End-to-End Active Learning for Computer Security Experts

Anaël Beaugnon, Pierre Chifflier, Francis Bach

To cite this version:

Anaël Beaugnon, Pierre Chifflier, Francis Bach. End-to-End Active Learning for Computer Security Experts. AAAI Workshop on Artificial Intelligence for Cyber Security (AICS), Feb 2018, New Orleans, United States. hal-01888976

HAL Id: hal-01888976
https://hal.science/hal-01888976

Submitted on 19 Oct 2018

End-to-End Active Learning for Computer Security Experts

Anaël Beaugnon[1,2], Pierre Chifflier[1], 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

Labelling a dataset for supervised learning is particularly expensive in computer security as expert knowledge is required for annotation. Some research works rely on active learning to reduce the labelling cost, but they often assimilate annotators to mere oracles providing ground-truth labels. Most of them completely overlook the user experience while active learning is an interactive procedure. In this paper, we introduce an end-to-end active learning system, ILAB, tailored to the needs of computer security experts. We have designed the active learning strategy and the user interface jointly to effectively reduce the annotation effort. Our user experiments show that ILAB is an efficient active learning system that computer security experts can deploy in real-world annotation projects.

Introduction

Supervised detection models must be trained on representative annotated datasets which are particularly expensive to build in computer security. Expert knowledge is required for annotation and data are often confidential. As a result, crowd-sourcing cannot be applied as in computer vision or natural language processing to acquire annotated datasets at low cost. Some annotated datasets related to computer security are public but they quickly become outdated and they often do not account for the idiosyncrasies of each deployment context. Computer security experts can build a representative training dataset *in-situ*: they annotate some instances picked from a pool of unlabelled data originating from the deployment environment. Experts are essential for annotating but they are an expensive resource. The labelling process must thus use expert time efficiently.

Active learning strategies (Settles 2010) have been introduced in the machine learning community to reduce the number of manual annotations. However, most research works on active learning overlook the user experience: they assume that annotators are mere oracles providing ground-truth labels (Settles 2011; Wagstaff 2012). Active learning is an interactive procedure where a user interface is needed to gather the annotations and to provide feedback showing the usefulness of the annotations. Besides, the user interface must be suitable to computer security experts who are usually not machine learning experts. Finally, the annotators should not waste their time waiting while the next annotation queries are computed.

We introduce the concept of *active learning system*: an active learning strategy integrated in a user interface. It is crucial to design both components jointly to effectively reduce experts annotation effort and to foster the adoption of active learning in annotation projects (Mac Aodha et al. 2014; Baldridge and Palmer 2009; Tomanek and Olsson 2009).

We introduce ILAB, an end-to-end active learning system, to bridge the gap between theoretical active learning and real-world annotation projects. ILAB is designed to help computer security experts to build annotated datasets with a reduced effort. We make the following contributions:

- We present ILAB, an end-to-end active learning system composed of an active learning strategy integrated in a user interface. The active learning strategy is designed to reduce experts waiting-periods: they can annotate some instances while the algorithm is computing new annotation queries. The graphical user interface is tailored to the needs of computer security experts. It has been designed for annotators who may have little knowledge about machine learning, and it can manipulate any data type (PDF or doc files, Windows audit logs, or NetFlow data for example).
- We ask intended end-users, computer security experts, to use ILAB on a large unlabelled NetFlow dataset coming from a production environment. These user experiments validate our design choices and highlight potential improvements.
- We provide an open source implementation of the whole active learning system, github.com/ANSSI-FR/SecuML, to foster comparison in future research works, and to allow computer security experts to annotate their own datasets.

The rest of the paper is organized as follows. First, we present some related works about active learning systems. Second, we formalize the problem and introduce the notations. Third, we describe ILAB, an end-to-end active learning system designed for computer security annotation projects. Finally, we present user experiments carried out on a real-world annotation project.

Related Work

Active Learning Systems

(Amershi et al. 2014) and (Settles 2010) describe generic guidelines that any annotation system should follow.

First of all, annotation systems must *offer an ergonomic user interface* to display the queries and to gather the corresponding answers. Special care shall be taken to the ergonomics of the interface as it may greatly affect the overall annotation time. Besides, annotation systems are intended for application domain experts who are likely to have no or little knowledge about machine learning. Consequently, the user interface must be *accessible to non-machine learning experts*.

Moreover, people will invest time to improve their model only if they view the task as more beneficial than costly. Annotation systems must *provide feedback to users to show them the benefit of their annotations*, and that they are on track to achieve their goal.

At each iteration, the integration of new annotations improves not only the performance of the detection model but also the relevance of the following queries. Frequent updates are thus beneficial, but they must be performed efficiently to *minimize waiting-periods*. An annotation system with long waiting-periods would alter the human-model interaction and is unlikely to be accepted by experts.

Some research works have introduced whole annotation systems but they are especially designed for image (Kulesza et al. 2014), video (Ayache and Quénot 2008) or text (Skeppstedt, Paradis, and Kerren 2017) annotations. Computer security detection problems are not based one these conventional data types but on PDF, doc, or exe files, NetFlow, pcap, or event logs for example. As a result, the annotation system must be generic and easily adapted to operate with these diverse data types.

Applications to Computer Security

(Amershi et al. 2011) introduces CueT, an interactive machine learning system to triage network alarms. This system works in dynamic, continually evolving environments, but it cannot be used with any data type.

ILAB active learning strategy has been presented in (Beaugnon, Chifflier, and Bach 2017) and compared to other methods designed for computer security (Almgren and Jonsson 2004; Görnitz et al. 2013; Stokes et al. 2008). It avoids the sampling bias issue (Schütze, Velipasaoglu, and Pedersen 2006) affecting (Almgren and Jonsson 2004) and (Görnitz et al. 2013) and reduces the waiting-periods compared to (Stokes et al. 2008).

(Almgren and Jonsson 2004) and (Görnitz et al. 2013) have not set up their strategy in a real-world annotation project, and they have not mentioned any user interface. (Stokes et al. 2008) have carried out user experiments with computer security experts, but they provide no detail about the user interface. Besides, the interactions between the expert and the model were poor due to a high execution time. The expert was asked to annotate 1000 instances each day, and new queries were computed every night. Computer security experts are unlikely to accept to perform so many annotations without getting any feedback about their usefulness.

In this paper, we focus on the whole annotation system with an active learning strategy and a graphical user interface. On the one hand, the active learning strategy is designed to reduce the waiting-periods. On the other hand, the user interface is convenient for non-machine learning experts and provides feedback to show the benefit of the annotations. We do not compare ILAB active learning system to (Almgren and Jonsson 2004; Görnitz et al. 2013; Stokes et al. 2008) since they have not designed or they provide too few details about the user interface.

Problem Statement

Context and 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 or the traffic of an IP address. 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).

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 pool-based 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 whole process, the expert cannot annotate more instances than the annotation budget $B \in \mathbb{N}$.

We assume that there is no adversary attempting to mislead the annotation process: a trusted computer security expert performs the annotation offline before the detection model is deployed in production.

Objective Our goal is to conceive an end-to-end active learning system intended for computer security experts who deploy detection systems. It must be composed of a query strategy and a corresponding user interface that fulfil the following constraints.

First, the annotation system must be able to deal with different types of data (PDF, doc files, pcap, NetFlow, etc.). Besides, the design of the active learning strategy must reduce the waiting-periods induced by the generation of the queries. Finally, the user interface must be suitable for non-machine learning experts and it should provide feedback frequently to show the benefit of the annotations.

ILAB

Active Learning Strategy

ILAB active learning strategy has been presented in (Beaugnon, Chifflier, and Bach 2017). We describe it briefly to allow the reader to understand the design of the whole active learning system.

The active learning strategy relies on a two-level label hierarchy: binary labels (`Malicious` vs. `Benign`) and families of instances sharing similar behaviors.

At each iteration, a binary detection model $\mathcal{M}$ is trained from the binary labels of the currently annotated instances (see Figure 1). By default a logistic regression model is trained as the coefficients allow to easily interpret the predictions and training is fast. If this model class is not complex enough for a given annotation project, other model classes can be plugged into ILAB but will result in longer waiting-periods and a loss of understandability.

Once $\mathcal{M}$ has been trained, the b annotation queries are computed. On the one hand, the $b_{\texttt{uncertain}}$ instances the closest to the decision boundary are queried as in uncertainty sampling (Lewis and Gale 1994). One the other hand, $b_{\texttt{families}}$ $(= b - b_{\texttt{uncertain}})$ instances are queried by means of rare category detection (Pelleg and Moore 2004).

Not all families are present in the initial labelled dataset and rare category detection fosters the discovery of yet unknown families to avoid sampling bias. 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. 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, 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 $\mathcal{D}_U^{\texttt{Malicious}}$. Then, the families are modelled with Gaussian Naive Bayes and two kinds of instances are queried from each family: 1) *low likelihood* instances to foster the discovery of yet unknown families, and 2) *high likelihood* instances to make sure the model is not confidently wrong.

More and more families are discovered and added to the annotated dataset across iterations. When a new family is discovered, it is taken into acount in rare category detection at the next iteration: the additional family is included in the training of the multi-class logistic regression and the Gaussian Naive Bayes models.

Short Waiting-Periods The generation of the different kinds of queries (uncertain, malicious and benign) are completely independent (see Figure 1). This reduces the expert waiting-periods in two ways: 1) the computations can be parallelized, and, 2) the expert can start annotating while the remaining queries are generated.

Active Learning System

Initialization The active learning process needs some initial labelled examples to learn the first supervised detection model. 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. We use ILAB to enrich the initial labelled dataset with more diverse malicious behaviors and to make it representative of the environment where the detection system is deployed.

Annotation Interface ILAB comes up with an annotation interface intended for non-machine learning experts.

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 for a given type of instances can be plugged into ILAB to ease the annotations considerably. The custom visualization should display the most important elements to make a decision, and it may point to external tools or information to provide some context. The custom visualization makes the annotation interface generic regarding the data types.

Experts can annotate the selected instance with the *Annotation* panel. For each label, it displays the list of the families already discovered. Experts 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 experts since the model is confident about these predictions. On the contrary, there is no suggestion for uncertain and low likelihood queries. The model is indeed unsure about the family of these instances and unreliable suggestions may mislead experts (Baldridge and Palmer 2009). The next query is displayed automatically after each annotation validation.

Monitoring Interface The annotation system must provide feedback to users to show them the benefit of their annotations, and that they are on track to achieve their goal. In simulated experiments (with an oracle answering the queries with the ground truth labels), the performance of the detection model $\mathcal{M}$ on an independent validation dataset is usually reported. This approach is, however, not applicable in a real-world setting: when you deploy an annotation system to build a training dataset you do not have access to an annotated validation dataset.

ILAB offers two kinds of feedback across iterations: 1) the number of malicious and benign families discovered so far, and, 2) the accuracy of the suggested labels/families. At

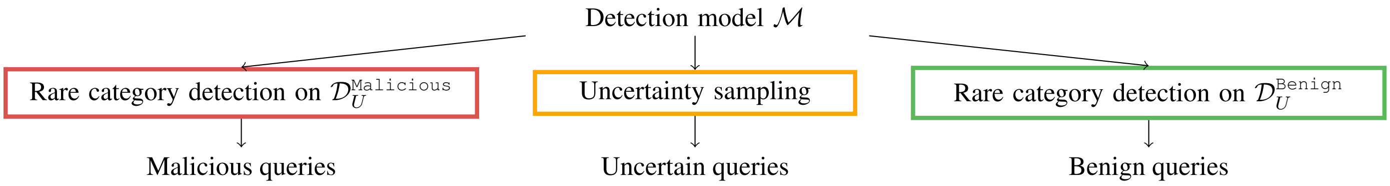

Detection model $\mathcal{M}$
Rare category detection on $\mathcal{D}_U^{\texttt{Malicious}}$
Uncertainty sampling
Rare category detection on $\mathcal{D}_U^{\texttt{Benign}}$
Malicious queries
Uncertain queries
Benign queries

Figure 1: Parallelization of the computations of the annotation queries

each iteration, ILAB suggests a family for the high likelihood queries. At the next iteration, the accuracy of these suggestions can be computed according to the last annotations performed by the expert.

This feedback can provide insight into the impact of new annotations. If the number of families discovered and the accuracy of the suggestions are stable for several iterations, it may be unnecessary to go on annotating.

The monitoring interface is modular to allow annotators to go more in depth into the behaviour of the detection model if they wish. For instance, they can display the most important features with their corresponding weights, the training and the cross validation performance (detection rate, false alarm rate, F-score, ROC, AUC, confusion matrix) of the detection model. This additional information is not displayed in the default view since it is not crucial for computer security experts involved in annotation projects and it may even lead to confusion.

Annotated Instances and Family Editor At the beginning of the annotation procedure, computer security experts have little knowledge about the dataset. Answering annotation queries increases their knowledge about the dataset and may change their decision-making process. Experts must, however, provide consistent annotations throughout the whole annotation process not to mislead the detection model.

ILAB offers two graphical user interfaces to help experts to remain consistent with their previous annotations. First, an interface displays the currently annotated instances grouped according to their associated label or family. Second, a family editor, similar to the one presented in (Kulesza et al. 2014), enables annotators to perform three actions over the families: `Swap Malicious / Benign`, `Change Name`, and `Merge Families`. Annotators can clarify the name of a family, regroup similar families or change the label corresponding to a given family.

Computer security experts work on diverse data types. A strength of ILAB is to be generic, so that they can use a unique annotation system. Once they get used to ILAB on a given detection problem, they will be more efficient at using it on other detection problems.

User Experiments

In this section, we collect feedback from intended end-users to validate our design choices. We aim to show that ILAB can be deployed in real-world annotation projects on large datasets with annotators who are not machine learning experts.

	Day 1	Day 2
Number of flows	$1.2 \cdot 10^8$	$1.2 \cdot 10^8$
Number of IP addresses	463913	507258
Number of features	134	134
Number of TRW alerts	72	82

Table 1: NetFlow Datasets

We ask computer security experts to acquire an annotated dataset from NetFlow data coming from a production environment with ILAB. We take a large NetFlow dataset as example but ILAB is a generic solution that can be applied on any data type (Beaugnon, Chifflier, and Bach 2017).

Annotation Project

The annotation project consists in acquiring a labelled dataset from unlabelled NetFlow data in order to learn a supervised detection model detecting IP addresses with an anomalous behaviour.

The flows are recorded at the border of a defended network. 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.

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.

The NetFlow data is recorded during two consecutive working days in 2016 (see Table 1). The features are computed for each external IP address with a 24-hour time window. The dataset `Day 1` constitutes the unlabelled pool from which some instances are queried for annotation, and the dataset `Day 2` serves as a validation dataset to analyse the alerts raised by the resulting detection model.

The active learning process is initialized with some annotated instances. The alerts raised by the Threshold Random Walk (TRW) (Jung et al. 2004) module of Bro (Paxson 1999) provide the initial anomalous examples and the normal examples are drawn randomly. All the initial labels are checked manually. 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.

ILAB is deployed to enrich this initial labelled dataset. The detection model should not be restricted to the detection of obvious scans, additional anomalous behaviours must be identified from the NetFlow data with ILAB.

Experimental Protocol

Four computer security experts take part in the experiments. They are used to working with NetFlow data, but they have no or little knowledge about machine learning. They have never used ILAB or any other annotation system before. The experiments are carried out independently with each expert for half a day.

The task is divided into two parts. First, the users acquire an annotated dataset with ILAB from the unlabelled pool `Day 1`. Then, they analyse the alerts raised on `Day 2` by the detection model trained on the annotated instances. Once the task is completed, we collect their feedback.

All the experiments are run 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.

ILAB Deployment ILAB active learning strategy has only two parameters: b the number of annotation queries answered at each iteration, and $b_{\texttt{uncertain}}$ the number of uncertain queries. b controls the trade-off between reducing waiting-periods and integrating expert feedback frequently. We set its value to $b = 100$ on the following principle: experts should not spend more time waiting for queries than annotating, but their feedback must still be integrated rather frequently to show them the benefit of their annotations. Some instances near the decision boundary are annotated to help the detection model make decision about these instances, but not too many since they are often harder to annotate (Baldridge and Palmer 2009; Settles 2011), and they may lead to sampling bias (Schütze, Velipasaoglu, and Pedersen 2006). As a result, we run ILAB with the parameters $b = 100$ and $b_{\texttt{uncertain}} = 10$.

The global annotation budget B is not set, but we stop the annotations after 90 minutes while letting annotators complete their current iteration. We timestamp and log all the users actions in the interface to assess the time required for annotating and the waiting-periods.

Before launching the annotation process, we ask the experts to check the initial annotated instances, and tell them that them may change the assigned labels and families as they wish. This step is crucial to ensure that the annotations they perform afterwards are consistent with the initial ones.

The first two participants have no information about the features of the detection model for the purposes of hiding the machine learning complexity. However, this approach may lead annotators to create families which cannot be properly discriminated by the model due to a lack of information in the features extracted from the NetFlow data. The last two participants know the features of the model, and we briefly explain the implications on the families they may create. Port numbers are a relevant example. The features include the number of bytes and packets sent and received globally, and for the port numbers 80, 443, 53 and 25. We emphasize that it is therefore counterproductive to create families corresponding to scans on specific services such as *Telnet scans* or *SSH scans*.

Alerts Analysis Once a dataset has been annotated with ILAB from `Day 1` data, we train a detection model with it, and ask the expert to analyse the alerts raised on `Day 2` data (as if the model is deployed in production). This step is crucial: the objective of computer security experts is not to acquire an annotated dataset, but to build a detection model and to assess its performance.

Feedback Collection Once the users have achieved the task, we collect their feedback through an informal discussion that covers the following topics: the relevance of the alerts raised by the resulting detection model, the ease of use of the interface, the waiting-periods, the usefulness of the *Family Editor* and *Annotated Instances* interfaces, and the feedback provided across iterations.

Results

Accessible to Non-Machine Learning Experts The participants have not faced any difficulty in building a detection model with ILAB even if they have little or no knowledge about machine learning. They have reported some minor ergonomic problems not related to machine learning especially. These issues will be addressed to further improve the user experience. Globally, the participants have been pleased with ILAB, and convinced that it will be beneficial to other annotation projects.

Annotation Queries and Answers Across iterations, ILAB has detected stealthier scans than the ones detected by TRW: *slow scans* (only one flow with a single defended IP address contacted on a single port), and *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 participants to annotate IP addresses with anomalous behaviours which are not malicious: *misconfigurations* and *backscatters*.

The participants answer the queries very differently. In particular, they disagree on the label corresponding to the *misconfigurations*. Some consider they are anomalous, while others think they are too common in network traffic. Besides, they annotate the instances with different levels of detail. Some create families based on combinations of services (*Web-DNS*, *Web-SMTP*, *Web-SMTP-DNS*, etc.), while others differentiate the services depending on whether the external IP address is the client or the server (*Web-from-external* and *Web-from-internal* for example). They build scan families with different levels of detail: *obvious-Syn-scans*, *obvious-Syn-scans-Reset-response*, *multihosts-multiports-scans*, *udp-multihosts-scans*, etc.

Table 2a presents the number of families created by each participant, and the number of families at the end of the annotation process after using the *Family Editor*. It reveals that the participants begin by creating specific families and they end up merging some of them to remove needless details. The participants have declared that the *Family Editor* and the *Annotated Instances* interfaces help them to provide consistent annotations throughout the annotation process. Furthermore, they have stated that these tools are crucial if the annotation process lasts several days, or if several annotators are involved.

Annotation Cost The cost of the annotation process is usually reported as the number of manual annotations (Almgren and Jonsson 2004; Görnitz et al. 2013; Stokes et al. 2008). However, all the annotations do not have the same cost in terms of time for making a decision and experts have their own annotation speed. Figures 2c and 2d present the average annotation cost, `i.e.`, the average time required to answer annotation queries, with the corresponding 95% confidence interval, for each participant, at each iteration and for each query type respectively.

Figure 2c shows that the annotation speed varies significantly from participant to participant. User 1 annotated particularly quickly: he had time to run three iterations during the 90 minutes while the other participants ran only two iterations. However, the annotation cost always decreases across iterations: they get used to the data they annotate and to the user interface and so they answer queries faster.

Uncertain queries, close to the decision boundary, are often considered harder to annotate (Baldridge and Palmer 2009; Settles 2011). The statistics presented in figure 2d support this statement for two participants out of four. This low agreement may be explained by the fact that we have run only a few iterations, and therefore the model has not yet converged and is still uncertain about instances easy to annotate for domain experts.

Figure 2d also points out that the benign queries are harder to annotate than the malicious ones for two out of four participants. One explanation is that computer security experts usually focus on malicious events, and as a result they are not used to analysing benign behaviours nor to group them into families.

Resulting Detection Model The performance of the resulting detection models are not evaluated automatically on a given validation dataset since the ground-truth cannot be established. Indeed, the participants have made different annotation choices both at the label and family levels.

The participants have analysed the alerts raised by their detection model on `Day 2` manually. They have noted that their detection model detects all the TRW alerts, and also stealthier anomalous behaviours which are consistent with their annotations. The top N alerts are obvious scans where many ports are scanned on many IP addresses. The randomly selected alerts correspond to the most common anomalies, `i.e.`, slow Syn scans on port 23.

The participants have pointed out that grouping alerts according to their predicted families ease their analysis, and reveal more interesting alerts, `i.e.` less common anomalous behaviours, than top N and random. They have reported that the assignment of families to alerts could be more accurate. Some families have been discovered only at the last iteration and therefore have too few examples to allow a good generalization of the classification model. The aim of the experiments was not to build a detection model ready for production, but to assess ILAB from a user point of view with an annotation procedure limited to an hour and a half. Building a detection model for production would require a longer annotation procedure and to stop the iterations when the number of families discovered has converged.

Short Waiting-Periods Table 2b presents an analysis of the cumulated computation times and waiting-periods throughout the whole annotation process. The duration of the generation of all the annotation queries by the active learning strategy is stored in the column *Computations*. The column *Waiting-Periods* corresponds to the cumulated waiting-periods : the time during which the users are waiting for the active learning strategy to computer new annotation queries.

The cumulated waiting-periods are smaller than the cumulated computation times since ILAB parallelizes the annotations and the computations: users can annotate some instances while the remaining annotation queries are computed. Users wait only while the detection model is updated, and the uncertain queries are generated. Then, they start answering the uncertain queries while the malicious and benign queries are generated. During our experiments, the computation of the malicious and benign queries has always been completed before the users have finished answering the uncertain queries. As a result, the participants have waited less than 5 seconds between each iteration. All the participants have declared that the waiting-periods are short enough not to damage the expert-model interaction.

ILAB design ensures a good expert-model interaction: the detection model is updated frequently with expert feedback without inducing long waiting-periods.

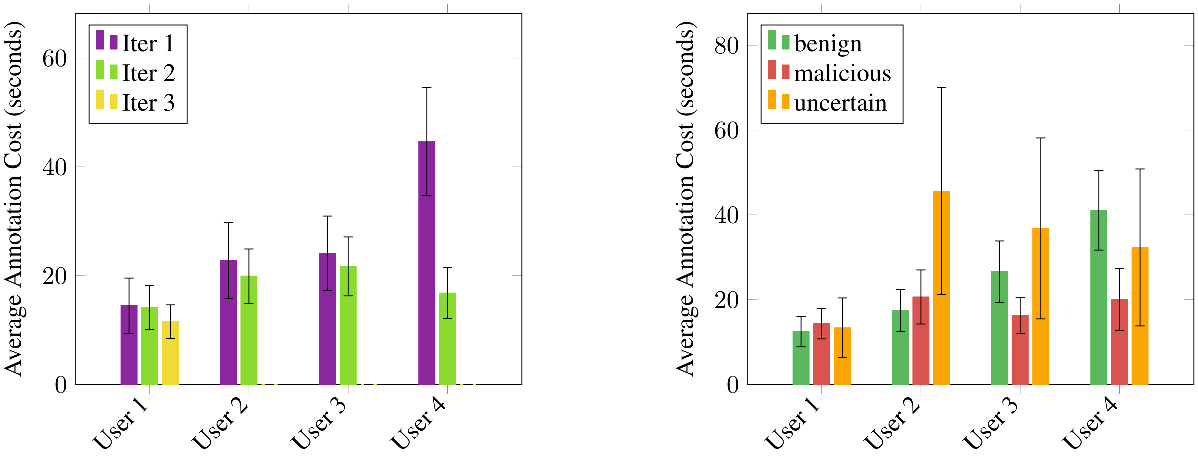
Feedback to Annotators The participants have appreciated the improvement of the detection model across iterations thanks to the clustering of the queries according to labels and families. They have assessed the false negatives while annotating the `Benign` queries, and the false positives while annotating the `Malicious` ones. Moreover,

User	# Queries	# Created Families	# Final Families
1	300	15	10
2	200	16	15
3	200	29	26
4	200	22	17

(a) Number of Created and Final Families

User	Whole Process	Computations	Waiting-Periods
1	1h28min	197.53 sec	10.81 sec
2	1h29min	91.56 sec	7.32 sec
3	1h36min	87.54 sec	7.31 sec
4	1h57min	93.56 sec	7.37 sec

(b) Computations and Waiting-Periods



(c) Average Annotation Cost across Iterations

(d) Average Annotation Cost for each Query Type

Figure 2: Results of the User Experiments

they have evaluated the relevance of the predicted families with the suggestions.

The two monitoring graphs displayed by ILAB, the number of discovered families and the precision of the suggestions, have not been mentioned by the participants as a means of seeing the benefit of their annotations. These graphs depict a global evolution over several iterations, while the method used by the participants allow to grasp local evolutions between two consecutive iterations. The number of iterations performed during the user experiments may be too low to show the relevance of these graphs.

Families Benefits We have noted that families help participants to provide consistent annotations throughout the annotation process, and afterwards they guide the analysis of the alerts of the resulting detection model. Families offer additional advantages that improve user experience.

In the first iteration, the malicious queries are selected uniformly among the malicious predictions since all the initial malicious instances belong to the *obvious scans* family. Two participants have reported that answering these queries is rather annoying: the annotation queries all have the same behaviour. On the contrary, when more families are discovered rare category detection is executed, and the participants have appreciated that the queries are clustered according to families, and present more diverse and less common behaviours. Two participants have relied on the aggregation of similar queries to get an overall view of the queries before making any decision.

Families provide a context that helps experts to answer the queries, and allow to query more interesting instances which exhibit less common behaviours.

Further User Feedback

Features Implications The first two participants had no information about the features of the detection model to hide the machine learning complexity. This lack of information led to the creation of families that could not be discriminated by the detection model. The first user ended up merging these too specific families, there was therefore no negative impact on the resulting detection model. On the contrary, the second user kept the specific families until the end of the annotation process which has damaged the performance of the detection model. The last two participants knew the features, and they did not create families that could not be discriminated properly by the detection model. They had no difficulty understanding the features included in the model, nor their implications on the families they might create. They, however, confirmed their desire to build more specific families that would necessitate additional features.

ILAB, as the state-of-the-art active learning strategies (Almgren and Jonsson 2004; Stokes et al. 2008; Görnitz et al. 2013), assumes that the features are set at the beginning of the annotation process and do not change across iterations. The user experiments have, however, shown that the discovery of new families may necessitate adding new features to allow the detection model to discriminate them properly. A new avenue of research is to consider active learning strategies where the features change across iterations. Human annotators could change the features manually, or they could be updated automatically according to the new annotations with a method similar to (Boullé 2014). In both cases, a particular care shall be taken to maintain short waiting-periods and to avoid numerical instabilities.

More Than Oracles Computer security experts involved in annotation projects are not mere oracles. At each iteration, they want to do more than answering queries: they want to understand why the model is uncertain about a prediction, and why it makes a mistake. It is therefore essential to tell them the features included in the detection model.

Finding the right compromise between hiding all the machine learning concepts and displaying a very complex monitoring is not an easy task. Besides, it largely depends on the profile of the annotators, whether they are simply annotating, or also involved in the deployment of the resulting detection model. ILAB graphical user interface allows to control this trade-off easily. On the one hand, the default view is accessible to computer security experts with little or no knowledge about machine learning. On the other hand, the modular construction of the interface allows to add more complex monitoring elements if annotators want to go more in depth into the evolution of the detection model.

Conclusion

We stress the importance to design an end-to-end active learning system, composed of an active learning strategy integrated in a user interface, to effectively reduce experts annotation effort. We have designed and implemented an end-to-end active learning system, ILAB, to bridge the gap between theoretical active learning and real-world annotation projects. We provide an open source implementation of the whole annotation system to foster research in this area, and to allow computer security experts to annotate their own datasets.

First, we present ILAB active learning strategy that is designed to minimize experts waiting-periods: it optimizes the time spent on annotating to compute queries. Then, we integrate this strategy in a flexible graphical user interface tailored to the needs of computer security experts. The custom visualization that can be plugged into the interface makes it generic regarding data types. Besides, the modular construction of the interface allows to meet the needs of the annotator. Finally, the user experiments show that ILAB in an efficient active learning system that can be deployed by computer security experts in real-world annotation projects. They also point out some avenues of research to further improve user experience in annotation projects.

References

Almgren, M., and Jonsson, E. 2004. Using active learning in intrusion detection. In *CSFW*, 88–98.

Amershi, S.; Lee, B.; Kapoor, A.; Mahajan, R.; and Christian, B. 2011. Cuet: human-guided fast and accurate network alarm triage. In *CHI*, 157–166.

Amershi, S.; Cakmak, M.; Knox, W. B.; and Kulesza, T. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35(4):105–120.

Ayache, S., and Quénot, G. 2008. Video corpus annotation using active learning. *Advances in Information Retrieval* 187–198.

Baldridge, J., and Palmer, A. 2009. How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation. In *EMNLP*, 296–305.

Beaugnon, A.; Chifflier, P.; and Bach, F. 2017. Ilab: An interactive labelling strategy for intrusion detection. In *RAID*.

Boullé, M. 2014. Towards automatic feature construction for supervised classification. In *ECML PKDD*, 181–196.

Görnitz, N.; Kloft, M. M.; Rieck, K.; and Brefeld, U. 2013. Toward supervised anomaly detection. *JAIR*.

Jung, J.; Paxson, V.; Berger, A. W.; and Balakrishnan, H. 2004. Fast portscan detection using sequential hypothesis testing. In *S&P*, 211–225.

Kulesza, T.; Amershi, S.; Caruana, R.; Fisher, D.; and Charles, D. 2014. Structured labeling for facilitating concept evolution in machine learning. In *CHI*, 3075–3084.

Lewis, D. D., and Gale, W. A. 1994. A sequential algorithm for training text classifiers. In *SIGIR*, 3–12.

Mac Aodha, O.; Stathopoulos, V.; Brostow, G. J.; Terry, M.; Girolami, M.; and Jones, K. E. 2014. Putting the scientist in the loop–accelerating scientific progress with interactive machine learning. In *ICPR*, 9–17.

Paxson, V. 1999. Bro: a system for detecting network intruders in real-time. *Computer networks* 31(23):2435–2463.

Pelleg, D., and Moore, A. W. 2004. Active learning for anomaly and rare-category detection. In *NIPS*, 1073–1080.

Schütze, H.; Velipasaoglu, E.; and Pedersen, J. O. 2006. Performance thresholding in practical text classification. In *CIKM*, 662–671.

Settles, B. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52(55-66):11.

Settles, B. 2011. From theories to queries: Active learning in practice. *JMLR* 16:1–18.

Skeppstedt, M.; Paradis, C.; and Kerren, A. 2017. Pal, a tool for pre-annotation and active learning. *JLCL* 31(1):91–110.

Stokes, J. W.; Platt, J. C.; Kravis, J.; and Shilman, M. 2008. Aladin: Active learning of anomalies to detect intrusions. *Technical Report. Microsoft Network Security Redmond, WA.*

Tomanek, K., and Olsson, F. 2009. A web survey on the use of active learning to support annotation of text data. In *ALNLP*, 45–48.

Wagstaff, K. L. 2012. Machine learning that matters. In *ICML*, 529–536.