End-to-End Active Learning for Computer Security Experts

Anaël Beaugnon
ANSSI, INRIA, ENS Paris
anael.beaugnon@ssi.gouv.fr

Pierre Chifflier
ANSSI
pierre.chifflier@ssi.gouv.fr

Francis Bach
INRIA, ENS Paris
francis.bach@ens.fr

ABSTRACT

Supervised detection models can be deployed in computer security detection systems to strengthen detection. However, acquiring a training dataset is particularly expensive in this context since expert knowledge is required to annotate. Some research works rely on active learning to reduce human effort, 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 computer security experts needs. We have designed the active learning strategy and the user interface jointly to effectively reduce 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.

CCS CONCEPTS

• **Security and privacy → Intrusion/anomaly detection and malware mitigation**; Usability in security and privacy; • **Human-centered computing → Interactive systems and tools**;

KEYWORDS

Active Learning, Interactive Systems, Anomaly Detection

ACM Reference Format:
Anaël Beaugnon, Pierre Chifflier, and Francis Bach. 2018. End-to-End Active Learning for Computer Security Experts. In *Proceedings of KDD 2018 Workshop on Interactive Data Exploration and Analytics (IDEA'18) (IDEA @ KDD'18)*. ACM, New York, NY, USA, 10 pages.

1 INTRODUCTION

The performance of supervised detection models, deployed in computer security detection systems, depends deeply on the quality of the training data. However, good training datasets are extremely difficult to acquire in the context of threat detection.

Some annotated datasets related to computer security are public, but they quickly become outdated and they often do not account for the idiosyncrasies of each deployment context. Besides, crowdsourcing cannot be exploited to get annotated datasets at low cost since the data are often sensitive and expert knowledge is required to annotate.

Security experts can deploy *annotation systems* to build representative training datasets *in-situ*. The annotation system picks some instances from a pool of unlabeled data originating from the deployment environment, displays them to security experts, and gathers their answers.

Security experts are essential for annotating but they are an expensive resource. The labeling process must thus exploit their time efficiently. *Active learning* [19] has been introduced in the machine learning community to reduce human effort. Most research works on active learning focus on query strategies to minimize the number of manual annotations. These works assume that annotators are mere oracles providing ground-truth labels while active learning is an interactive procedure where user experience should not be overlooked [3, 20, 22]. A user interface is needed to gather the annotations and it must be suitable for security experts who may have little or no knowledge about machine learning. Besides, some feedback must show the usefulness of their annotations, and they should not wait too long while the next annotation queries are computed.

In this paper, we define an *active learning system* as an *annotation system* that leverages an *active learning strategy* to select the instances to be annotated. It is crucial to design both components jointly to effectively reduce annotation effort and to foster the adoption of active learning in annotation projects [14]. Security experts do not want to minimize only the number of manual annotations, but the overall time spent annotating.

We introduce an end-to-end active learning system, ILAB, designed to help computer security experts to build annotated datasets with a reduced effort. We have described ILAB active learning strategy and extensively compared it to state-of-the-art methods [2, 8, 21] in [6]. In this paper, we integrate ILAB active learning strategy in an annotation system to bridge the gap between theoretical active learning and real-world annotation projects.

We make the following contributions:

- We integrate ILAB active strategy in an annotation system tailored to security experts needs. We have designed the graphical interface for annotators who may have little knowledge about machine learning, and it can manipulate any data type (e.g. PDF files, Android applications, or system event logs). Moreover, it helps security experts provide consistent annotations even if they delineate the detection target and the alert taxonomy as they annotate.
- We ask intended end-users, computer security experts, to use ILAB on a large unlabeled 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 [1] to foster comparison in future research works, and to allow computer security experts to annotate their own datasets.

IDEA @ KDD'18, August 20th, 2018, London, United Kingdom

The rest of the paper is organized as follows. First, Section 2 introduces the context and formalizes the problem, and Section 3 presents some related works. Then, Section 4 describes ILAB, an end-to-end active learning system. Finally, Section 5 explains the protocol of the user experiments, while the results are set out in Sections 6 and 7.

2 PROBLEM STATEMENT

2.1 Context

Pool-Based Active Learning. In this paper, we consider pool-based active learning [19] as a way to train detection models *in-situ*. The unlabeled pool is composed of unlabeled data acquired from deployment environments (e.g. files, network traffic captures, or logs).

Annotation: Binary Label and Family. Annotating consists in assigning a binary label, malicious or benign, and optionally a family detailing the binary label. Instances sharing the same family behave similarly and have the same level of criticality. 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 email messages coming from the same spam campaign.

The family information is critical in the context of threat detection. A binary answer indicating only whether an alert has been triggered is not satisfactory. Detection methods must provide information about the cause of alerts to ease their exploitation by security experts. Tagging alerts with malicious families can ease their analysis.

Detection Target and Alert Taxonomy. The annotations determine both the *detection target* and the *alert taxonomy* of the detection system. The binary labels determine the detection target, i.e. in which circumstances an alert should be triggered, while the malicious families establish the alert taxonomy, i.e. how the alerts are tagged.

The detection target and the alert taxonomy are generally not perfectly delineated at the beginning of annotation projects. Security experts have usually vague specifications in mind that they refine as they examine new instances queried by the active learning strategy.

2.2 Notations

Let $\mathcal{D} = \{x_i \in \mathbb{R}^m\}_{1 \leq i \leq n}$ be the dataset we want to annotate partially to train a supervised detection model $\mathcal{M}$. It contains n instances described by m real-valued features. For example, each instance $x \in \mathcal{D}$ could represent a PDF file or the traffic of an IP address.

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 $y \in \mathcal{L}$. Our aim is to create an annotated dataset

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

maximizing the accuracy of the detection model $\mathcal{M}$ trained on $\mathcal{D}_L$. The annotated dataset $\mathcal{D}_L$ is built with an iterative pool-based active learning strategy. At each iteration, a security expert annotates, with a label and a family, $b \in \mathbb{N}$ instances selected from the pool of remaining unlabeled instances denoted by $\mathcal{D}_U$. During the whole process, the expert cannot annotate more instances than the annotation budget $B \in \mathbb{N}$.

2.3 Objective

Our goal is to conceive an end-to-end pool-based active learning system tailored to security experts needs to streamline annotation projects. We assume that no adversary attempts to mislead the annotation process: a trusted security expert performs the annotations offline before the detection model is deployed in production.

The two components of the active learning system, the active learning strategy and the annotation system, must fulfill the following constraints to effectively reduce human effort in annotation projects.

Active Learning Strategy. The objective of the active learning strategy is to build the annotated dataset $\mathcal{D}_L$ that maximizes the accuracy of the detection model $\mathcal{M}$ while asking the expert to annotate at most B instances. Besides, the strategy must be scalable to work on large datasets while maintaining short waiting-periods.

The real challenge faced by active learning strategies is to avoid sampling bias [6, 17] to ensure a well-performing detection model, while keeping short waiting-periods to guarantee a good expert-model interaction.

Annotation System. The annotation system must provide an ergonomic user interface to streamline the annotation process. It must be suitable for non-machine learning experts since the intended end-users may have little or no knowledge about this data analysis technique.

First of all, the annotation system must provide an annotation interface to display and gather the answers to the annotation queries. It must be generic enough to be workable on any threat detection annotation project. As a result, the annotation interface should be able to display different data types such as PDF files, Windows Office documents, Android applications, or Windows event logs.

Moreover, the annotation system should not be reduced to an annotation interface. It should provide feedback frequently to show experts the benefit of their annotations, and that they are on track to achieve their goal. Besides, it should help security experts provide consistent annotations, even if they delineate the detection target and the alert taxonomy throughout the annotation process.

3 RELATED WORK

Specific Annotation Systems. Some annotation systems have been especially designed for text [7], image [11], or video [4] annotations. These annotation systems are not flexible enough to operate with all the data types processed by computer security detection systems.

Structured Labeling [11]. Machine learning is based on the idea that similar inputs should have similar outputs. Annotators must thus provide consistent labels to avoid degrading the performance of the resulting classification model. Kulesza et al. [11] have introduced *structured labeling* to help annotators define and refine their *concept*, i.e. the abstract notion of the target class annotators are labeling for, as they annotate data. Thanks to structured labels, annotators can organize their concept definition by grouping and tagging data. Structured labeling increases label consistency by helping annotators recall labeling decisions. The structure is

malleable (annotators can create, delete, split and merge tags), it is well suited for situations where annotators are likely to frequently refine their concept definition as they observe new data.

In the context of detection systems, the *concept* corresponds to the detection target, i.e. the abstract notion of benign behavior, and suspicious behaviors that should trigger alerts. Besides, we can draw a parallel between the tags defined in structured labeling and the benign and malicious families. Structured labeling can be very convenient in annotation projects aiming to build computer security detection models. Indeed, at the beginning of annotation projects, security experts have a vague idea of their detection target, and it may evolve throughout the annotation process. Some annotation queries may puzzle them: they may wonder whether an alert should be triggered or not. Some annotation queries can even question previous annotations.

Active Learning vs. Random Sampling. There are some reassessment about the benefit of active learning strategies over random sampling [7]. Some consider it is not worth deploying active learning strategies in annotation systems: it may be complex and lead to a computation overhead. Computer vision and natural language processing annotation projects can take advantage of low-cost annotators on crowd-sourcing market places. In this scenario, there is no need for active learning since annotations are cheap. However, crowd-sourcing does not suit detection systems since the data they process are often sensitive, and annotating requires expert knowledge.

Moreover, the data is often unbalanced, with a tiny portion of malicious instances, and the less common malicious behaviors are often the most interesting. If the annotation system queries instances selected uniformly, it is likely to query only benign and very common malicious behaviors.

To sum up, in the context of threat detection, annotation systems must leverage an active learning strategy to be effective. The active learning strategy should, nevertheless, be designed carefully to minimize the computation overhead.

Applications to Computer Security. Stream-based active learning [19] has been applied to computer security detection problems [18, 23] to follow threat evolution. In this setting, the detection model in production has been initially trained on an annotated dataset representative of the deployment environment. In our case, such a representative annotated dataset is unavailable and the objective is to acquire it offline to train the initial detection model.

Some works focus on pool-based active learning to build annotated datasets for detection systems [2, 6, 8, 21]. However, most research works [2, 8] have only run simulations on fully annotated datasets: an oracle answers the annotation queries automatically with the ground-truth labels. They have not set up their strategy in real-world annotation projects, and they have not mentioned any user interface.

Stokes et al. [21] have carried out user experiments with computer security experts. Aladin includes a graphical user interface but the authors do not provide any detail about it. Besides, the interactions between the expert and the model are poor due to a high execution time. The expert is asked to annotate a thousand instances each day, and new queries are computed every night. Their solution reduces the waiting-periods, but it significantly damages the expert-model interaction since the expert feedback is integrated only once a day.

In brief, the user experience is often overlooked: research works focus mostly on active learning strategies and not on their integration in annotation systems. User interfaces designed to set up active learning strategies in real-world annotation projects have, however, a significant impact on the overall user experience [3, 19, 20] and on the actual application of such methods in practice [14].

4 ILAB

4.1 Active Learning Strategy

ILAB active learning strategy has been presented in [6] and compared to state-of-the-art methods [2, 8, 21]. 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 annotation 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 may result in longer waiting-periods and a loss of understandability.

Once $\mathcal{M}$ has been trained, the b annotation queries are computed: 1) the $b_{\texttt{uncertain}}$ instances the closest to the decision boundary are queried as in uncertainty sampling [13], and 2) $b_{\texttt{families}}$ ($= b - b_{\texttt{uncertain}}$) instances are queried by means of rare category detection [16].

Not all families are present in the initial annotated 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 threat 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 trained 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 modeled 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, rare category detection takes it into account at the next iteration: the additional family is included in the training of the multi-class logistic regression and the Gaussian Naive Bayes models.

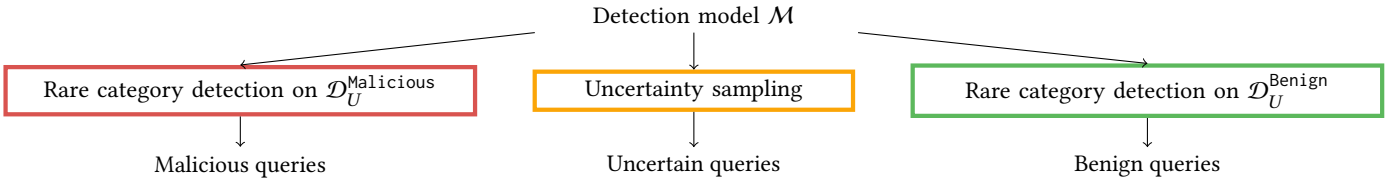


Figure 1: Parallelization of the Computations of the Annotation Queries.

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.

4.2 Annotation System

ILAB annotation system obviously includes an *Annotation Interface* to display and gather the answers to the annotation queries. Moreover, it offers additional graphical user interfaces to ease data annotation: a *Monitoring Interface*, a *Family Editor* and an *Annotated Instances Interface*.

4.2.1 Annotation Interface. Security experts answer ILAB queries from the graphical user interface depicted in Figure 2. It is intended for non-machine learning experts, and the layout of the panels is designed to ensure a logical reading order. Experts can select a type of queries with one of the top buttons: *Uncertain* for the instances near the decision boundary, *Malicious* and *Benign* for the queries generated by rare category detection. The *Annotation Queries* panel displays the queries. Malicious and benign queries are grouped by families. The bottom panel displays the queried instances (*Description* panel), and gathers the annotations (*Annotation* panel).

Description Panel. The *Description* panel contains information about the instance that the security expert must annotate. It consists of a standard visualization depicting the instance features, and of optional problem-specific visualizations. Figure 2 shows the custom visualization we have implemented for NetFlow data[1].

We strongly encourage security experts to design and implement convenient problem-specific visualizations, since they can considerably ease the annotations. They should display the most relevant information to help annotators make decisions. Security experts can implement several custom visualizations to show the instances from different angles.

Annotation Panel. 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 one.

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, ILAB makes 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 [5].

[1]We have hidden the IP addresses for privacy reasons.

The next query is displayed automatically after each annotation validation. Experts can click on the `Next Iteration` button to generate the next queries after answering all the queries of the current iteration. If some queries have not been answered, a pop-up window asks the annotator to answer them.

4.2.2 Monitoring Interface. ILAB *Monitoring Interface* (see Figure 3) displays information about the current detection model (coefficients of the logistic regression model, and performance indicators computed on the annotated dataset through cross validation), and feedback about the annotation progress.

Annotation systems must provide feedback to experts to show them the benefit of their annotations, and that they are on track to achieve their goal [3]. In simulated experiments, where an oracle answers the queries automatically with the ground-truth labels, the performance of the detection model $\mathcal{M}$ on an independent validation dataset is usually reported. Nevertheless, this approach is not applicable in a real-world setting: when security experts deploy an annotation system to build a training dataset they do not have access to an annotated validation dataset.

ILAB displays two kinds of feedback that do not require an annotated validation dataset: 1) the number of malicious and benign families discovered so far, and, 2) the accuracy of the suggested labels and families. At each iteration, ILAB suggests a family for the high likelihood queries. At the next iteration, ILAB computes the accuracy of these suggestions 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, the security expert may stop annotating.

4.2.3 Annotated Instances and Family Editor. ILAB offers two user interfaces to help security experts refine the detection target and the alert taxonomy while remaining consistent with their previous annotations: a *Family Editor* and an *Annotated Instances Interface*.

Family Editor. The family editor, similar to the one introduced by Kulesza et al. [11], enables annotators to perform three actions over the families:

(1) *Change Name* to clarify the name of a family ;
(2) *Merge Families* to regroup similar families ;
(3) *Swap Malicious / Benign* to change the label corresponding to a given family.

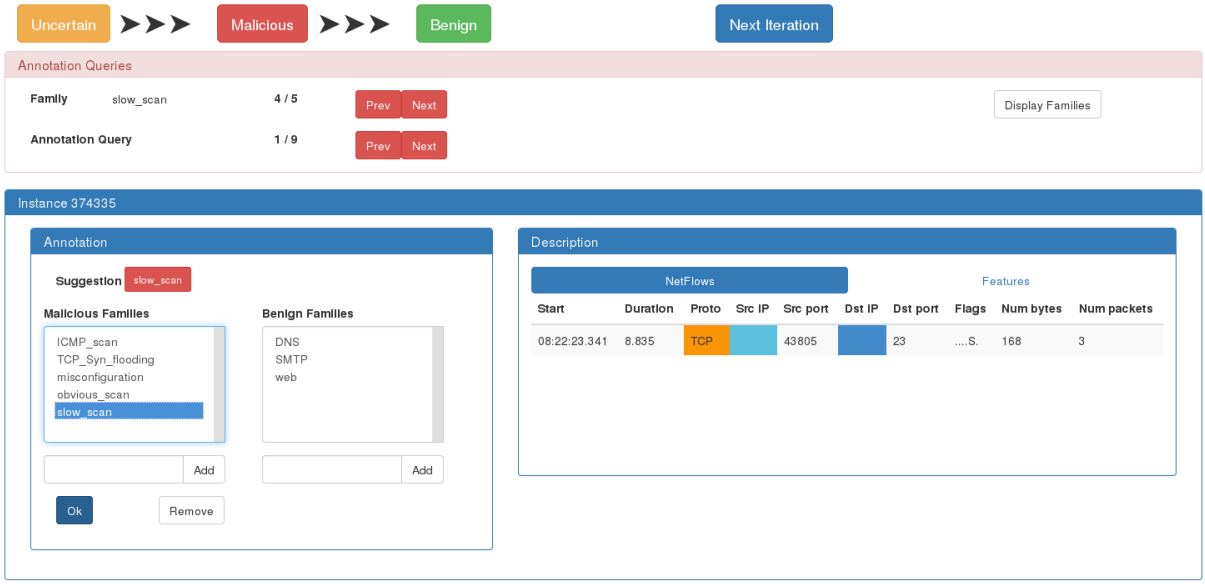


Figure 2: ILAB Annotation Interface.

Annotated Instances Interface. This interface enables experts to review their previous annotations. It displays the currently annotated instances grouped according to their associated label or family.

Security experts can leverage this interface to examine the instances of a given family, or to rectify previous annotations. Thanks to the *Family Editor*, they can perform high-level changes on the families, but they cannot split them. They can split a family thanks to the *Annotated Instances Interface* by going through all its instances and updating the annotations.

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.

4.3 Deployment

4.3.1 Settings of the Parameters. Security experts need to set three parameters to deploy ILAB in annotation projects. First, all active learning strategies share two parameters: 1) the global annotation budget B, and 2) the number of annotation queries answered at each iteration, b. Besides, ILAB strategy has one specific parameter, $b_{\texttt{uncertain}}$, the number of uncertain queries.

Setting the Parameter b. This parameter controls the trade-off between reducing waiting-periods and integrating expert feedback frequently. One the one hand, simulations where oracles answer annotation queries with ground-truth labels are often carried out with $b = 1$. This setting does not suit real-world annotation projects since it would induce too frequent waiting-periods for security experts. One the other hand, Stokes et al. [21] have set b to 1000 in their user experiments. This high iteration budget damages the expert-model interaction : security experts spend their day annotating, and their feedback is taken into account only every night to improve the detection model.

Security experts should set the value of the parameter b 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. The value of b is therefore data dependent: it must be set according to the average time required to answer annotation queries.

Setting the Parameter $b_{\texttt{uncertain}}$. This parameter fixes the portion of the iteration budget b dedicated to uncertain queries. 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 [20], and they may lead to sampling bias [17].

Setting the Parameter B. The global annotation budget B specifies the stop condition of the annotation process. It can be set to a maximum number of annotations, or to a global time spent annotating.

In the case of an unlimited budget, security experts can end the annotation process when convergence is reached: several iterations have not led to the discovery of a new family, and the model predictions are consistent with the expert annotations. ILAB *Monitoring Interface* informs security experts about the state of convergence of the annotation procedure with the number of families discovered and the accuracy of the suggested annotations.

4.3.2 Initialization. The active learning process needs some initial annotated instances to train the first supervised detection model. This initial supervision can be difficult to acquire for computer security detection problems. The `Malicious` class is usually

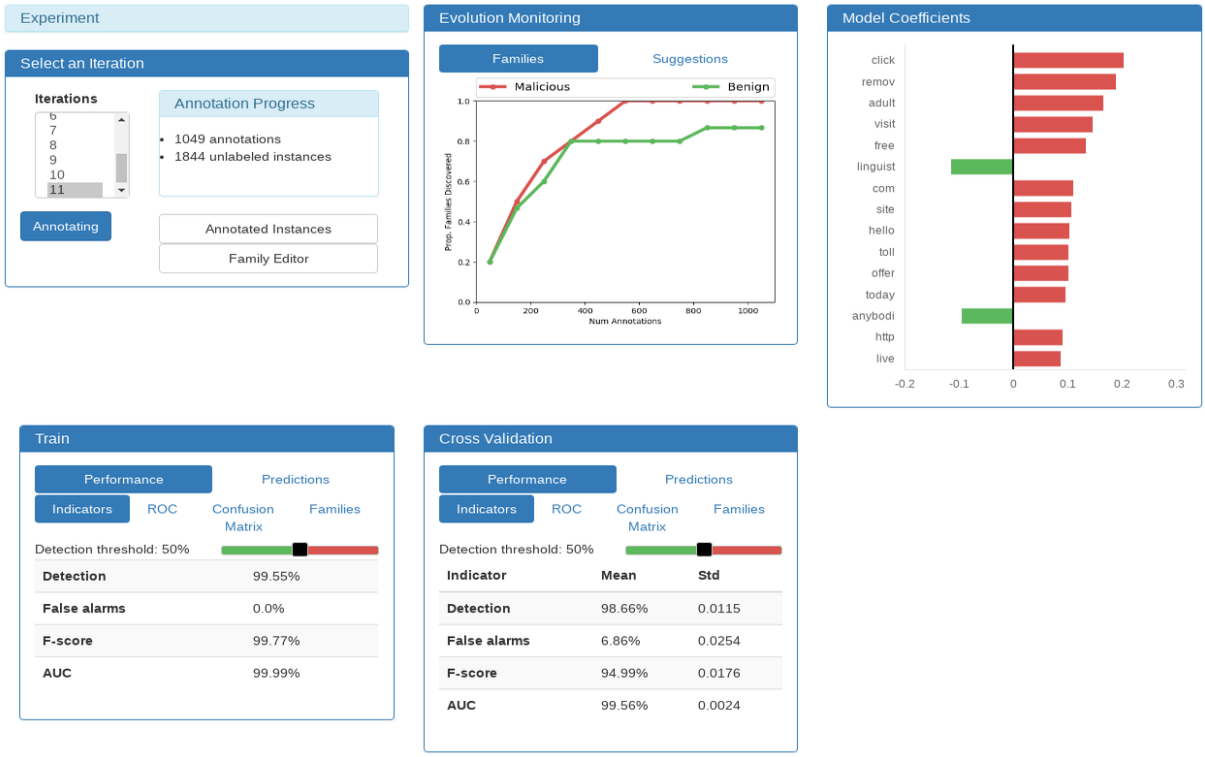


Figure 3: ILAB Monitoring Interface.

underrepresented for uniform random sampling to be effective at collecting a representative annotated dataset.

If a public annotated dataset is available for the detection problem considered, it can serve as initial supervision. Otherwise, misuse detection techniques widely deployed in detection systems can provide `Malicious` examples at low cost, and random sampling can provide `Benign` examples. In both cases, the initial annotated 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 annotated dataset with more diverse malicious behaviors and to make it representative of the environment where the detection system is deployed.

5 USER EXPERIMENTS

In this section, we ask security experts to use ILAB to acquire an annotated dataset from unlabeled NetFlow data. The primary objective is to collect feedback from intended end-users to validate our design choices. Another objective is to highlight possible improvements that will be beneficial to other annotation projects.

The competing active learning methods [2, 8, 21] compared to ILAB with simulations in [6] have not designed or they provide too few details about the user interface. As a result, they are not considered during the user experiments.

	Day 1	Day 2
Number of flows	$1.2 \cdot 10^8$	$1.2 \cdot 10^8$
Number of IP addresses	463,913	507,258
Number of features	134	134
Number of TRW alerts	72	82

Table 1: NetFlow Datasets

5.1 Annotation Project

The annotation project consists in acquiring an annotated dataset from unlabeled NetFlow data in order to train a supervised detection model detecting IP addresses with an anomalous behavior.

The flows are recorded at the border of a defended network. They are described by attributes and summary statistics: source and destination IP addresses, source and destination ports, protocol (e.g. TCP, UDP, ICMP, ESP), 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 24-hour 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 (e.g. `....S`, `.A..S.`, `.AP.SF`).

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 are recorded during two consecutive working days in 2016 (see Table 1). The `Day 1` dataset constitutes the unlabeled pool from which some instances are queried for annotation, and the `Day 2` dataset serves as a validation dataset to analyze the alerts triggered by the resulting detection model.

The active learning process is initialized with some annotated instances. The alerts triggered by the Threshold Random Walk (TRW) [10] module of Bro [15] provide the initial anomalous examples and the normal examples are drawn randomly. All the initial annotations are checked manually. The initial annotated 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 annotated dataset. The detection model should not be restricted to the detection of obvious scans, additional anomalous behaviors must be identified from the NetFlow data with ILAB.

5.2 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, and are divided into three parts. First, the users acquire an annotated dataset with ILAB from the unlabeled pool `Day 1`. Second, they analyze the alerts triggered on `Day 2` by the detection model trained on the annotated instances with DIADEM [1]. Finally, we collect their feedback.

All the experiments are run on a dual-socket computer with 64 Go RAM. Processors are Intel Xeon E5-5620 CPUs clocked at 2.40 GHz with 4 cores each and 2 threads per core. We timestamp and log all the users' actions in ILAB graphical interface to assess the time required for annotating and the waiting-periods.

ILAB Deployment. We set the parameters of ILAB active learning strategy according to the guidelines presented in Section 4.3.1: $b = 100$ and $b_{\texttt{uncertain}} = 10$. We do not set the global annotation budget B to a number of manual annotations, but we stop the annotations after 90 minutes while letting annotators complete their current iteration.

The first two participants have no information about the features of the detection model for the purposes of hiding the machine learning complexity. This approach may lead annotators to create families that the detection model cannot properly discriminate due to a lack of information about 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* (port 23) or *SSH scans* (port 22).

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.

Alerts Analysis. Once experts have annotated a dataset with ILAB, we leverage DIADEM [1] to apply the resulting detection model to `Day 2` data. Then, the security expert analyzes the alerts triggered on `Day 2` data from DIADEM alert visualization interface. This step is crucial: the objective of 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 triggered 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.

6 VALIDATION OF ILAB DESIGN

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. We will address these issues 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.

Detection Target and Alert Taxonomy. Across the iterations, ILAB has queried 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 behaviors which are not malicious: *misconfigurations* and *backscatters*.

To sum up, the detection target has evolved across the iterations thanks to ILAB queries. At the beginning of the annotation process, the annotated dataset contains only obvious scan activities, and ILAB queries other anomalous behaviors. The rare category detection analyses carried out by ILAB are effective for pointing out new anomalous behaviors.

Table 4a presents the number of families created by each participant, and the number of families at the end of the annotation process. It reveals that the participants begin by creating specific families and they end up merging some of them with the *Family Editor* to remove needless detail. The participants have declared that the *Family Editor* and the *Annotated Instances Interface* help them provide consistent annotations throughout the annotation

process. Furthermore, they have stated that these tools are crucial if the annotation process lasts several days.

In brief, the participants rely on ILAB to define the malicious and benign families. At the beginning of the annotation project, they have a vague idea of how to group the benign and malicious behaviors into families. Then, ILAB queries bring them to change their families definitions. The user experiments show that the *Family Editor* and the *Annotated Instances Interface* are critical components of ILAB to define the families interactively.

Annotation Cost. The cost of the annotation process is usually reported as the number of manual annotations [2, 8, 21]. 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. We report 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 (see Figure 4c) and for each query type (see Figure 4d).

Figure 4c shows that the annotation speed varies significantly from participant to participant, and the annotation cost always decreases across the iterations. The experts get used to the data they annotate and to ILAB user interface and so they answer queries faster. Moreover, they get a more precise idea of the detection target and of the malicious and benign families as they annotate, so they spend less time making decision.

Uncertain queries, close to the decision boundary, are often considered harder to annotate [5, 9, 20]. The statistics presented in figure 4d support this statement for only 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 security experts.

Figure 4d 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 security experts are not used to analyzing benign behaviors and to group them into families. They analyze malicious behaviors when they design misuse detection techniques, and they are accustomed to grouping malicious behaviors into families when they define alert taxonomies.

Resulting Detection Model. The participants have assessed that the triggered alerts on `Day 2` are consistent with their malicious annotations and that the number of false positives is low enough to meet operational constraints. 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 malicious families eases their analysis, and reveals more interesting alerts, i.e. less common malicious behaviors, than top N and random. The families of some alerts have, however, been wrongly predicted due to a lack of annotated instances for some malicious families. Some families have been discovered only at the last iteration and too few examples are in the annotated dataset for the detection model to generalize properly. More iterations are required to improve the automatic qualification of the alerts.

Short Waiting-Periods. Table 4b presents an analysis of the cumulated computation times and waiting-periods throughout the whole annotation process. The column *Computations* stores the duration of the computation of all the annotation queries. The column *Waiting-Periods* corresponds to the cumulated waiting-periods : the time during which the users are waiting for the active learning strategy to compute new annotation queries. *Efficiency* represents the percentage of time allocated to the improvement of the detection model (annotating, editing families, inspecting annotated instances) during the annotation process.

The cumulated waiting-periods are smaller than the cumulated computation times since ILAB parallelizes the annotations and the computations: experts can annotate some instances while the remaining annotation queries are computed. Experts wait only while the detection model is trained on the current annotated instances, and the uncertain queries are generated. Then, they start answering the uncertain queries while ILAB generates the malicious and benign queries. During our experiments, ILAB has always completed the computation of the malicious and benign queries before the experts 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 divide-and-conquer approach ensures a good expert-model interaction: the detection model is updated frequently with expert feedback without inducing long waiting-periods.

Feedback to Annotators. Three out of the four participants have declared that they have perceived the benefit of their annotations across the iterations. In their view, they appreciate the improvement of the detection model thanks to the clustering of the queries according to labels and families. They assess the false negatives while annotating the `Benign` queries, and the false positives while annotating the `Malicious` ones. Moreover, they evaluate the relevance of the predicted families with the suggestions.

The participants have not mentioned the two feedback graphs displayed by ILAB (the number of discovered families and the accuracy of the suggestions), 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 grasps 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.

7 FURTHER FEEDBACK

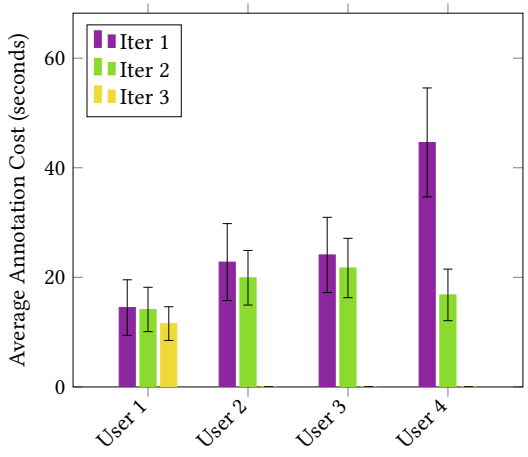
Security Experts Annotate Differently. The participants answer the queries very differently. In particular, they disagree on the label corresponding to the *misconfigurations*. Some consider they are anomalous, while others think they are too common in network traffic. Besides, they annotate the instances with different levels of detail. Some create families based on combinations of services (e.g. *Web-DNS*, *Web-SMTP*, *Web-SMTP-DNS*), while others differentiate the services depending on whether the external IP address is the client or the server (e.g. *Web-from-external* and *Web-from-internal*). In short, at the end of the annotation project, the participants have neither the same detection target, nor the same alert taxonomy.

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

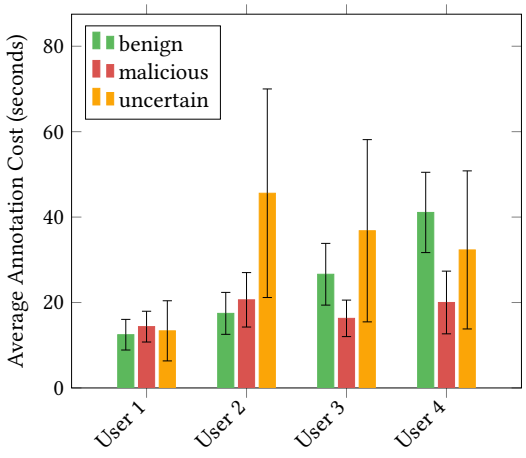
(a) Number of Created and Final Families.

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 4: Results of the User Experiments.

These discrepancies are not surprising since the participants have annotated independently, but we can draw lessons from these user experiments for future annotation projects involving several annotators. The main difficulty with several annotators is the definitions of the detection target and the alert taxonomy. They are usually not well delineated at the beginning of annotation projects, and security experts refine their specifications as they annotate queried instances.

One way is to ask the annotators to agree on the detection target and the alert taxonomy during a preliminary stage. During the first iterations, the annotators answer the annotation queries together. Once the iterations do not lead to the identification of new families, we can stop the preliminary stage. At the end of the preliminary stage, the detection target and the alert taxonomy are better delineated. The annotators can then use ILAB alternately to gather more annotations. If they discover a new family, or they are uncertain about an annotation, they can consult each other to make decision. An interesting avenue of research is to adapt ILAB to work with multi-annotators operating in parallel.

Data Annotation and Feature Extraction are Intertwined. The first two participants have no information about the features of the detection model to hide the machine learning complexity. This lack of information has led to the creation of families that the detection model could not discriminate. The first participant has ended up merging these too specific families, there has therefore been no negative impact on the resulting detection model. On the contrary, the second participant has kept the specific families until the end of the annotation process. It has damaged the performance of the detection model.

The last two participants know the features, and they have not created families that the detection model could not discriminate properly by the detection model. They have had no difficulty understanding the features included in the model, nor their implications on the families they can create. They have, however, confirmed their desire to build more specific families that necessitate additional features.

ILAB, as the state-of-the-art active learning strategies, assumes that the features are set at the beginning of the annotation process and do not change across the iterations. The user experiments have, nevertheless, shown that the discovery of new families may necessitate adding new features so that the detection model discriminates them properly. The detection target and the alert taxonomy are usually not well delineated at the beginning of the annotation process, so it is hard to anticipate which features should be extracted.

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 [12]. In both cases, a particular care shall be taken to maintain short waiting-periods and to avoid numerical instabilities.

Security Experts are More Than Oracles. Annotation projects where security experts build detection models differ significantly from crowd-sourcing annotation projects. Security experts involved in annotation projects are not mere oracles. At each iteration, they do more than answering queries. They delineate the detection target

and the alert taxonomy, and they want to understand the behavior of the detection model.

During the user experiments, some participants have wondered why the detection model is uncertain or wrong about a prediction. Security experts are willing to follow the evolution of the detection model across the iterations.

In order to address this need, ILAB annotation system is not reduced to an *Annotation Interface*. It also offers a *Monitoring Interface* to help security experts understand the behavior of detection models. Moreover, the *Family Editor* and the *Annotated Instances Interface* assist security experts in delineating the detection target and the alert taxonomy.

To sum up, ILAB active learning system is not a mere annotation system, it helps security experts build detection models interactively.

8 CONCLUSION

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 to foster research in this area, and to allow computer security experts to annotate their own datasets [1].

First, we present ILAB active learning strategy that minimizes 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 security experts needs. Our user experiments show that ILAB in an efficient solution that security experts can deploy in real-world annotation projects. They also point out some avenues of research to further improve user experience.

REFERENCES

[1] 2018. SecuML. https://github.com/ANSSI-FR/SecuML. (2018).
[2] Magnus Almgren and Erland Jonsson. 2004. Using active learning in intrusion detection. In *CSFW*. 88–98.
[3] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *AI Magazine* 35, 4 (2014), 105–120.
[4] Stéphane Ayache and Georges Quénot. 2008. Video corpus annotation using active learning. *Advances in Information Retrieval* (2008), 187–198.
[5] Jason Baldridge and Alexis Palmer. 2009. How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation. In *EMNLP*. 296–305.
[6] Anaël Beaugnon, Pierre Chifflier, and Francis Bach. 2017. ILAB: An Interactive Labelling Strategy for Intrusion Detection. In *RAID*.
[7] Seamus Clancy, Sam Bayer, and Robyn Kozierok. 2012. *Active Learning with a Human In The Loop.* Technical Report. MITRE CORP BEDFORD MA.
[8] Nico Görnitz, Marius Micha Kloft, Konrad Rieck, and Ulf Brefeld. 2013. Toward supervised anomaly detection. *JAIR* (2013).
[9] Ben Hachey, Beatrice Alex, and Markus Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *CoNLL*. 144–151.
[10] Jaeyeon Jung, Vern Paxson, Arthur W Berger, and Hari Balakrishnan. 2004. Fast portscan detection using sequential hypothesis testing. In *S&P*. 211–225.
[11] Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. 2014. Structured labeling for facilitating concept evolution in machine learning. In *CHI*. 3075–3084.
[12] Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. 2017. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327* (2017).
[13] David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR*. 3–12.
[14] Oisin Mac Aodha, Vassilios Stathopoulos, Gabriel J Brostow, Michael Terry, Mark Girolami, and Kate E Jones. 2014. Putting the Scientist in the Loop–Accelerating Scientific Progress with Interactive Machine Learning. In *ICPR*. 9–17.
[15] Vern Paxson. 1999. Bro: a system for detecting network intruders in real-time. *Computer networks* 31, 23 (1999), 2435–2463.
[16] Dan Pelleg and Andrew W Moore. 2004. Active learning for anomaly and rare-category detection. In *NIPS*. 1073–1080.
[17] Hinrich Schütze, Emre Velipasaoglu, and Jan O Pedersen. 2006. Performance thresholding in practical text classification. In *CIKM*. 662–671.
[18] D Sculley, Matthew Eric Otey, Michael Pohl, Bridget Spitznagel, John Hainsworth, and Yunkai Zhou. 2011. Detecting adversarial advertisements in the wild. In *KDD*. 274–282.
[19] Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52, 55-66 (2010), 11.
[20] Burr Settles. 2011. From theories to queries: Active learning in practice. *JMLR* 16 (2011), 1–18.
[21] Jack W Stokes, John C Platt, Joseph Kravis, and Michael Shilman. 2008. Aladin: Active learning of anomalies to detect intrusions. *Technical Report. Microsoft Network Security Redmond, WA* (2008).
[22] Kiri L Wagstaff. 2012. Machine Learning that Matters. In *ICML*. 529–536.
[23] Colin Whittaker, Brian Ryner, and Marria Nazif. 2010. Large-Scale Automatic Classification of Phishing Pages.. In *NDSS*, Vol. 10.