed similarly.

Let $\mathcal{D}_U^{\texttt{Malicious}}$ be the set of instances whose predicted label by $\mathcal{M}$ is `Malicious` and $\mathcal{D}_L^{\texttt{Malicious}}$ be the set of malicious instances already annotated by the expert. First, a multi-class logistic regression model is learned from the families specified in $\mathcal{D}_L^{\texttt{Malicious}}$ to predict the family of the instances in $\mathcal{D}_U^{\texttt{Malicious}}$. Let $\mathcal{C}_f$ be the set of instances from $\mathcal{D}_L^{\texttt{Malicious}} \cup \mathcal{D}_U^{\texttt{Malicious}}$ whose family (annotated or predicted) is f. Each family f is modelled with a Gaussian distribution $\mathcal{N}(\mu_f, \Sigma_f)$ depicted by an ellipsoid is Figure 3. The mean μ_f and the diagonal covariance matrix Σ_f are learned with Gaussian Naive Bayes [10]. We denote by $p_{\mathcal{N}(\mu_f,\Sigma_f)}(x)$ the probability that x follows the Gaussian distribution $\mathcal{N}(\mu_f, \Sigma_f)$.

Annotation Queries. The family annotation budget $b_{\texttt{families}}$ is evenly distributed among the different families. We now explain which unlabelled instances are queried for annotation from each family.

First, ILAB asks the expert to annotate instances that are likely to belong to a yet unknown family to avoid sampling bias. These instances are located at the edge of the ellipsoid, they have a low likelihood of belonging to the family f [26, 40]:

$$\underset{x \in \mathcal{C}_f \setminus \mathcal{D}_L^{\texttt{Malicious}}}{\arg\min} \; p_{\mathcal{N}(\mu_f,\Sigma_f)}(x). \tag{2}$$

Then, ILAB queries representative examples of each family for annotation. These instances are close to the centre of the ellipsoid, they have a high likelihood of belonging to the family f:

$$\underset{x \in \mathcal{C}_f \setminus \mathcal{D}_L^{\texttt{Malicious}}}{\arg\max} \; p_{\mathcal{N}(\mu_f,\Sigma_f)}(x). \tag{3}$$

Half the budget is allocated to low likelihood instances, and the other half to high likelihood instances. Low likelihood instances are likely to belong to yet unknown families that is why these annotation queries foster the discovery of new families. They are, however, more likely to be outliers that may impair the detection model performance. ILAB also asks the expert to annotate high likelihood instances to get more representative examples of the families in the labelled dataset for a better generalization of the detection model.

5 Comparison with State of the Art Labelling Strategies

5.1 Datasets

Labelling strategies are generic methods that can be applied to any detection problem once the features have been extracted. We consider a system and a network detection problem: 1) detection of malicious PDF files with the dataset Contagio[3], and 2) network intrusion detection with the dataset NSL-KDD[4]. These datasets cannot be used to train a model intended for production as they are non-representative of real-world data. However, our comparisons are relevant as we are not comparing attack detection models but labelling strategies in order to train attack detection models on new problems.

Contagio is a public dataset composed of 11,101 malicious and 9,000 benign PDF files. We transform each PDF file into 113 numerical features similar to the ones proposed by Smutz and Stavrou [35, 36].

NSL-KDD contains 58,630 malicious and 67,343 benign instances. Each instance represents a connection on a network and is described by 7 categorical features and 34 numerical features. The 7 categorical features (e.g. `protocol_type` with the possible values `tcp`, `udp` or `icmp`) are encoded into several binary features corresponding to each value (e.g. `tcp` $\rightarrow [1, 0, 0]$, `udp` $\rightarrow [0, 1, 0]$, `icmp` $\rightarrow [0, 0, 1]$). We end up with 122 features.

Dataset	#instances	#features	#malicious families	#benign families
Contagio_10%	$10,000$	113	16	30
NSL-KDD_10%	$74,826$	122	19	15

Table 1: Description of the Public Datasets

The malicious instances in NSL-KDD are annotated with a family but the benign ones are not, and Contagio does not provide any family information. The families are, however, required to run simulations with Aladin and ILAB, and to assess the sampling bias of the different labelling strategies. We have assigned families to the remaining instances with a k-means clustering and the number of families k has been selected visually with the silhouette coefficient [28].

Neither dataset has a proportion of malicious instances representative of a typical network (55% for Contagio and 47% for NSL-KDD). We have uniformly sub-sampled the malicious class to get 10% of malicious instances. Table 1 describes the resulting datasets: Contagio_10% and NSL-KDD_10%.

[3] `http://contagiodump.blogspot.fr/`
[4] `http://www.unb.ca/cic/research/datasets/nsl.html`

5.2 Labelling Strategies

We compare ILAB with uncertainty sampling [20], Aladin [40], and Görnitz et al. labelling method [14]. Since there is no open source implementation of these labelling strategies, we have implemented them in Python with the machine learning library scikit-learn [25]. All the implementations are released to ease comparison in future research works. We briefly present each labelling strategy, we provide some details about our implementations and how we set the additional parameters if relevant.

Uncertainty Sampling [20]. At each iteration, a binary logistic regression model is trained on the labelled instances, and the expert is asked to annotate the b most uncertain predictions, i.e. the closest to the decision boundary. Uncertainty sampling has no additional parameter.

Görnitz et al. labelling strategy [14]. At each iteration, a semi-supervised anomaly detection model is trained on both the labelled and the unlabelled instances. The model relies on an adaptation of an unsupervised anomaly detection model, Support Vector Data Description (SVDD) [42], that takes into account labelled instances. It consists in a sphere defined by a centre $c \in \mathbb{R}^m$ and a radius $r \in \mathbb{R}$: the instances inside are considered benign, and the ones outside malicious. The labelling strategy queries instances that are both close to the decision boundary and have few malicious neighbours to foster the discovery of new malicious families. The nearest neighbours are computed with the Euclidean distance with the scikit-learn ball tree implementation [23] that is effective with a large number of instances in high dimension.

Semi-supervised SVDD has no open source implementation, so we have implemented it for our experiments with the information provided in [12–14]. The parameters c, r, and the margin $\gamma \in \mathbb{R}$ are determined with the quasi-Newton optimization method BFGS [46] available in scipy [17]. The optimization algorithm requires initial values for c, r, and γ that are not specified in the papers. We initialize c with the mean of the unlabelled and benign instances, r with the average distance of the unlabelled and benign instances to the centre c, and γ with the default value 1. Moreover, the detection model has three parameters: $\eta_U \in \mathbb{R}$ and $\eta_L \in \mathbb{R}$, the weights of the unlabelled and labelled instances, and κ the weight of the margin γ. The authors provide no information about how to set these parameters. When we set them to the default value 1, numerical instabilities prevent the optimization algorithm from converging properly, and lead to an extremely high execution time and very poor performance (more than 2 hours for training the model on Contagio_10% to get an AUC below 93%). We have thus worked on the setting of these parameters. We have set η_U and η_L to the inverse of the number of unlabelled and labelled instances, to give as much weight to unlabelled and labelled instances, and to ensure numerical stability. The detection model is trained without any kernel as in the experiments presented in [12–14].

Finally, the labelling strategy requires to set two additional parameters: $k \in \mathbb{N}$ the number of neighbours considered, and $\delta \in [0, 1]$ the trade-off between querying instances close to the decision boundary and instances with few malicious neighbours. We use $k = 10$ as in [14] and the default value $\delta = 0.5$.

Aladin [40]. Aladin runs rare category detection on all the data. It asks the expert to annotate uncertain instances lying between two families to refine the decision boundaries, and low likelihood instances to discover yet unknown families. Aladin does not have additional parameters.

This labelling strategy relies on a multi-class logistic regression model and a multi-class Gaussian Naive Bayes model. The logistic regression parameters are selected automatically with a grid search 4-fold cross validation optimizing the AUC [16]. The penalty norm is either ℓ_1 or ℓ_2 and the regularization strength is selected among the values $\{0.01, 0.1, 1, 10, 100\}$. The Gaussian Naive Bayes model is trained without any prior.

ILAB. ILAB labelling strategy has only an additional parameter: $b_{\texttt{uncertain}}$. It is set to 10% of the number of annotations performed at each iteration, i.e. $b_{\texttt{uncertain}} = 10$ in our case. Some instances near the decision boundary are annotated to help the detection model make a decision about these instances, but not too many since these instances are often harder to annotate for the expert [3, 15, 33] and they may lead to a sampling bias [29].

The logistic regression and Gaussian Naive Bayes models are trained the same way as for Aladin.

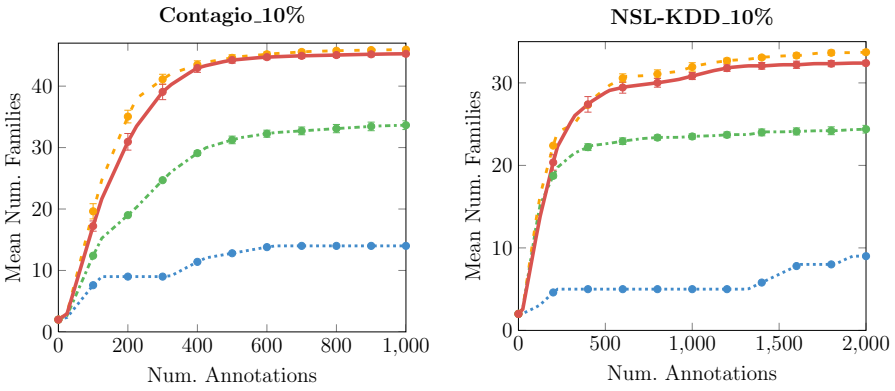
5.3 Results

The datasets Contagio_10% and NSL-KDD_10% are split uniformly into two datasets: (1) an active learning dataset (90%) used as a pool to build the labelled dataset $\mathcal{D}_L$, and (2) a validation dataset (10%) to assess the performance of the detection model trained on $\mathcal{D}_L$. The different labelling strategies are compared with simulations where the annotation queries are answered by an oracle providing the ground truth labels and families.

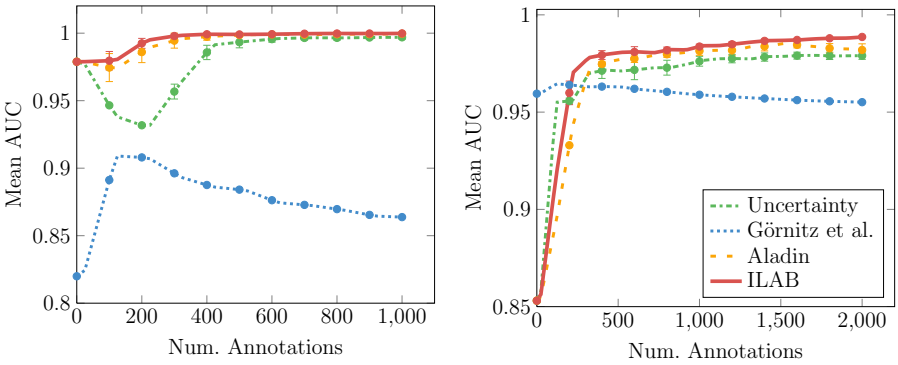
All the strategies are run with $b = 100$ annotations at each iteration. The annotation budget is set to $B = 1000$ for Contagio_10%, and to $B = 2000$ for NSL-KDD_10% as this dataset contains more instances. The initial labelled datasets are composed of instances belonging to the most represented families: 7 malicious instances and 13 benign instances.

All the experiments are run on Linux 3.16 on a dual-socket computer with 64Go RAM. Processors are Intel Xeon E5-5620 CPUs clocked at 2.40 GHz with 4 cores each and 2 threads per core. Each labelling strategy is run 15 times and we report the average performance with the 95% confidence interval.

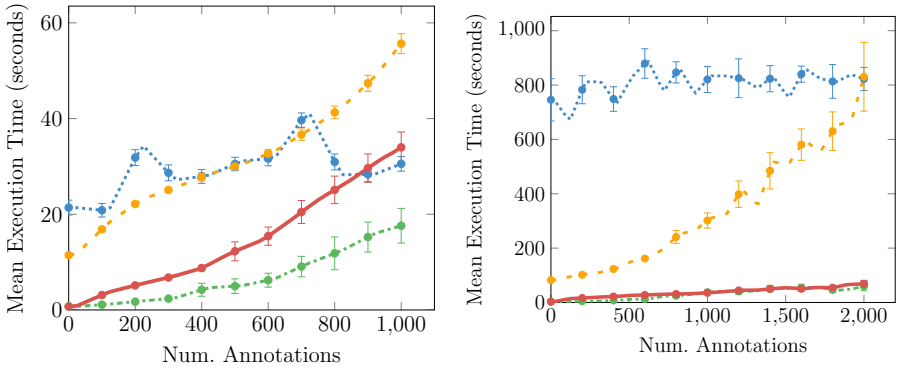
First, we compare the number of known families across the iterations to assess sampling bias (see Figure 4a). Then, we compare the performance of the detection models on the validation dataset (see Figure 4b). Finally, we monitor the execution time of the query generation algorithms to evaluate the expert waiting time between iterations (see Figure 4c).



(a) Average Number of Families Discovered



(b) Average Detection performance (AUC) on the Validation Dataset



(c) Average Annotation Queries Generation Execution Time

Fig. 4: Comparison of the labelling strategies

Contagio_10% (on the left) and NSL-KDD_10% (on the right)

Families Detection. Figure 4a shows that uncertainty sampling and Görnitz et al. labelling strategy miss many families during the annotation process. Both labelling strategies suffer from sampling bias. Görnitz et al. labelling strategy relies on k-nearest neighbours to detect yet unknown malicious families but only close to the decision boundary, that is why many families further from the decision boundary are not discovered. Their strategy to foster the discovery of yet unknown families is not effective on both datasets.

ILAB dedicates only a part of its annotation budget to the detection of yet unknown families, that is why Aladin detects slightly more families than ILAB. ILAB queries some high likelihood instances which are unlikely to belong to new families, but they allow to keep the detection performance increasing across the iterations (see Figure 4b).

ILAB and Aladin discover about as many families across the iterations on both datasets. These labelling strategies are effective at avoiding sampling bias. They are designed to detect rare categories, and they are able to discover almost all the families on both datasets.

Detection Performance. Figure 4b represents the evolution of the Area Under the Curve (AUC) [16] on the validation dataset. It shows that ILAB performs better than the other labelling strategies on both datasets.

Görnitz et al. labelling strategy performs very poorly on Contagio_10%. The detection performance increases at the first iteration, but then it keeps on decreasing when new instances are added to the labelled dataset. This peculiar behaviour can be explained by the simplicity of the SVDD detection model which cannot discriminate the benign from the malicious instances properly. The geometry of the data prevents SVDD from isolating the benign instances from the malicious instances in a sphere. We notice the same behaviour less pronounced on NSL-KDD_10%. A solution to address this issue is to train SVDD with a kernel to increase the complexity of the model. However, this solution will considerably increase the execution time which is already too high to ensure a good expert-model interaction (see Figure 4c).

Görnitz et al. labelling strategy performs much better initially on NSL-KDD_10% than the other labelling strategies. Indeed, thanks to semi-supervision, Görnitz et al. use not only the 20 initial labelled instances to train their detection model, but also all the instances from the unlabelled pool. Görnitz et al. semi-supervised detection model is, however, not as effective as logistic regression initially on Contagio_10%. SVDD makes the assumption that the unlabelled instances are mostly benign, and so the malicious instances in the unlabelled pool may damage the detection model performance.

Uncertainty sampling has a better detection performance than ILAB during the first iterations on NSL-KDD_10% because it allocates all its annotation budget to refining the decision boundary. On the contrary, ILAB dedicates 90% of its annotation budget to rare category detection to avoid sampling bias. In the end, uncertainty sampling suffers from sampling bias and converges to a poorer performance.

The detection performance of uncertainty sampling and Aladin decreases during the first iterations on Contagio_10%. This undesirable behaviour is caused by sampling bias: non-representative instances are queried for annotation, added to the training dataset and prevent the detection model from generalizing properly. Uncertainty sampling queries instances close to the decision boundary that are hard to classify for the detection model, but not representative of the malicious or benign behaviours. Aladin queries only uncertain and low likelihood instances which are not necessarily representative of the malicious and benign behaviours either. ILAB addresses this problem by dedicating a part of its annotation budget to high likelihood instances to get representative examples of each family. Therefore, the detection performance keeps on increasing across the iterations.

Scalability. Figure 4c depicts the query generation execution time (in seconds) across the iterations. Görnitz et al. query generation algorithm is very slow. For NSL-KDD_10%, the expert waits more than 10 minutes between each iteration while the labelling strategy computes the annotation queries. A third of the execution time corresponds to the computation of the semi-supervised SVDD model, and the remaining two thirds corresponds to the k-nearest neighbour algorithm. The execution time of Görnitz et al. labelling strategy is thus too high to ensure a good expert-model interaction even on a dataset containing fewer than 100,000 instances.

ILAB has an execution time comparable to uncertainty sampling. For NSL-KDD_10%, the expert waits less than 1 minute between each iteration. On the contrary, Aladin execution time increases drastically when new instances are added to the labelled dataset and new families are discovered. Aladin runs rare category detection on all the instances, while ILAB runs it on the malicious and the benign instances separately. ILAB divide and conquer approach reduces the execution time as running rare category detection twice on smaller datasets with fewer families is faster than running it on the whole dataset. Aladin's authors were aware of this high execution time. During their experiments, the expert was asked to annotate 1000 instances each day, and the new annotation queries were computed every night. Their solution reduces the expert waiting time, but it significantly damages the expert-model interaction since the expert feedback is integrated only once a day.

In conclusion, uncertainty sampling and Görnitz et al. labelling strategy suffer from sampling bias. Aladin and ILAB are the only labelling strategies able to avoid sampling bias thanks to rare category detection performed at the family level (see Figure 4a). ILAB main advantage over Aladin is its divide and conquer approach that significantly reduces the execution time (see Figure 4c) and thus improves the expert-model interaction. Our comparisons show that ILAB is both an effective and a scalable labelling strategy that can be set up on real-world annotation projects.

6 Real-World Annotation Project on NetFlow Data

In this section, we deploy ILAB on a large unlabelled NetFlow dataset originating from a production environment.

NetFlow. As stated in [5]: "NetFlow is a network protocol proposed and implemented by Cisco [6] for summarizing network traffic as a collection of network flows. A flow is defined as a unidirectional sequence of packets that share specific network properties (e.g. IP source/destination addresses, and TCP or UDP source/destination ports)." Each flow is described by attributes and summary statistics: source and destination IP addresses, source and destination ports, protocol (TCP, UDP, ICMP, ESP, etc.), start and end time stamps, number of bytes, number of packets, and aggregation of the TCP flags for TCP flows.

Dataset and Features. The flows are recorded at the border of a defended network. We compute features describing each external IP address communicating with the defended network. from its flows during a given time window. We compute the mean and the variance of the number of bytes and packets sent and received at different levels: globally, for some specific port numbers (80, 443, 53 and 25), and for some specific TCP flags aggregates (`....S`, `.A..S.`, `.AP.SF`, etc.). Besides, we compute other aggregated values: number of contacted IP addresses and ports, number of ports used, entropy according to the contacted IP addresses and according to the contacted ports. In the end, each external IP address is described by 134 features computed from its list of flows.

Num. flows	$1.2 \cdot 10^8$
Num. IP addresses	$463,913$
Num. features	134
Num. TRW alerts	70

Table 2: NetFlow Dataset

The NetFlow data is recorded during a working day in 2016. The features are computed for each external IP address with a 24-hour time window. The NetFlow dataset is large: it is composed of 463,913 IP addresses represented by 134 real-valued features (see Table 2). A second dataset has been recorded the following day for the validation of the resulting detection model. The results are, however, not reported due to space constraints since the main focus is the deployment of the labelling strategy in an annotation project.

ILAB Graphical User Interface. A security expert answers ILAB annotation queries from the graphical user interface depicted in Figure 5. The top buttons allow the expert to select a type of annotation queries: *Uncertain* for the instances near the decision boundary, *Malicious* and *Benign* for the annotation queries generated by rare category detection. The panel below allows to go through the annotation queries corresponding to each family.

By default, each instance is described only by its features which may be hard to interpret, especially when they are in high dimension. A custom visualization which may point to external tools or information can be displayed to ease the

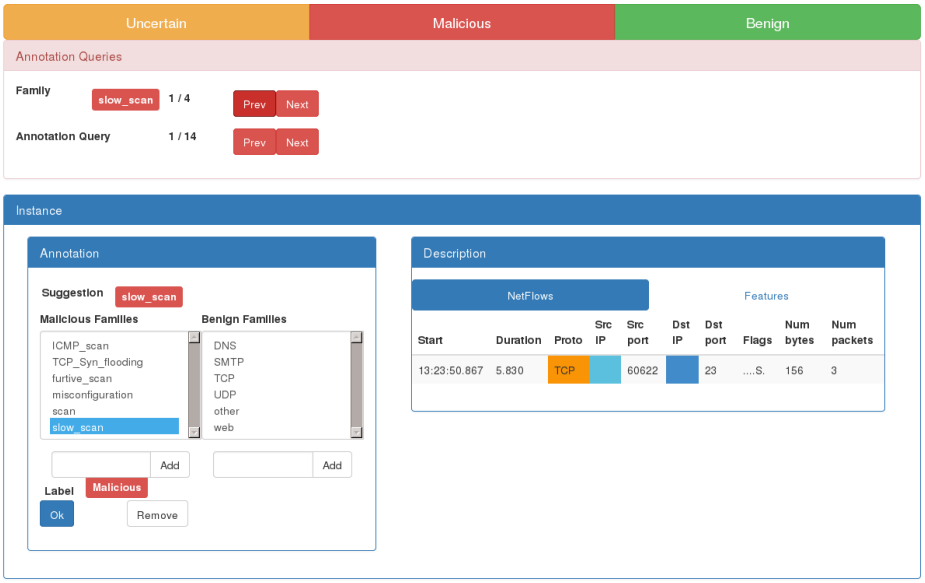


Fig. 5: ILAB Graphical User Interface for Annotating

annotations. Figure 5 depicts the custom visualization we have implemented for NetFlow data[5].

Finally, the expert can annotate the selected instance with the *Annotation* panel. For each label, it displays the list of the families already discovered. The expert can pick a family among a list or add a new family. The interface suggests a family for high likelihood queries and pre-selects it. It helps the expert since the model is confident about these predictions. On the contrary, there is no suggestion for the uncertainty sampling and the low likelihood queries. The model is indeed uncertain about the family of these instances and unreliable suggestions may mislead the expert [3].

ILAB in Practice. First, we need some labelled instances to initialize the active learning process. The alerts raised by the Threshold Random Walk (TRW) [18] module of Bro [24] provide the initial anomalous examples and the normal examples are drawn randomly. The initial labelled dataset is composed of 70 *obvious scans* detected by TRW, and of 70 normal examples belonging to the *Web*, *SMTP* and *DNS* families. Malicious activities in well-established connections cannot be detected without the payload, which is not available in NetFlow data, that is why we consider the families *Web*, *SMTP* and *DNS* to be normal. All the initial labels are checked individually by the expert to avoid poisoning the model.

This initial labelled dataset is not representative of all the anomalous behaviours we want to detect. We run ILAB with the parameters $B = 1000$, $b = 100$

[5] The IP addresses have been hidden for privacy reasons.

and $b_{\texttt{uncertain}} = 10$ to acquire a representative labelled dataset. Across the iterations, ILAB has discovered stealthier scans: *ICMP scans*, *slow scans* (only one flow with a single defended IP address contacted on a single port), *furtive scans* (a slow scan in parallel with a well-established connection). Besides, it has detected *TCP Syn flooding* activities designed to exhaust the resources of the defended network. Finally, ILAB has asked the expert to annotate IP addresses with anomalous behaviours which are not malicious: *misconfigurations* and *backscatters*.

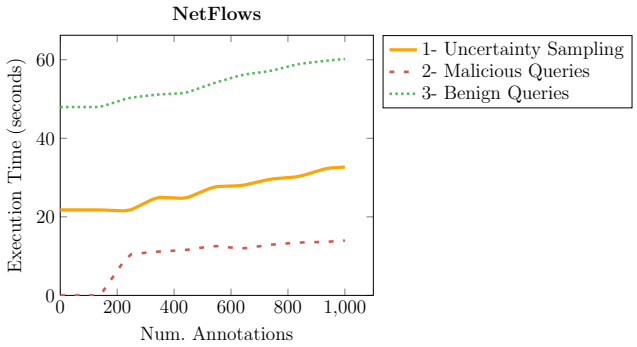


Fig. 6: ILAB Execution Time

Low Expert Waiting Time. ILAB divide and conquer approach allows the expert to annotate some instances while the labelling strategy is still computing annotation queries. First, the binary detection model is trained and the uncertainty sampling queries are computed. The binary detection model is indeed required to predict the label of the unlabelled instances to run rare category detection afterwards. Then, rare category detection is performed on the malicious predictions while the expert annotates the uncertain instances. Finally, rare category detection is computed on the benign predictions while the expert annotates the malicious annotation queries. The malicious predictions are analysed before the benign ones, because their number is smaller, so the analysis is faster (see Figure 6).

In practice, running rare category detection takes less time than the annotations. As a result, the expert must only wait while the uncertain queries are computed (see the orange curve *Uncertainty Sampling* in Figure 6). During the NetFlow annotation project the expert has waited less than 40 seconds at each iteration. ILAB low computation cost ensures a good expert-model interaction: the detection model is updated frequently with expert feedback without inducing long waiting-periods.

Families Benefits. ILAB and Aladin deal with the sampling bias problem thanks to rare category detection performed at the family level. At first glance, this solution may seem to increase the annotation cost as it requires experts to provide

a more precise information than a binary label. However, asking experts to provide a family does not increase the annotation cost in practice: experts place instances in "mental bins" corresponding to families to provide a label [26]. Experts must understand the type of the instance to provide a label, and, therefore, assigning a family does not require an additional effort.

Besides, the clustering of the annotation queries according to families (see Figure 5) decreases the average annotation cost. Families provide a context that helps the expert answer the queries. Annotation queries related to the same family are likely to share the same label and family, and thus, it reduces the amount of context switching during the annotation process. On the contrary, uncertainty sampling and Görnitz et al. labelling strategy ask the expert to annotate a list of unrelated instances without any context.

Finally, an alert raised by a supervised detection model can be hard to interpret for the security expert. This issue called semantic gap by Sommer et al. [38] is due to the binary output (`Malicious` or `Benign`) of the detection model. The families acquired with ILAB can bridge the semantic gap by enriching the alerts with a malicious family to help the expert supervising the detection system take the necessary actions.

7 Conclusion

We introduce ILAB a novel interactive labelling strategy that streamlines annotation projects. It relies on active learning and rare category detection to avoid sampling bias. We demonstrate that ILAB offers a better scalability than two state-of-the-art labelling strategies [14, 40] without damaging the effectiveness. Up to our knowledge, [40] and [14] had never been compared. We provide open source implementations to foster comparison in future research works.

ILAB divide and conquer approach reduces the computation cost, and allows the expert to annotate some instances while the labelling strategy is still computing annotation queries. Thus, ILAB provides a good expert-model interaction: the detection model is updated frequently with expert feedback without inducing long waiting-periods.

The NetFlow annotation project shows that ILAB is a workable labelling strategy that can be applied to a large dataset originating from a production environment. ILAB is a generic labelling strategy that can be applied to other detection problems once the feature extraction task has been performed. It is designed for security experts who deploy intrusion detection systems, and we provide an open source implementation of the graphical user interface to allow them to label their own datasets. For future work, we plan to run broader experiments with independent computer security experts to assess ILAB from an end-user's point of view and to improve its usability from their feedback.

References

1. Almgren, M., Jonsson, E.: Using active learning in intrusion detection. In: CSFW. pp. 88–98 (2004)

2. Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., Dagon, D.: From throw-away traffic to bots: detecting the rise of DGA-based malware. In: USENIX Security. pp. 491–506 (2012)
3. Baldridge, J., Palmer, A.: How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation. In: EMNLP. pp. 296–305 (2009)
4. Berlin, K., Slater, D., Saxe, J.: Malicious behavior detection using windows audit logs. In: AISEC. pp. 35–44 (2015)
5. Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In: ACSAC. pp. 129–138 (2012)
6. Claise, B.: Cisco systems netflow services export version 9 (2004)
7. Corona, I., Maiorca, D., Ariu, D., Giacinto, G.: Lux0r: Detection of malicious PDF-embedded JavaScript code through discriminant analysis of API references. In: AISEC. pp. 47–57 (2014)
8. Dasgupta, S., Hsu, D.: Hierarchical sampling for active learning. In: ICML. pp. 208–215 (2008)
9. Druck, G., Settles, B., McCallum, A.: Active learning by labeling features. In: EMNLP. pp. 81–90 (2009)
10. Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning, vol. 1. Springer series in statistics, Berlin (2001)
11. Gascon, H., Yamaguchi, F., Arp, D., Rieck, K.: Structural detection of android malware using embedded call graphs. In: AISEC. pp. 45–54 (2013)
12. Görnitz, N., Kloft, M., Brefeld, U.: Active and semi-supervised data domain description. In: ECML-PKDD. pp. 407–422 (2009)
13. Görnitz, N., Kloft, M., Rieck, K., Brefeld, U.: Active learning for network intrusion detection. In: AISEC. pp. 47–54 (2009)
14. Görnitz, N., Kloft, M.M., Rieck, K., Brefeld, U.: Toward supervised anomaly detection. JAIR (2013)
15. Hachey, B., Alex, B., Becker, M.: Investigating the effects of selective sampling on the annotation task. In: CoNLL. pp. 144–151 (2005)
16. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143(1), 29–36 (1982)
17. Jones, E., Oliphant, T., Peterson, P.: SciPy: open source scientific tools for Python (2014)
18. Jung, J., Paxson, V., Berger, A.W., Balakrishnan, H.: Fast portscan detection using sequential hypothesis testing. In: S&P. pp. 211–225 (2004)
19. Khasawneh, K.N., Ozsoy, M., Donovick, C., Abu-Ghazaleh, N., Ponomarev, D.: Ensemble learning for low-level hardware-supported malware detection. In: RAID, pp. 3–25 (2015)
20. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: SIGIR. pp. 3–12 (1994)
21. Miller, B., Kantchelian, A., Afroz, S., Bachwani, R., Dauber, E., Huang, L., Tschantz, M.C., Joseph, A.D., Tygar, J.: Adversarial active learning. In: AISEC. pp. 3–14 (2014)
22. Nappa, A., Rafique, M.Z., Caballero, J.: The MALICIA dataset: identification and analysis of drive-by download operations. IJIS 14(1), 15–33 (2015)
23. Omohundro, S.M.: Five balltree construction algorithms. International Computer Science Institute Berkeley (1989)
24. Paxson, V.: Bro: a system for detecting network intruders in real-time. Computer networks 31(23), 2435–2463 (1999)

25. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. JMLR 12, 2825–2830 (2011)
26. Pelleg, D., Moore, A.W.: Active learning for anomaly and rare-category detection. In: NIPS. pp. 1073–1080 (2004)
27. Rieck, K.: Computer security and machine learning: Worst enemies or best friends? In: SysSec. pp. 107–110 (2011)
28. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics 20, 53–65 (1987)
29. Schütze, H., Velipasaoglu, E., Pedersen, J.O.: Performance thresholding in practical text classification. In: CIKM. pp. 662–671 (2006)
30. Sculley, D.: Online active learning methods for fast label-efficient spam filtering. In: CEAS. pp. 1–4 (2007)
31. Sculley, D., Otey, M.E., Pohl, M., Spitznagel, B., Hainsworth, J., Zhou, Y.: Detecting adversarial advertisements in the wild. In: KDD. pp. 274–282 (2011)
32. Settles, B.: Active learning literature survey. University of Wisconsin, Madison 52(55-66), 11 (2010)
33. Settles, B.: From theories to queries: Active learning in practice. JMLR 16, 1–18 (2011)
34. Settles, B.: Active learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 6(1), 1–114 (2012)
35. Smutz, C., Stavrou, A.: Malicious PDF detection using metadata and structural features. In: ACSAC. pp. 239–248 (2012)
36. Smutz, C., Stavrou, A.: Malicious PDF detection using metadata and structural features. In: Technical Report. George Mason University (2012)
37. Snow, R., O'Connor, B., Jurafsky, D., Ng, A.Y.: Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In: EMNLP. pp. 254–263 (2008)
38. Sommer, R., Paxson, V.: Outside the closed world: On using machine learning for network intrusion detection. In: S&P. pp. 305–316 (2010)
39. Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., Nakao, K.: Statistical analysis of honeypot data and building of kyoto 2006+ dataset for NIDS evaluation. In: BADGERS. pp. 29–36 (2011)
40. Stokes, J.W., Platt, J.C., Kravis, J., Shilman, M.: Aladin: Active learning of anomalies to detect intrusions. Technical Report. Microsoft Network Security Redmond, WA (2008)
41. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: CISDA (2009)
42. Tax, D.M., Duin, R.P.: Support vector data description. Machine learning 54(1), 45–66 (2004)
43. Tomanek, K., Olsson, F.: A web survey on the use of active learning to support annotation of text data. In: ALNLP. pp. 45–48 (2009)
44. Veeramachaneni, K., Arnaldo, I.: AI2: Training a big data machine to defend. In: DataSec. pp. 49–54 (2016)
45. Whittaker, C., Ryner, B., Nazif, M.: Large-scale automatic classification of phishing pages. In: NDSS. vol. 10 (2010)
46. Wright, S., Nocedal, J.: Numerical optimization. Springer Science 35, 67–68 (1999)
47. Zhang, T., Oles, F.: The value of unlabeled data for classification problems. In: ICML. pp. 1191–1198 (2000)